Optimization Oct 14 - Oct 16, 2024

Lectures 7-8: Integer Programming

Dan A. Iancu

In this lecture, we consider the following type of optimization problems:

$$\begin{aligned} \min\ & c^{\intercal}x + d^{\intercal}y \\ & Ax + By = b \\ & x, y \geq 0 \\ & x \text{ integer} \end{aligned}$$

We call this problem a **mixed integer programming** problem. If there are no continuous problem, then we simply call this an integer program (IP). Moreover, if x is further constrained to take value in $\{0,1\}^n$, we call this a zero-one or **binary** optimization problem. As we will soon see, these problems offer a very powerful modeling framework, but the downside is that they are generally hard to solve to optimality. We then discuss a few special cases when these problems are solvable (essentially, as easy as LPs) and we highlight some of the algorithms used to tackle these problems in full generality.

1 Modeling Techniques

Integer programming offers a very rich modeling framework. Here are some examples to illustrate this.

1.1 Binary choice

A binary variable can represent one of two alternatives. For instance, consider the classical **knapsack problem**.

Example 1 (The zero-one knapsack problem)**.** *We are given n items. The j-th item has weight w_j and its value/reward is r_j. Given a bound K on the weight that can be carried in a knapsack, we would like to select items to maximize the total value.*

To model this problem, we define a binary variable x_j which is 1 if item j is chosen, and 0 otherwise. The problem can then be formulated as follows:

$$\begin{aligned} \text{maximize}\ & \sum_{j=1}^{n} r_j x_j \\ \text{subject to}\ & \sum_{j=1}^{n} w_j x_j \leq K \\ & x_j \in \{0,1\}, \quad j = 1, \ldots, n. \end{aligned}$$

1.2 Logical constraints

We already saw several examples of logical constraints in our third class. Here, we briefly remind you of some of the basic building blocks. Suppose that we have activities/projects A and B and we use binary variables with the same name to indicate whether each activity is conducted; so $A = 1$ if and only if activity A is done. Then:

- to impose the condition: "if activity A is done, then activity B should also be done," we should add the constraint $A \leq B$. This exactly implements the **logical "or"** between the two projects: note that the condition that A or B should be done means $A + B \geq 1$, which is exactly equivalent to our constraint.

- To implement the logical **not**, we can use $1 - A$. That is, A **is not done** if and only if $1 - A = 1$.

- To implement the logical **and** – for instance, to create the binary variable $Z = A \cdot B$ – we can add the constraints:

$$Z \leq X,\ Z \leq Y,\ Z \geq X + Y - 1.$$

Moreover, if x is an n-dimensional vector of continuous or discrete decisions and $a \in \mathbb{R}^n, b \in \mathbb{R}$, to implement the condition that

$$Y = 1 \quad \Leftrightarrow \quad a^\intercal x + b \geq 0,$$

we should add the two constraints:

$$\begin{aligned} &a^\intercal x + b \geq m \cdot (1–Y) \\ &a^\intercal x + b + \epsilon \leq (M + \epsilon) \cdot Y, \end{aligned}$$

where m and M are the smallest and largest value, respectively, that $a^\intercal x + b$ can take over any feasible x, and ϵ is a very small tolerance parameter. The parameter arises because with continuous variables x, it is impossible to enforce precisely a strict inequality. However, if $x \in \mathbb{Z}^n$, the epsilon can be replaced with a finite value to obtain an exact reformulation.

Facility Location. As a classical example of logical constraints in practice, consider the facility location problem where we have n potential locations and m clients who need service. There is a fixed cost c_j for opening a facility at location j and a cost d_{ij} for serving client i from facility j. The goal is to select a set of facility locations and assign each client to one of the facilities at minimum cost. (For a visualization, see Figure 1.)

$$\begin{aligned} \min\ & \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} \\ & \sum_{j=1}^{n} x_{ij} = 1 \\ & x_{ij} \leq y_j \\ & x_{ij}, y_j \in \{0, 1\} \end{aligned}$$

Here, the equality constraint $\sum_{j=1}^{n} x_{ij} = 1$ model the requirement that each client i is exactly matched with one of the facilities j, and $x_{ij} \leq y_j$ ensures that clients can only be matched with open facilities.

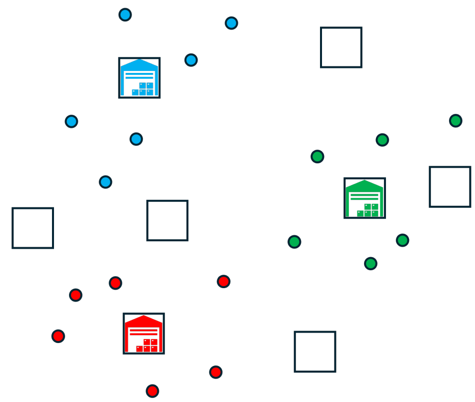


Figure 1: A facility location problem. Black squares denote potential locations for facilities, circles denote customers, and the cost function is Euclidean distance. The color-coding denotes the match between the three open facility and the customers.

1.3 Restricted range of values

Suppose we need to restrict a variable x to take values in a set $\{a_1, \ldots, a_m\}$. This can be achieved with the following constraints:

$$x = \sum_{j=1}^{m} a_j y_j, \quad \sum_{j=1}^{m} y_j = 1, \quad y_j \in \{0, 1\}.$$

1.4 Arbitrary piecewise linear cost functions

Binary variables allow reformulating an arbitrary piecewise-linear cost function. Suppose that $a_1 < a_2 < \cdots < a_k$ and that we have a continuous[1] piecewise linear function $f(x)$ specified by the points $(a_i, f(a_i))$ for $i = 1, \ldots, k$, defined on the interval $[a_1, a_k]$ (see Figure 2). Then, any $x \in [a_1, a_k]$ can be expressed in the form

$$x = \sum_{i=1}^{k} \lambda_i a_i,$$

where $\lambda_1, \ldots, \lambda_k$ are nonnegative scalars that sum to one.

Importantly, although the choice of coefficients $\lambda_1, \ldots, \lambda_k$ used to represent a particular x is not unique, this becomes unique if we require that at most two consecutive coefficients λ_i can be nonzero. In this case, any $x \in [a_i, a_{i+1}]$ is represented uniquely as $x = \lambda_i a_i + \lambda_{i+1} a_{i+1}$, with $\lambda_i + \lambda_{i+1} = 1$, and

$$f(x) = \sum_{i=1}^{k} \lambda_i f(a_i).$$

We also need to model the additional constraint that at most two consecutive coefficients λ_i are nonzero. To this effect, we consider a binary variable y_i, $i = 1, \ldots, k-1$, which can be

[1] Discontinuous functions can also be readily accommodated by introducing additional constraints.

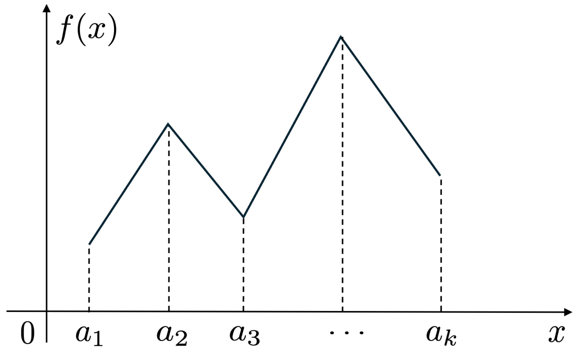


Figure 2: A piecewise linear cost function.

equal to 1 only if $a_i \leq x \leq a_{i+1}$, and must be 0 otherwise. The problem is then formulated as the following mixed integer programming problem:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{k} \lambda_i f(a_i) \\
\text{subject to} \quad & \sum_{i=1}^{k} \lambda_i = 1, \\
& \lambda_1 \leq y_1, \\
& \lambda_i \leq y_{i-1} + y_i, \quad \forall i = 2, \ldots, k-1, \\
& \lambda_k \leq y_{k-1}, \\
& \sum_{i=1}^{k-1} y_i = 1, \\
& \lambda_i \geq 0, \quad y_i \in \{0, 1\}, \quad \forall i.
\end{aligned}
$$

Notice that if $y_j = 1$, then $\lambda_i = 0$ for i different than j or $j+1$.

These constraints are so important in practice that they bear a special name: ***special ordered sets (SOS) of type 2)****. When adding an SOS constraint of type 2, you just need to specify a list of non-negative decision variables of which at most two can be non-zero and these must be consecutive in their ordering. (In our formulation above, these would be the ordered list of λ variables, λ_1, λ_2, etc.) Several modeling languages support adding such constraints directly and would handle the addition of any necessary binary variables and constraints "behind the scenes." For instance, in Gurobipy, you can add an SOS constraint with the syntax* `Model.addSOS ( type, vars, weights=None )` *where* `type` *specifies the type of SOS constraint (*`GRB.SOS_TYPE2` *for type 2 SOS). In case you are wondering, a type 1 SOS constraint specifies that exactly one variable from a given list can be nonzero. For more details, you can read here.*

1.5 Set covering, set packing, and set partitioning problems

Consider a set of ground objects $M = \{1, \ldots, m\}$ and let $M_1, M_2, \ldots, M_n$ be a given collection of subsets of M. We are also given a weight c_j for each set M_j in the collection. (Depending on the application, we may want more or less weight!)

Set covering. In the **set covering** problem, we seek a collection of sets M_j so that their union includes (i.e., **covers**) M and has minimum weight. To capture this mathematically, define an **incidence matrix** A with one row for each ground object $i = 1, \ldots, m$ and one column for each set $M_j, j = 1, \ldots, n$ and such that $A_{i,j} = 1$ if $i \in M_j$ and $A_{i,j} = 0$ otherwise. The set cover problem is then:

$$\begin{aligned} \text{minimize}_x \; & w^\intercal x \\ & Ax \geq e \\ & x \in \{0, 1\}^n. \end{aligned}$$

The facility location example is a type of covering problem (customers must be covered from the open locations) and the ambulance placement problem you saw in the first homework is another example. There are many others in practice: crew scheduling in public transportation (where the elements of M are specific shifts or bus routes to cover and the sets M_j denote the ability/availability of each driver j), sensor placement (elements are locations that require sensing and each set M_j corresponds to a sensor placement choice that covers some locations), etc.

Set packing. In the **set packing** problem, we try to include as many disjoint sets M_j as possible in order to maximize the weight of the included elements. Mathematically, with the same incidence matrix A as above, the set packing problem is then:

$$\begin{aligned} \text{maximize}_x \; & w^\intercal x \\ & Ax \leq e \\ & x \in \{0, 1\}^n. \end{aligned}$$

Again, there are many practical examples. Consider flight crew scheduling, where an airline needs to assign flight crews to flights, but a crew cannot be assigned to overlapping flight schedules (e.g., two flights departing at the same time). The elements $i \in M$ are the flights and each set M_j represents the flights that a crew can cover based on availability. For another example, consider a logistics company that needs to allocate containers to ships, but each ship has limited capacity. Different shipments may overlap in terms of size and weight, and only disjoint combinations of shipments can be placed on a single ship. The elements $i \in M$ would be the shipments that need to be loaded, and each set M_j represents a collection of shipments that fit within the capacity of a ship without overlapping in size and weight. (The airline revenue management problem was also a type of packing problem!)

Set partitioning. In a **set partitioning** problem, we seek sets M_j that form a partition of M, i.e., they are disjoint and they cover M. Both maximization and minimization versions are possible here:

$$
\begin{aligned}
\text{maximize}_x \ & w^\intercal x \\
& Ax = e \\
& x \in \{0,1\}^n.
\end{aligned}
$$

1.6 Matching Problems

Matching problems are among the most ubiquitous in practice: riders being matched with drivers (in ride-sharing platforms), patients awaiting for a transplant being matched with an organ available for transplantation, etc.

To formulate a matching problem, consider a set U of tasks that must be completed and a set V of persons available to complete the tasks. Each task can be assigned to at most one person and each person is only able to complete only some of the tasks (e.g., due to skills). If task $i \in U$ is assigned to person $j \in V$, there is a reward of w_{ij}. A matching is an assignment of tasks to persons so that each task is done by at most one person and each person works on at most one task, and the goal is to find a matching that maximizes the total reward. We represent the matching abstractly through an undirected, bipartite graph $G = (\mathcal{N}, \mathcal{E})$ where the set of nodes $\mathcal{N}$ is partitioned into the two sets U, V ($\mathcal{N} = U \cup V, U \cap V = \emptyset$), nodes $i \in U$ denote tasks, nodes $j \in V$ denote persons, and an edge $\{i, j\} \in \mathcal{E}$ with $i \in U$ and $j \in V$ indicates that j is able to complete task i. (See Figure 3 for a visualization.)

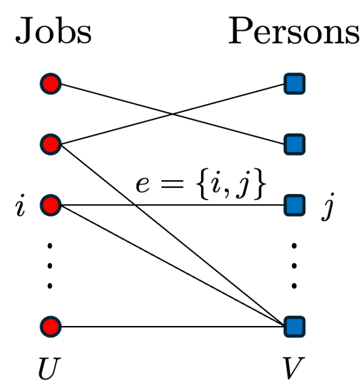


Figure 3: A maximum weight matching problem.

With decisions $x_e \in \{0,1\}$ denoting whether edge $e = \{i, j\}$ is selected – meaning task i is assigned to person j – the maximum weight matching problem is:

$$
\begin{aligned}
\text{maximize} \ & \sum_{e \in E} w_e x_e \\
\text{subject to} \ & \sum_{e \in \delta(i)} x_e \le 1, \quad \forall\, i \in N, \\
& x_e \in \{0,1\},
\end{aligned}
$$

where we use the notation $\delta(i) := \{j : \{i, j\} \in \mathcal{E}\}$ to capture all nodes j adjacent to node i.

Other variations of this problem are possible. For instance, you may encounter matching problems that involve minimizing a cost and subject to a constraint that one side has to be matched fully (e.g., all the jobs must be completed), in which case the constraints would become $\sum_{e\in\delta(i)} x_e \geq 1, \forall\, i \in U$. It is also common to consider a **perfect matching**, which is one where there is no unmatched node in the graph. (This is only possible in bipartite graphs with $|U| = |V|$, and then the constraints would read $\sum_{e\in\delta(i)} x_e = 1$ for any i.) Lastly, matching problems can be formulated in more general graphs rather than the bipartite examples we considered.

1.7 The minimum spanning tree problem

Let $G = (\mathcal{N}, \mathcal{E})$ be an undirected graph with node set $\mathcal{N}$ ($|\mathcal{N}| = n$) and edge set $\mathcal{E}$ ($|\mathcal{E}| = m$). Every edge $e \in \mathcal{E}$ has an associated cost c_e. We consider the problem of finding the **minimum spanning tree** (MST), i.e., a subset of the edges that connect all the nodes in $\mathcal{N}$ at minimum cost. To formulate the problem, we define a variable x_e for each $e \in \mathcal{E}$ that is equal to 1 if edge e is included in the tree and zero otherwise.

For this problem we actually consider two distinct formulations. The first is based on the idea that a spanning tree on n nodes should be a **connected graph containing** $n-1$ **edges**. To have $n-1$ edges, the following constraint must be satisfied:

$$\sum_{e\in\mathcal{E}} x_e = n - 1.$$

For the tree to be connected, any subset of nodes $S \subset \mathcal{N} (S \neq \emptyset)$ should be connected with nodes in $\mathcal{N} \setminus S$ through at least one edge. So if we define the **cutset** $\delta(S)$:

$$\delta(S) := \big\{\{i,j\} : i \in S, j \notin S\big\}, \tag{1}$$

we can provide the following **cutset formulation** for the MST problem:

$$\textbf{(Cutset MST)} \qquad \begin{aligned} \text{minimize } & \sum_{e\in\mathcal{E}} c_e x_e \\ & \sum_{e\in\mathcal{E}} x_e = n-1, \\ & \sum_{e\in\delta(S)} x_e \geq 1, \quad S \subset N, S \neq \emptyset \\ & x_e \in \{0,1\}. \end{aligned} \tag{2}$$

Note that the cutset formulation involves an exponential number of constraints, one for each subset $S \subset \mathcal{N}, S \neq \emptyset$.

An alternative – and equivalent – definition of a tree is based on the idea that a tree on n nodes should have exactly $n-1$ edges and no cycles. It can be shown that the tree is guaranteed to not contain a cycle if for any nonempty set $S \subset \mathcal{N}$, the number of edges with both endpoints in S is less than or equal to $|S| - 1$. For any $S \subset \mathcal{N}$, we define

$$\mathcal{E}(S) = \big\{\{i,j\} \in \mathcal{E} : i,j \in S\big\}, \tag{3}$$

and we can express these constraints as:

$$\sum_{e \in \mathcal{E}(S)} x_e \leq |S| - 1, \quad S \subset N, S \neq \emptyset, N.$$

This leads to the following IP formulation of the MST problem:

$$\textbf{(Subtour-elimination MST)} \qquad \begin{aligned} \text{minimize } & \sum_{e \in \mathcal{E}} c_e x_e \\ & \sum_{e \in \mathcal{E}} x_e = n - 1, \\ & \sum_{e \in \mathcal{E}(S)} x_e \leq |S| - 1, \quad S \subset N, S \neq \emptyset, N, \\ & x_e \in \{0, 1\}. \end{aligned} \tag{4}$$

This is called the **subtour elimination formulation** because it contains constraints that eliminate all subtours (cycles over subsets of vertices). Note that this also involves an exponential number of constraints.

The two formulations – cutset and subtour elimination – can be visualized in Figure 4.

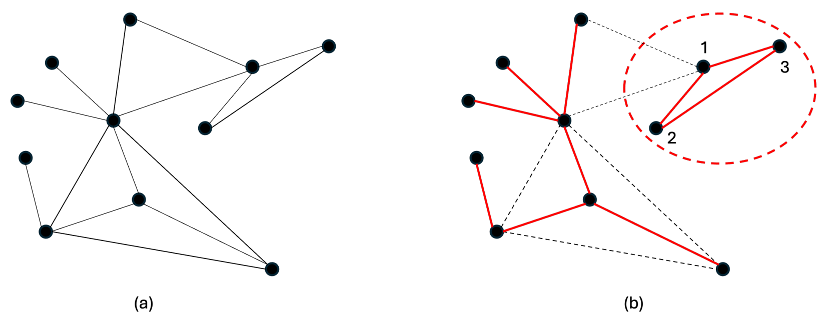


Figure 4: Formulation for the Minimum Spanning Tree Problem. The initial graph $G = (\mathcal{N}, \mathcal{E})$ is depicted in (a). Panel (b) shows a choice of edges that satisfies $\sum_{e \in \mathcal{E}} x_e = n - 1$ but is not a valid tree. Note that the **cutset** formulation would rule this out because the subset of nodes $S = \{1, 2, 3\}$ is not connected with nodes $\mathcal{N} \setminus S$, i.e., $\delta(S) = 0$. The **subtour elimination** formulation would also rule this out because $\sum_{e \in \mathcal{E}(S)} x_e = |S| > |S| - 1$.

1.8 Traveling salesperson problem

Given an undirected graph $G = (\mathcal{N}, \mathcal{E})$ and cost c_e for each edge, the objective is to find a **tour** (i.e. a cycle that visits each node exactly once) with minimum cost. To model this problem, we again use a variable x_e to denote whether an edge belongs to the tour. Mirroring the MST problem, the TSP also admits two formulations – a **cutset formulation** and a **subtour elimination** formulation – as follows.

$$\text{(Cutset TSP)} \quad \begin{aligned} &\text{minimize } \sum_{e\in} c_e x_e \\ &\sum_{e\in\delta(\{i\})} x_e = 2, \forall i \in N \\ &\sum_{e\in\delta(S)} x_e \geq 2, \forall S \subset N, S \neq \emptyset. \end{aligned} \tag{5}$$

Note that this is slightly different than the cutset MST formulation. In the cutset TSP formulation, any node i should have **exactly one edge coming into it and one edge leaving it** and any nontrivial subset of nodes S ($S \neq \emptyset, N$) should have **at least two edges** joining S with $Nset \setminus S$. This is because in TSP, we are interested in a tour, whereas in the MST we wanted a tree (which should be free of tours!)

The following formulation is also valid for the TSP (we omit the objective):

$$\text{(Subtour-elimination TSP)} \quad \begin{aligned} &\sum_{e\in\delta(\{i\})} x_e = 2, \forall i \in N \\ &\sum_{e\in\mathcal{E}(S)} x_e \leq |S| - 1, \forall S \subset N, S \neq \emptyset \end{aligned} \tag{6}$$

The key difference with the MST formulation lies again in the first set of constraints.

The two formulations are depicted in Figure 5.

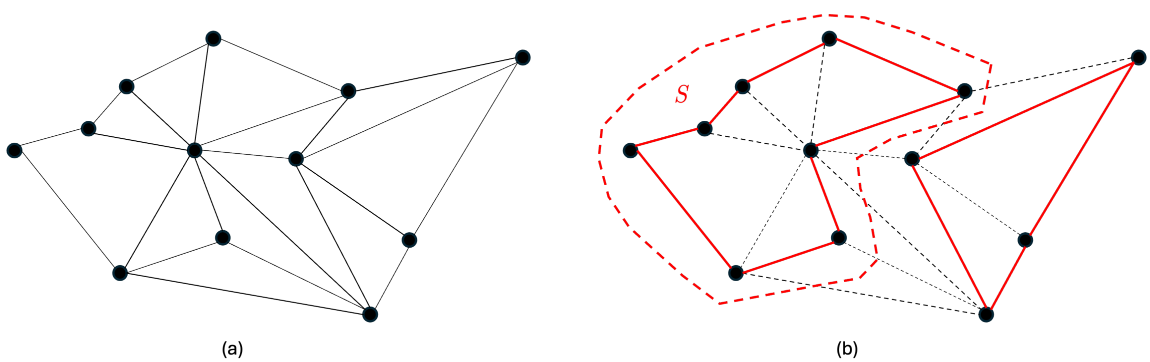


Figure 5: Formulation for Traveling Salesman Problem (TSP). The initial graph $G = (\mathcal{N}, \mathcal{E})$ is depicted in (a). Panel (b) shows a choice of edges that satisfies $\sum_{e\in\delta(i)} x_e = 2$ for any $i \in \mathcal{N}$, but is not a valid tour. Note that the **cutset** formulation rules this out because the subset of nodes $S = \{1, 2, 3\}$ is not connected with nodes $\mathcal{N} \setminus S$, i.e., $\delta(S) = 0$. The **subtour elimination** formulation would also rule this out because $\sum_{e\in\mathcal{E}(S)} x_e = |S| > |S| - 1$.

2 The Bad News First

Unfortunately, linear optimization **over integers** is **significantly harder** than over continuous variables. The following examples illustrate some of the challenges.

Example 2 (Solution Not Attained)**.** *Consider the optimization problem:*

$$\begin{aligned}\sup_{x,y}\ & x + \sqrt{2}y \\ & x + \sqrt{2}y \le \frac{1}{2} \\ & x, y \in \mathbb{Z}.\end{aligned}$$

The optimal value is not attained.

You can probably quickly see that the optimal value in this problem is $\frac{1}{2}$ and it would be achieved with any choice of x and y such that $x + \sqrt{2}y = \frac{1}{2}$. But unfortunately, no integer values of x, y would ever satisfy this with equality. Note that this problem would never arise with continuous x, y, where the optimal value would be trivially achieved.

Example 3 (No Strong Duality)**.** *Consider the following pair of optimization programs:*

$$(\mathscr{P})\ \begin{aligned}\min\ & x \\ & 2x = 1 \\ & x \ge 0\end{aligned} \qquad\qquad (\mathscr{D})\ \begin{aligned}\max_p\ & p \\ & 2p \le 1\end{aligned}$$

With $x \in \mathbb{R}$ and $p \in \mathbb{R}$, the problems constitute a primal-dual pair; both are feasible and the optimal value (for each) is $\frac{1}{2}$. With $x \in \mathbb{Z}$ and $p \in \mathbb{Z}$, problem $(\mathscr{P})$ does not have any feasible solution, but problem $(\mathscr{D})$ is feasible and has optimal value 0.

This example shows that strong duality fails with discrete variables: we have an optimization problem that has a finite optimal value (the dual $(\mathscr{D})$) but its dual is infeasible. (It is easy to construct examples where the mirroring situation also happens, i.e., the primal minimization has a finite optimal value but the dual maximization problem is infeasible).

In fact, IPs are – in theory and practice – significantly more difficult than LPs.

Theorem 1. *Given a matrix $A \in \mathbb{Q}^{m \times n}$ and a vector $b \in \mathbb{Q}^m$, the problem: "does $Ax \le b$ have an integral solution x" is **NP-complete**.*

The theorem states that the "feasibility problem" in integer programming is already NP-complete, which means it is the hardest type of problem in NP. (We will not be discussing complexity results too much in this class, but as a quick reminder, problems in NP are problems that admit a polynomial-time verification of a YES instance. For instance, in our IP feasibility problem, if we are given an x that is actually feasible, it is easy to verify whether it works – we just need to check the constraints!) For a proof of the result, see Theorem 18.1 in Schrijver (1997).

3 Linear Relaxation and Strength of IP Formulations

Despite these negative results, a substantial body of theory and very scalable algorithms have been developed to solve IPs. In the subsequent discussion, we focus on optimization problems with rational entries: $A \in \mathbb{Q}^{m\times n}, b \in \mathbb{Q}^m, c \in \mathbb{Q}^n$, and we assume that the **feasible set of the IPs of interest is bounded**. Rational entries are needed both for theoretical purposes (e.g., to ensure that optimal solutions exist) but also for the purely practical necessity of representing optimization problems on computers with finite memory. Considering bounded feasible sets only simplifies a few statements, but is not really needed for any of the theory. (In practice, this is not a terrible assumption anyway because we rarely deal with optimization problems that are truly unbounded!)

Let us start in the same way we started with linear optimization, and consider the problem of finding a good **lower bound** for an IP. Clearly, we could obtain a bound if we relaxed the integrality requirements. The following definition allows us to formalize this.

Definition 1 (LP relaxation)**.** *Given the generic integer program:*[2]

$$\begin{aligned} \min\ & c^\intercal x + d^\intercal y \\ & Ax + By = b \\ & x, y \geq 0 \\ & x \in \{0,1\}^{n_1}, y \in \mathbb{Z}^{n_2}, \end{aligned}$$

its linear programming relaxation is obtained by replacing the requirement $x \in \{0,1\}^{n_1}$ with $x \in [0,1]^{n_1}$ and replacing the requirement $y \in \mathbb{Z}^{n_2}$ with $y \in \mathbb{R}^{n_2}$.

The **LP relaxation** entails changing the binary requirement on x into a (continuous) restriction to the interval $[0,1]$ and removing the integrality requirement on y. The feasible set of the original IP is therefore contained in the feasible set of its LP relaxation (which also justifies the name!). The following observation is immediate.

Observation 1. *The optimal value of the LP relaxation to an IP provides a lower bound on the optimal value of the IP. Moreover, if the optimal solution to the LP relaxation is feasible for the original IP, then that solution is optimal for the IP.*

In practice, the LP relaxation could be quite strong but also quite weak, and critically, **this depends on the formulation of the IP!** To appreciate this point, let us consider again some of our earlier motivating examples.

3.1 Strength of IP Formulations in Our Examples

3.1.1 Facility Location.

In §1.2, we presented an IP formulation for the facility location problem. For convenience, we replicate it here (omitting the objective) and we also introduce a new formulation for

[2] A similar definition also applies to mixed-integer problems. In that case, restrictions on any continuous variables would remain unchanged.

the feasible set that we refer to as the **aggregate facility location (AFL)** formulation:

$$
\begin{array}{ll}
(FL) & (AFL) \\
\displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \dots, m & \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \dots, m \\
x_{ij} \le y_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n & \displaystyle\sum_{i=1}^{m} x_{ij} \le m y_j, \quad j = 1, \dots, n \\
x_{ij}, y_j \in \{0, 1\} & x_{ij}, y_j \in \{0, 1\}.
\end{array}
$$

The main difference is that (AFL) replaces the constraints $x_{ij} \le y_j$ in (FL) with the constraints $\sum_{i=1}^{m} x_{ij} \le m y_j$. Because the latter constraint forces x_{ij} to be 0 whenever $y_j = 0$ but allows x_{ij} to be 1 if $y_j = 1$, it is a valid reformulation. So the two formulations **result in the same feasible set of integer points** x, y and therefore also **the same optimal solutions and optimal costs**.

On first glance, the (AFL) formulation might seem superior because it has $m + n$ constraints, whereas the (FL) formulation has $m + m \cdot n$ constraints.

But consider their corresponding LP relaxations. We define the following two polyhedra, which are the feasible sets of the two relaxations:

$$
P_{\mathrm{FL}} = \left\{ (x, y) : \sum_{j=1}^{n} x_{ij} = 1, \forall i, \quad x_{ij} \le y_j, \forall i, j, \quad 0 \le x_{ij} \le 1, \quad 0 \le y_j \le 1 \right\}
$$

$$
P_{\mathrm{AFL}} = \left\{ (x, y) : \sum_{j=1}^{n} x_{ij} = 1, \forall i, \quad \sum_{i=1}^{m} x_{ij} \le m \cdot y_j, \forall j, \quad 0 \le x_{ij} \le 1, \quad 0 \le y_j \le 1 \right\}
$$

Clearly, $P_{\mathrm{FL}} \subseteq P_{\mathrm{AFL}}$ and the inclusion can actually be **strict**. In other words, the feasible set of the LP relaxation for formulation (FL) is closer to the set of integer solutions than the LP relaxation of formulation (AFL). The situation corresponds visually to Figure 6.

So if Z_{IP} is the optimal cost of the facility location IP and Z_{FL} and Z_{AFL} are the optimal costs of the two LP relaxations, we obtain that:

$$
Z_{\mathrm{AFL}} \le Z_{\mathrm{FL}} \le Z_{\mathrm{IP}},
$$

so **(FL) provides a better (i.e., higher) lower bound on optimal cost than (AFL)**.

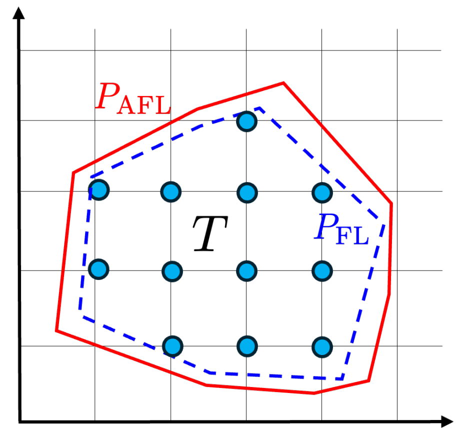


Figure 6: The feasible sets P_{FL} and P_{AFL} for the two LP relaxations for the facility location problem. Note that the feasible points T for the IP (and their convex hull) are the only integer points contained in both P_{FL} and P_{AFL}, but the (FL) formulation provides a tighter relaxation than the (AFL) formulation.

3.1.2 Minimum Spanning Tree Revisited

Recall the minimum spanning tree (MST) construction and the two formulations – cutset and subtour-elimination – which we replicate below for convenience.

(Cutset MST)

$$\sum_{e\in\mathcal{E}} x_e = n-1,$$
$$\sum_{e\in\delta(S)} x_e \geq 1, \quad S\subset\mathcal{N}, S\neq\emptyset$$
$$x_e\in\{0,1\}$$

(Subtour-elimination MST)

$$\sum_{e\in\mathcal{E}} x_e = n-1,$$
$$\sum_{e\in\mathcal{E}(S)} x_e \leq |S|-1, \quad S\subset\mathcal{N}, S\neq\emptyset,$$
$$x_e\in\{0,1\}.$$

Theorem 2. *With P_{cut} and P_{sub} denoting the feasible sets of the two LP relaxations,*
i) $P_{sub}\subseteq P_{cut}$ and examples exist where $P_{sub}\subset P_{cut}$.
ii) P_{cut} can have fractional extreme points.

Proof. a) For any set S of nodes, we have

$$\mathcal{E} = \mathcal{E}(S)\cup\delta(S)\cup\mathcal{E}(\mathcal{N}\setminus S).$$

Therefore,

$$\sum_{e\in\mathcal{E}(S)} x_e + \sum_{e\in\delta(S)} x_e + \sum_{e\in\mathcal{E}(\mathcal{N}\setminus S)} x_e = \sum_{e\in\mathcal{E}} x_e.$$

For $x\in P_{\mathrm{sub}}$, and for $S\neq\emptyset, N$, we have

$$\sum_{e\in\mathcal{E}(S)} x_e \leq |S|-1,$$

and

$$\sum_{e \in \mathcal{E}(\mathcal{N} \setminus S)} x_e \leq |N \setminus S| - 1.$$

Because

$$\sum_{e \in \mathcal{E}} x_e = n - 1,$$

we obtain that

$$\sum_{e \in \delta(S)} x_e \geq 1,$$

and therefore $x \in P_{\text{cut}}$.

b) We refer the interested reader to Bertsimas and Tsitsiklis (1997) for an example. □

3.1.3 Traveling Salesperson Problem Revisited

Lastly, recall the cutset and subtour-elimination formulations for the TSP.

(Cutset TSP)

$$\begin{aligned} \sum_{e \in \delta(\{i\})} x_e &= 2, \forall i \in N \\ \sum_{e \in \delta(S)} x_e &\geq 2, \forall S \subset N, S \neq \emptyset \end{aligned}$$

(Subtour-elimination TSP)

$$\begin{aligned} \sum_{e \in \delta(\{i\})} x_e &= 2, \forall i \in N \\ \sum_{e \in \mathcal{E}(S)} x_e &\leq |S| - 1, \forall S \subset N, S \neq \emptyset. \end{aligned}$$

Letting P_{TScut} and P_{TSsub} be the polyhedra corresponding to the LP relaxations of these two formulations, it turns out that the two formulations are equally strong, i.e., $P_{\text{TScut}} = P_{\text{TSsub}}$ (see Bertsimas and Weismantel (2005) and Bertsimas and Tsitsiklis (1997) for proofs.)

3.2 Strength of IP Formulation

These examples show that different formulations of the IP could result in different LP relaxations and therefore different lower bounds on the IP's optimal value. Because we no longer have strong duality, the quality of the lower bounds will be **critical** when solving IPs, so it is important to understand what makes some formulations better than others – and also consider what an "ideal" formulation could look like.

To understand this, let T denote all the feasible points to an IP, define

$$\text{conv}\,(T) = \left\{ \sum_{x \in T} \lambda_x \cdot x \,:\, \lambda \geq 0,\, e^\intercal \lambda = 1 \right\}$$

as their convex hull, and let P denote the feasible (polyhedral) region of an LP relaxation to our IP. Because we assumed that the feasible set for the IP is bounded, the set T is finite and conv (T) **is a polyhedral set**! Then, we clearly have (see Figure 7 for a visualization):

$$T \subseteq \text{conv}\,(T) \subseteq P.$$

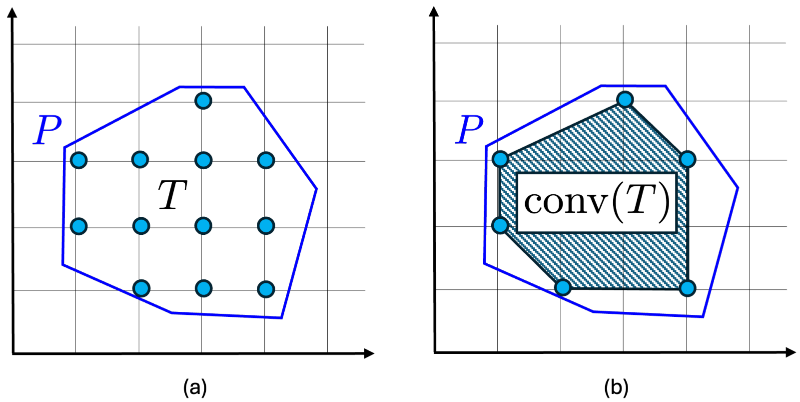


Figure 7: (a) Depicts the feasible set of the LP relaxation – the polyhedron P – and the set T of all the integer points in P. (b) Depicts the convex hull of the integer points, conv (T). The optimal value for the IP is same as the optimal value over the set conv (T).

This shows that the **ideal** LP relaxation would be one that exactly corresponds to conv (T)! Put differently, if we had access to an explicit representation of conv (T) – for instance, as an inequality description conv $(T) = \{x : Dx \leq d\}$ – then we could immediately solve our IP by **solving a linear program on the polyhedral set** conv (T).

We highlight some important take-aways, which we summarize in the following remarks.

Remark 1 (Quality of formulations)**.** ***The quality of a formulation*** *for an IP with feasible set T can be judged by how closely its LP relaxation approximates* conv (T)*. In particular, for two formulations A and B with the same feasible set of integer points and with P_A and P_B denoting the feasible sets of their LP relaxations, A is said to be* ***stronger*** *(i.e., results in an improved lower bound) than B if $P_A \subset P_B$.*

Remark 2 (Models with more constraints)**.** ***Constraints*** *play a more subtle role in an IP formulation than in an LP formulation. Whereas in an LP, formulations with more constraints should be avoided*[3]*, a valid IP formulation with more constraints is typically stronger. Adding more (valid) constraints in an IP formulation thus involves a trade-off between the strength and the and the size of the formulation.*

The results in this section will lead us in two different directions. §4 examines what types of IP formulations are "ideal," meaning they result in LP relaxations that exactly correspond to the convex hull of all integer feasible solutions. When that is not possible, §**??** shows how to add **valid cuts**, which are linear inequalities that remove fractional points from the feasible set of the LP relaxation without removing any integer points.

4 Ideal Formulations With Total Unimodularity

This section examines the first set of conditions that guarantee an ideal IP formulation, i.e., one where the LP relaxation's feasible region would have only **integral** extreme points.

[3]For LPs, introducing constraints increases the problem size and also introduces degeneracy, which can complicate algorithms like simplex.

Let $\mathcal{F} = \{x \in \mathbb{Z}_+^n \mid Ax \leq b\}$ be the set of integer points for an IP formulation, where $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$,[4] and let P denote the feasible set of its LP relaxation:

$$P = \{x \in \mathbb{R}_+^n \mid Ax \leq b\}.$$

Our goal is to identify conditions on the matrix A such that P **is integral**, i.e., $P = \text{conv}(\mathcal{F})$. We start by recalling Cramer's rule.

Proposition 1 (Cramer's Rule)**.** *Let $A \in \mathbb{R}^{n \times n}$ be a nonsingular matrix. For $b \in \mathbb{R}^n$,*

$$Ax = b \quad \Longrightarrow \quad x = A^{-1}b \quad \Longrightarrow \quad x_i = \frac{\det(A^i)}{\det(A)}, \ \forall i,$$

where A^i is the matrix with columns $A^i_j = A_j$ for all $j \in \{1, \ldots, n\} \setminus \{i\}$ and $A^i_i = b$.

To motivate the definition of total unimodularity, consider the polyhedron

$$P = \{x \in \mathbb{R}_+^n \mid Ax = b\}$$

with $A \in \mathbb{Z}^{m \times n}$ of full row rank and $b \in \mathbb{Z}^m$. For each vertex x of P, there exists a basis $B \subset \{1, \ldots, n\}$ such that $x_B = A_B^{-1}b$ and $x_N = 0$. For matrices with $\det(A_B) = \pm 1$, Cramer's rule ensures that A_B^{-1} is integral. Therefore, integrality of x can be guaranteed if we require that $\det(A_B)$ is equal to ± 1. This motivates the following definition.

Definition 2 (Unimodularity, Total unimodularity)**.**

1. *A matrix $A \in \mathbb{Z}^{m \times n}$ of full row rank is* ***unimodular*** *if the determinant of A_B is 1 or -1 for every basis B.*

2. *A matrix $A \in \mathbb{Z}^{m \times n}$ is* ***totally unimodular*** *if the determinant of each square submatrix of A is 0, 1, or -1.*

Note that all entries of a totally unimodular matrix (which are 1×1 submatrices of A) must belong to the set $\{0, 1, -1\}$. However, that is not the case for unimodular matrices; for instance, the matrix

$$A = \begin{bmatrix} 3 & 2 \\ 1 & 1 \end{bmatrix}.$$

is unimodular. The reason we carry both definitions is to be able to make statements about optimization problems in standard form and in inequality form. We will provide several characterizations that allow checking quickly whether a matrix is (totally) unimodular. For now, to appreciate why the definitions are important, we state the main result of interest.

[4] The restriction to integer matrices is without loss of generality here: if the entries were rational, we could multiply all the equations by the least common multiple of (the absolute values of) all denominators.

Theorem 3.

1. *The matrix $A \in \mathbb{Z}^{m\times n}$ of full row rank* ***is unimodular if and only if*** *the polyhedron $P(b) = \{x \in \mathbb{R}^n_+ \mid Ax = b\}$ is integral for all $b \in \mathbb{Z}^m$ with $P(b) \neq \emptyset$.*

2. *The matrix A* ***is totally unimodular if and only if*** *the polyhedron $P(b) = \{x \in \mathbb{R}^n_+ \mid Ax \leq b\}$ is integral for all $b \in \mathbb{Z}^m$ with $P(b) \neq \emptyset$.*

Proof. (a) "$\Rightarrow$" Assume that A is unimodular. Consider $b \in \mathbb{Z}^m$ with $P(b) \neq \emptyset$. Any extreme point $x \in P(b)$ can be written as (x_B, x_N), where $x_B = A_B^{-1}b$ and $x_N = 0$ for some basis B. Because A is unimodular, $\det(A_B) = \pm 1$, which by Cramer's rule implies that x_B (and therefore also x) is integral.

"$\Leftarrow$" Suppose that $P(b) \neq \emptyset$ is integral for any integral b. Let B be any basis of A. We claim that it's sufficient to argue that A_B^{-1} is integral; because A_B is integral and $\det(A_B) \cdot \det(A_B^{-1}) = 1$, that would imply that $\det(A_B) \in \{1, -1\}$ and thus that A is unimodular. To prove that A_B^{-1} is integral, consider a right-hand-side $b = A_B \cdot z + e_i$, where z is an integral vector. We have that $A_B^{-1} \cdot b = z + A_B^{-1}e_i$. Thus, by choosing z sufficiently large so that $z + A_B^{-1}e_i \geq 0$ (which can readily be done by increasing the entries), we obtain a basic feasible solution for $P(b)$. Because this is integral by assumption, this implies that $A_B^{-1}e_i$ must be integral. Repeating the argument for all e_i proves that A_B^{-1} is integral.

(b) We claim that A is totally unimodular if and only if the matrix $[A, I]$ is unimodular. Moreover, we claim that for any $b \in \mathbb{Z}^m$, the extreme points of the polyhedron $\{x \in \mathbb{R}^n_+ \mid Ax \leq b\}$ are integral if and only if the extreme points of the polyhedron $\{(x, y) \in \mathbb{R}^{n+m}_+ \mid Ax + Iy = b\}$ are integral. (These will be the subject of future propositions, but the proofs follow by suitably expanding determinants.) The result then follows from part (a). $\square$

The critical consequence from Theorem 3 is that the optimal value in the IP $\min\{c^\intercal x \mid Ax \leq b, x \in \mathbb{Z}^n_+\}$ is obtained by solving the LP $\min\{c^\intercal x \mid Ax \leq b, x \in \mathbb{R}^n_+\}$.

Detecting (total) unimodularity is therefore quite critical, so we provide a few additional characterizations followed by examples.

Proposition 2. *Consider a matrix $A \in \{0, 1, -1\}^{m \times n}$. The following are equivalent:*

1. *A is totally unimodular.*
2. *$A^\intercal$ is totally unimodular.*
3. *$[A^\intercal \;\; -A^\intercal \;\; I \;\; -I]$ is totally unimodular.*
4. *$\{x \in \mathbb{R}^n_+ \mid Ax = b, 0 \le x \le u\}$ is integral for all integral b, u.*
5. *$\{x \mid a \le Ax \le b, \ell \le x \le u\}$ is integral for all integral a, b, ℓ, u.*
6. *Each collection of columns of A can be partitioned into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries $0, +1$, and -1. (By part 2, a similar result also holds for the rows of A.)*
7. *Each nonsingular submatrix of A has a row with an odd number of non-zero components.*
8. *The sum of entries in any square submatrix with even row and column sums is divisible by four.*
9. *No square submatrix of A has determinant +2 or -2.*

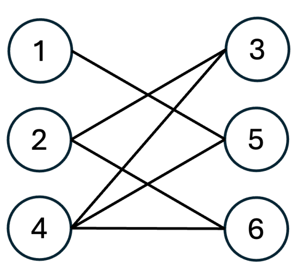
For a proof of these results, see Theorem 19.3 in Schrijver (1997).

4.1 Examples of Totally Unimodular Matrices

4.1.1 Node-Edge Incidence Matrix for Bipartite Graphs

Let $G = (\mathcal{N}, \mathcal{E})$ be an **undirected graph** and let $A \in \{0, 1\}^{|\mathcal{N}| \times |\mathcal{E}|}$ be the node-edge incidence matrix of G, i.e., $A_{i,e} = 1$ if and only if $i \in e$. Then, A **is TU if and only if the graph G is bipartite**. The proof follows from Proposition 2 (#6) and is omitted.

Figure 8 shows an example. Recalling our discussion of matching problems in §1.6, it can be seen that if these are defined on bipartite graphs, all matching formulations will admit integral LP relaxations.

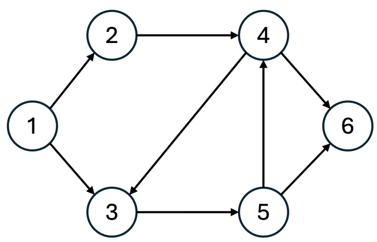


	$\{1,5\}$	$\{2,3\}$	$\{2,6\}$	$\{4,3\}$	$\{4,5\}$	$\{4,6\}$
1	1	0	0	0	0	0
2	0	1	1	0	0	0
3	0	1	0	1	0	0
4	0	0	0	1	1	1
5	1	0	0	0	1	0
6	0	0	1	0	0	1

Figure 8: Undirected bipartite graph and its node-edge incidence matrix.

4.2 Node-Arc Incidence Matrix for Directed Graphs

Let $D = (N, E)$ be a directed graph and let M be the $N \times E$ incidence matrix of D, where $M_{v,e} = 1$ if and only if $e = (\cdot, v)$ (arc e enters node v), $M_{v,e} = -1$ if and only if $e = (v, \cdot)$ (arc e leaves node v), and $M_{v,e} = 0$ otherwise. Then, M is TU. Figure 9 shows an example.



	$(1,2)$	$(1,3)$	$(2,4)$	$(4,3)$	$(3,5)$	$(5,4)$	$(4,6)$	$(5,6)$
1	-1	-1	0	0	0	0	0	0
2	1	0	-1	0	0	0	0	0
3	0	1	0	1	-1	0	0	0
4	0	0	1	-1	0	1	-1	0
5	0	0	0	0	1	-1	0	-1
6	0	0	0	0	0	0	1	1

Figure 9: Directed graph and its node-arc incidence matrix.

The important consequence of this result is that all network flow problems with integral arc capacities and integral demand or supply at nodes will admit an integral LP relaxation. The Prosche Motors problem on the second homework is one such example – so with integral data, the optimal solution is guaranteed to be integral.

4.3 Interval Matrices

If $A \in \{0,1\}^{m \times n}$ and each column of A has its values of 1 consecutively (under some ordering of the columns of A), then A is TU. Such matrices are called **interval matrices**. An example is the matrix below.

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

4.4 Network Matrices

All of the examples above are special instances of **network matrices**. To formalize this broad class, let $D = (V, A)$ be a directed graph and let $T = (V, A_0)$ be a directed tree on V. Let M be the $A_0 \times A$ matrix defined by, for $a = (v, w) \in A$ and $a' \in A_0$:

$$M_{a',a} = \begin{cases} +1 & \text{if the unique } v - w \text{ path in } T \text{ passes through } a' \text{ forwardly} \\ -1 & \text{if the unique } v - w \text{ path in } T \text{ passes through } a' \text{ backwardly} \\ 0 & \text{if the unique } v - w \text{ path in } T \text{ does not pass through } a'. \end{cases}$$

Then, M is TU. For an example, consider the directed graph D and directed tree T in Figure 10. The corresponding network matrix is shown in Table 1.

There is a very famous result in combinatorial optimization due to Seymour (1980), who showed that every TU matrix arises, in a certain way, from network matrices and just **two** other matrices. Importantly, testing whether a matrix is TU can be done in polynomial time; for more details, see Schrijver (1997).

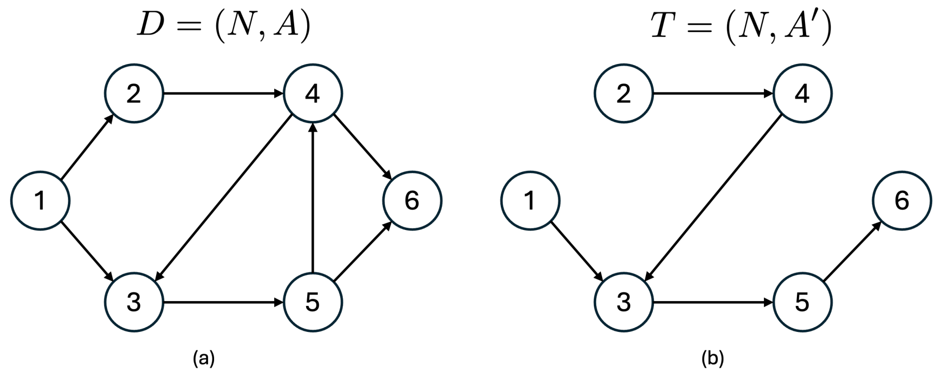


Figure 10: Directed graph (a) and a directed tree (b).

	$(1,2)$	$(1,3)$	$(2,4)$	$(4,3)$	$(3,5)$	$(5,4)$	$(4,6)$	$(5,6)$
$(1,3)$	1	1	1	0	0	0	0	0
$(2,4)$	-1	0	0	0	0	0	0	0
$(4,3)$	-1	0	0	1	0	-1	1	0
$(3,5)$	0	0	0	0	1	-1	1	0
$(5,6)$	0	0	0	0	0	0	1	1

Table 1: Network matrix corresponding to the directed graph and tree in Figure 10.

5 Dual Integrality and Submodular Functions

Next, we discuss an alternative way to show integrality of polyhedra based on linear optimization duality. This will also allow us to discuss submodular and supermodular functions, which are extremely important concepts in their own right in optimization.

The approach here is based on a simple observation: to show that the LP relaxation of an IP has integral extreme points, it suffices to check that the optimal value of any LP with integer cost vector is an integer. The following proposition summarizes the idea.

Proposition 3. *Let P be a nonempty polyhedron with at least one extreme point. The polyhedron P is integral if and only if the optimal value $Z_{LP} := \min\{c^\intercal x \mid x \in P\}$ is an integer, for all $c \in \mathbb{Z}^n$.*

The intuition should be quite clear; the proof is straightforward and is omitted.

Therefore, to show integrality of a polyhedron P, it suffices to show that $Z_{LP} \in \mathbb{Z}$ for all $c \in \mathbb{Z}^n$. One way to show that is to **construct a dual optimal integral solution** for any such c.[5] We illustrate this with an example that is important in its own right.

[5] These ideas are related to the concept of **total dual integrality** (TDI), which has been studied extensively in combinatorial optimiziation. For a more general treatment, the interested reader can refer to Schrijver (1997).

5.1 Polymatroid Polyhedra and Submodular Functions

For a given finite set $N = \{1, \dots, n\}$, consider a function $f(S)$ defined on subsets S of N.

Definition 3 (Sub-, super-modular)**.** *A set function $f : 2^N \to \mathbb{R}$ is* ***submodular*** *if*

$$f(S) + f(T) \geq f(S \cap T) + f(S \cup T), \quad \forall S, T \subset N$$

and it is ***supermodular*** *if the reverse inequality holds.*

Note that the condition in the definition may not make a lot of sense written this way, but it is equivalent to:

$$f(S) - f(S \cap T) \geq f(S \cup T) - f(T), \quad \forall S, T \subset N.$$

In this form, note that the set difference between the sets appearing on the left of the inequality is exactly $S \setminus (S \cap T) = S \setminus T$, which exactly matches the difference between the sets on the right because $(S \cup T) \setminus T = S \setminus T$. So the condition is stating that the gains obtained when adding $S \setminus T$ to the set $S \cap T$ are greater than the gains obtained when adding the same set to the larger set T. The following alternative definitions will make this intuition even more clear.

Proposition 4. *A set function $f : 2^N \to \mathbb{R}$ is* ***submodular if and only if****:*

(a) For any $S, T \subseteq N$ such that $S \subseteq T$ and $k \notin T$:

$$f(S \cup \{k\}) - f(S) \geq f(T \cup \{k\}) - f(T).$$

(b) For any $S \subseteq N$ and any j, k with $j, k \notin S$ and $j \neq k$:

$$f(S \cup \{j\}) - f(S) \geq f(S \cup \{j, k\}) - f(S \cup \{k\}). \tag{3.2}$$

For a proof, see Bertsimas and Weismantel (2005) or Bach (2010).

These equivalent definitions should make it clear that a submodular function has the certain **"diminishing returns"** or **"decreasing differences"** property: the marginal gain when adding an element k to a larger set T is smaller than the gain when adding k to a smaller set S (or equivalently, the marginal gain from including an extra element j is smaller when some other element k is also included). In economics, a submodular **cost** function captures economies of scale, whereas a submodular **profit function** captures substitution. (Supermodular functions are the exact opposite.) On first glance, one may perceive submodular functions as a discrete analog to concave functions, but that analogy only holds solely in terms of economic intuition, but **not** from an optimization standpoint! In fact, **in terms of optimization, submodular functions behave more like convex functions**, e.g., there are efficient algorithms to minimize them, they admit a very elegant link to convexity (through the Lovasz extension) and they also admit a duality theory.

Submodular and supermodular functions play central roles in a variety of fields, including operations research, economics, and computer science. The scope of our treatment here will be limited, but we direct the interested reader to Bach (2013) for a concise overview and the book Schrijver (2003) for an in-depth treatment.

Subsequently, we are interested in submodular functions that are non-negative and **increasing**[6] in the set inclusion sense, i.e.,

$$f(S) \le f(T), \quad \forall S \subset T \subseteq N.$$

5.1.1 Examples

A few quick examples of (monotone) submodular functions.

- **Linear functions.** A function $f : 2^N \to \mathbb{R}$ is **modular** if

  $$f(A) = \sum_{i \in A} w_i$$

  for some weights $w : N \to \mathbb{R}$. Such functions are both supermodular and submodular. If $w_i \ge 0$ for all $i \in N$, then f is also increasing.

- **Compositions with linear functions.** As a generalization of the linear case, consider any weights $w \ge 0$ and any **concave** function $g : \mathbb{R} \to \mathbb{R}$. Then, the function $f : 2^N \to \mathbb{R}$

  $$f(S) = g\Big(\sum_{i \in S} w_i\Big)$$

  is submodular. If g is increasing, f is also increasing.

  For a different example, suppose that g is **convex** and the weights w are zero except for two weights with opposite signs, i.e., $\exists\, i \ne j : w_i \le 0,\, w_j \ge 0$. Then, $f : 2^N \to \mathbb{R}$ defined as

  $$f(S) = g\Big(\sum_{i \in S} w_i\Big)$$

  is submodular.

- **Set systems and coverage functions.** Given a universe U and n subsets $A_1, A_2, \ldots, A_n \subset U$, we can define several natural submodular functions on the set $N = \{1, 2, \ldots, n\}$. First, the coverage function given by

  $$f(S) = \left| \bigcup_{i \in S} A_i \right|$$

  is submodular. This naturally extends to the weighted coverage function: given a non-negative weight function $w : U \to \mathbb{R}_+$,

  $$f(S) = w\left(\bigcup_{i \in S} A_i \right)$$

[6] We use "increasing" and "decreasing" in weak sense, and use "strictly" to emphasize strict relationships.

is submodular. Another related function defined by

$$f(S) = \sum_{x \in U} \max_{i \in S} w(A_i, x)$$

is also submodular, where $w(A_i, x)$ is a non-negative weight for A_i covering x. All these functions are increasing.

- **Valuation functions with decreasing marginal values**. Sometimes we assume that a certain function is submodular not because it arises in a specific combinatorial way, but because it arises in a setting where it's natural to have decreasing marginal returns. An example are combinatorial auctions, where each player has a valuation function $w : 2^N \to \mathbb{R}$ on subsets of items. This might have a specific form, like

$$w(S) = \min\left(\sum_{j \in S} v_j, B\right),$$

or it might be given by a black box. However, we might assume that the (unknown) function is submodular just it may be natural to expect that having more items decreases the benefit of acquiring another item.

- **Optimal TSP cost on tree graphs.** Consider an **undirected tree graph** $G = (N, E)$ with a positive cost for traversing the edges ($c_e \geq 0$ for every edge $e \in E$). For every $S \subseteq N$, define $f(S)$ as the optimal (i.e., smallest) cost for a TSP that goes through all the nodes in S. Then, $f(S)$ is submodular.

- **Network optimization.** Submodular functions also arise in network optimization models. For instance, consider a directed graph where there are capacities on the edges that constrain how much flow can be transported through the edge. Then, if we define $f(S)$ as the maximum flow that can be received at a set of sink nodes S, the function $f(S)$ is submodular.

- **Inventory and supply chain management.** Lastly, submodular functions appear frequently in the study of supply chain and inventory management, such as when characterizing perishable inventory systems, dual sourcing, and inventory control problems with trans-shipment.

Returning to our setting, let us consider the following problem:

$$\begin{aligned} \text{maximize } & \sum_{j=1}^{n} r_j x_j \\ & \sum_{j \in S} x_j \leq f(S), \ \forall S \subseteq N \\ & x \in \mathbb{Z}_+^n. \end{aligned}$$

This problem essentially looks like an extension of the knapsack problem that we considered earlier, except that there is one constraint for every possible subset $S \subseteq N$. Let $\mathcal{F}$ denote the set of feasible integer solutions and let

$$P(f) = \left\{ x \in \mathbb{R}^n_+ \;\middle|\; \sum_{j \in S} x_j \leq f(S),\ \forall S \subset N \right\}$$

denote the feasible set of the LP relaxation.

We next state and prove the main result in this section: the polyhedron $P(f)$, which is called a **polymatroid**, is integral for any $f(S)$ is submodular and increasing.

Theorem 4. *If f is submodular, increasing, integer valued, and $f(\emptyset) = 0$, then*

$$P(f) = \operatorname{conv}(\mathcal{F}).$$

Proof. Consider the linear relaxation and its dual:

$$\begin{aligned} &\text{maximize } \sum_{j=1}^{n} r_j x_j & \qquad &\text{minimize } \sum_{S \subset N} f(S) y_S \\ &\sum_{j \in S} x_j \leq f(S), \quad S \subset N, & &\sum_{S : j \in S} y_S \geq r_j,\ j \in N, \\ &x_j \geq 0,\ j \in N & &y_S \geq 0, \quad S \subset N. \end{aligned} \tag{7}$$

The key intuition behind the proof is that in a maximization like the one in the primal above, the use of a submodular function to evaluate the right-hand-sides implies that a **greedy** heuristic actually produces an optimal solution. So we will construct such a greedy solution for the primal and also a feasible solution for the dual with the same cost.

Suppose $r_1 \geq r_2 \geq \ldots \geq r_k > 0 \geq r_{k+1} \geq \ldots \geq r_n$. Let $S^j = \{1, \ldots, j\}$ for $j \in N$, and $S^0 = \emptyset$. We prove that the following primal and dual solutions x and y are optimal for the primal and dual problem, respectively.

$$x_j = \begin{cases} f(S^j) - f(S^{j-1}), & \text{for } 1 \leq j \leq k, \\ 0, & \text{for } j > k. \end{cases}$$

$$y_S = \begin{cases} r_j - r_{j+1}, & \text{for } S = S^j, \quad 1 \leq j < k, \\ r_k, & \text{for } S = S^k, \\ 0, & \text{otherwise.} \end{cases}$$

Because f is integer valued, $x \in \mathbb{Z}^n$. Moreover, x is primal feasible: f is increasing, which implies $x_j \geq 0$, and for all $T \subset N$, we have:

$$
\begin{aligned}
\sum_{j \in T} x_j &= \sum_{j \in T, j \leq k} \left(f(S^j) - f(S^{j-1}) \right) \\
\text{(because } f \text{ submodular)} \quad &\leq \sum_{j \in T, j \leq k} \left(f(S^j \cap T) - f(S^{j-1} \cap T) \right) = \\
&= f(S^k \cap T) - f(\emptyset) \\
\text{(because } f \text{ monotone)} \quad &\leq f(T) - f(\emptyset) \\
\text{(because } f(\emptyset) = 0\text{)} \quad &= f(T).
\end{aligned}
$$

To show that y is dual feasible, note that $y_S \geq 0$ and:

$$
\sum_{S: j \in S} y_S = y_{S^j} + \ldots + y_{S^k} = r_j, \text{ if } j \leq k \quad \text{and} \quad \sum_{S: j \in S} y_S = 0 \geq r_j, \text{ if } j > k.
$$

The primal objective value is

$$
\sum_{j=1}^{k} r_j \left(f(S^j) - f(S^{j-1}) \right),
$$

and the dual objective value is

$$
\sum_{j=1}^{k-1} (r_j - r_{j+1}) f(S^j) + r_k f(S^k) = \sum_{j=1}^{k} r_j \left(f(S^j) - f(S^{j-1}) \right).
$$

From strong duality, the two problems have the same optimal value. Because this is true for every $r \in \mathbb{Z}^n$, it follows that $P(f) = \operatorname{conv}(\mathcal{F})$. □

An analogous result holds in the context of the following minimization problem

$$
\begin{aligned}
\min \ & \sum_{j=1}^{n} c_j x_j \\
& \sum_{j \in S} x_j \geq f(S), \ \forall S \subseteq N, \\
& x \in \mathbb{Z}_+^n,
\end{aligned}
$$

where the function f is **supermodular**. The arguments are identical and are omitted.

Importantly, the proof above highlighted that a **greedy solution is optimal** for problem (7). The intuition is directly tied to the diminishing returns property of submodular functions and can appreciated when interpreting the problem as a generalized knapsack problem. Because any item j brings more reward when included in a smaller (rather than larger) set S, it is optimal to include the items in decreasing order of their rewards r_i as long as the rewards are positive.

Several important extensions of this result are possible. For instance, a similar result also holds when the function $f(S)$ is the **minimum of two** submodular, increasing, integer-valued functions. For details, see Bertsimas and Weismantel (2005) or Schrijver (2003).

References

Francis Bach. Convex analysis and optimization with submodular functions: a tutorial. Technical report, INRIA, HAL, 2010. URL https://hal.science/hal-00527714v2.

Francis Bach. *Learning with Submodular Functions: A Convex Optimization Perspective*. Now Publishers Inc., 2013. URL https://www.di.ens.fr/~fbach/2200000039-Bach-Vol6-MAL-039.pdf.

Dimitris Bertsimas and John Tsitsiklis. *Linear Optimization*. Athena Scientific, 1997.

Dimitris Bertsimas and Robert Weismantel. *Optimization Over Integers*. Dynamic Ideas, 2005.

Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1997.

Alexander Schrijver. *Combinatorial Optimization*. Algorithms and Combinatorics. Springer Berlin, Heidelberg, 2003.