Modeling with Integer Programming

Laura Galli

September 26, 2016

We can use 0-1 (binary) variables for a variety of purposes, such as:

- Modeling yes/no decisions
- Enforcing disjunctions
- Enforcing logical conditions
- Modeling fixed costs
- Modeling piecewise linear functions

1 Modeling Yes/No Decisions

We can use binary variables to model yes/no decisions.

Example: 0-1 Knapsack

- We are given a set of items $j = 1 \ldots n$ with associated values $p_j$ and weights $w_j$.
- We wish to select a subset of maximum value such that the total weight is less than a constant $b$.
- We can associate a 0-1 variable $x_j$ with each item indicating whether it is selected or not.

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} p_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} w_j x_j \leq b \\
& \quad x \in \{0,1\}^n.
\end{align*}
\]

Variants are: Subset-Sum, KP-Min, Equality-KP, Multiple-Choice-KP, Bounded-KP, Two-Constrained-KP, Multiple-KP.
Example: (Linear) Assignment

- We are given \( n \) people and \( m \) jobs.
- Each job must be done by exactly one person.
- Each person can do at most one job.
- The cost of person \( j = 1, \ldots, n \) doing job \( i = 1, \ldots, m \) is \( c_{ij} \).
- We wish to find a minimum cost assignment.
- We can associate a 0-1 variable \( x_{ij} \) to each possible assignment (job \( i \) to person \( j \)) indicating whether it is selected or not.

\[
\begin{align*}
\min & \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, m) \\
& \quad \sum_{i=1}^{m} x_{ij} \leq 1 \quad (j = 1, \ldots, n) \\
& \quad x \in \{0, 1\}^{mn}.
\end{align*}
\]

Note that we can always assume \( m = n \), w.l.o.g.:

- If \( m > n \) the problem is infeasible (pigeonhole principle).
- If \( n > m \) we can set \( m = n \) adding \( n - m \) “dummy” jobs such that the corresponding cost \( c_{ij} = 0 \) for \( i = m + 1, \ldots, n \) and \( j = 1, \ldots, n \).
- So, the model becomes:

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, n) \\
& \quad \sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, \ldots, n) \\
& \quad x \in \{0, 1\}^{n^2}.
\end{align*}
\]

- Thus, assignment problems deal with the question of how to assign \( n \) items to \( n \) other items.
- There are different ways in math to describe an assignment. For example, we can view an assignment as a bijective mapping \( \psi \) between two finite sets \( U \) and \( V \) of \( n \) elements.
- This way, we get a representation of an assignment by a permutation \( \psi \), we can just represent the permutation as \( \psi = \psi(1), \psi(2), \ldots, \psi(n) \).
• Every permutation $\psi$ corresponds in a unique way to an $n \times n$ permutation matrix $X_\psi$ with $x_{ij} = 1$ if $j = \psi(i)$, 0 otherwise.

• So there is a one-to-one correspondence between assignments and permutations of $\{1, \ldots, n\}$ elements.

Note that this is an “easy” problem, indeed constraint matrix is TU.

**Example: Generalized Assignment**

A $NP$-hard version is the so called Generalized Assignment Problem, where each person $j = 1, \ldots, n$ has a “capacity” $b_j$ (e.g., max working hours), and each job $i = 1, \ldots, m$ assigned to person $j$ uses up $w_{ij}$ of the person capacity $b_j$. Each job must be assigned exactly to one person, and each person cannot exceed his/her own capacity.

$$
\begin{align*}
\min & \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, m) \\
& \quad \sum_{i=1}^{m} w_{ij} x_{ij} \leq b_j \quad (j = 1, \ldots, n) \\
& \quad x \in \{0,1\}^{mn}.
\end{align*}
$$

2 Selecting from a Set, Fixed Costs, and Dependent Decisions

• We can use binary variables to model conditions such as at most one, exactly one, at least one when selecting elements from a set $M$:

• Let $M = \{1 \ldots m\}$ be a finite set.

• Let $\{M_j\}, j \in N = \{1 \ldots n\}$ be a collection of subsets of $M$.

• We say $F \subseteq N$ covers $M$ if $\cup_{j \in F} M_j = M$ (i.e., at least one).

• We say $F \subseteq N$ is a packing of $M$ if $M_j \cap M_k = \emptyset, \forall (j, k), j \neq k$ (i.e., at most one).

• If $F$ is both a packing and a cover, we say $F$ is a partition of $M$ (i.e., exactly one).

• These problems can be easily represented as IPs.

• We define an incidence matrix $A_{m \times n}$ of the family $\{M_j\}$:
  - Each row of $A$ represents an item of set $M$
  - Each column of $A$ represents a subset $M_j$ of the items
• We can associate a 0-1 variable to each subset indicating whether the subset is selected or not.

• This is a 01-IP model for the set covering problem:

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad Ax \geq 1 \\
& \quad x \in \{0, 1\}^n,
\end{align*}
\]

where \(c_j, \ j = 1 \ldots n\) is the cost vector of the \(n\) subsets.

Similarly, the model for set partitioning:

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad Ax = 1 \\
& \quad x \in \{0, 1\}^n,
\end{align*}
\]

and the model for set packing:

\[
\begin{align*}
\text{max} & \quad c^T x \\
\text{s.t.} & \quad Ax \leq 1 \\
& \quad x \in \{0, 1\}^n.
\end{align*}
\]

• We can use binary variables to model fixed costs: \(h(x) = f + px\) if \(0 < x \leq C\) and \(h(x) = 0\) if \(x = 0\). To do this we define an additional binary variable \(y = 1\) if \(x > 0\) and \(y = 0\) otherwise. We replace \(h(x)\) by \(fy + px\), and add the constraints \(x \leq Cy, \ y \in \{0, 1\}\).

• Note that this is not a completely satisfactory formulation, because although the costs are correct when \(x > 0\), it is possible to have the solution \(x = 0, \ y = 1\). So we did not model the set \(X = \{(0, 0), (x, 1)\text{ for } 0 < x \leq C\}\), but the set \(X \cup \{0, 1\}\). However, as a typical objective is of minimization, this will not arise in an optimal solution.

• We can use binary variables to enforce the condition that a certain action \(X\) can only be taken if some other action \(Y\) is also taken, i.e., dependent decisions. We introduce two binary variables \(x\) and \(y\) for actions \(X\) and \(Y\), respectively, and impose the constraint: \(x \leq y\).
Example: Uncapacitated Facility Location

- We are given \( n \) potential facility locations and \( m \) customers that must be serviced from those locations.
- There is a fixed cost \( d_j \) of opening facility \( j \).
- There is a cost \( c_{ij} \) associated with serving customer \( i \) from facility \( j \).
- We can use two sets of binary variables:
  - \( y_j \) is a binary variable taking value 1 if facility \( j \) is opened, 0 otherwise.
  - \( x_{ij} \) is a binary variable taking value 1 if customer \( i \) is served by facility \( j \), 0 otherwise.

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, m) \\
& \quad x_{ij} \leq y_j \quad (i = 1, \ldots, m; j = 1 \ldots n) \\
& \quad x \in \{0, 1\}^{mn}, \quad y \in \{0, 1\}^{n}.
\end{align*}
\]

Alternatively, we can model dependency constraints (1) as follows:

\[
\sum_{i=1}^{m} x_{ij} \leq m y_j \quad (j = 1 \ldots n).
\]

Example: Capacitated Facility Location

In the capacitated version we are given a demand \( r_i \) for each customer \( i \) and a capacity \( b_j \) for each facility \( j \). The following is a valid IP formulation:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, m) \\
& \quad \sum_{i=1}^{m} r_i x_{ij} \leq b_j y_j \quad (j = 1, \ldots, n) \\
& \quad x \in \{0, 1\}^{mn}, \quad y \in \{0, 1\}^{n}.
\end{align*}
\]
Example: Bin Packing

We are given \( n \) bins with the same capacity \( b \) and \( m \) items with weight \( r_i, \ i = 1 \ldots m \). We want to pack all items in the bins and minimize the number of bins used:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} y_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, m) \\
& \quad \sum_{i=1}^{m} r_i x_{ij} \leq b y_j \quad (j = 1, \ldots, n) \\
& \quad x \in \{0, 1\}^{mn}, \quad y \in \{0, 1\}^n.
\end{align*}
\]  

Example: Cutting Stock

There are \( n \) rolls of fixed size \( b \) awaiting to be cut, aka raws. There are \( s \) manufacturers who want different numbers \( d_i \) of rolls of various-sized widths \( w_i, i = 1, \ldots, s \), aka finals. How should the raws be cut so that the least number of rolls are cut? The following model by Kantorovich is a valid formulation for the cutting stock problem:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} y_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} \geq d_i \quad (i = 1, \ldots, s) \\
& \quad \sum_{i=1}^{s} w_i x_{ij} \leq b y_j \quad (j = 1, \ldots, n) \\
& \quad x \in \mathbb{Z}^{sn}, \quad y \in \{0, 1\}^n.
\end{align*}
\]

An alternative, stronger formulation, is due to Gilmore & Gomory. The idea is to enumerate all possible raw cutting patterns. A pattern \( j \in J \) is described by the vector \((a_{1j}, a_{2j}, \ldots, a_{sj})\), where \( a_{ij} \) represents the number of final rolls of width \( w_i \) obtained from cutting a raw roll according to pattern \( j \). In this model we have an integer variable \( x_j \) for each pattern \( j \in J \), indicating how many times pattern \( j \) is used, i.e., how many raw rolls are cut according to pattern \( j \).

\[
\begin{align*}
\min & \quad \sum_{j \in J} x_j \\
\text{s.t.} & \quad \sum_{j \in J} a_{ij} x_j \geq d_i \quad (i = 1, \ldots, s) \\
& \quad x \in \mathbb{Z}^{|J|}_+.
\end{align*}
\]

Clearly, the Gilmore-Gomory model must be solved using column generation techniques.

Note that Bin Packing and Cutting Stock are in fact the same problem! In bin packing we are given an item vector \((r_i), \ i = 1, \ldots, m\) and we want to minimize the number of bins of capacity \( b \) used to pack all items. In cutting stock we are given a size vector \((w_i), \ i = 1, \ldots, s\), a multiplicity vector \(d_i, \ i = 1, \ldots, s\), and we want to minimize the number of bins of capacity \( b \) used to pack all items. So, in priciple the problems are equivalent. Yet, cutting stock uses a more “compact” input representation, thus:
• poly-time algorithm for BPP is not necessarily poly-time for CSP
• poly-size formulation for BPP is not necessarily poly-size for CSP

..because the size of the input of CSP can be exponentially small compared to BPP input size!

Example: Fixed Charge Network Design

This problem often appears in design problems that involve material flows in networks, such as water supply, heating, roads, telecomm. etc.

• We are given a directed graph \( G = (V, A) \).
• Each node \( i \in V \) has a demand/supply/transit \( b_i \).
• An arc \((i, j) \in A\) means that direct shipping from \( i \) to \( j \) is allowed (possibly with maximum capacity \( u_{ij} \) if the network is capacitated).
• There is also a variable cost \( c_{ij} \) associated with each unit of flow along arc \((i, j)\).
• There is a fixed cost \( d_{ij} \) associated with “opening” arc \((i, j)\) (think of this as the cost to “build” the link).
• We can associate two variables with each arc:
  – \( x_{ij} \) is a binary variable taking value 1 if arc \((i, j)\) is open, 0 otherwise.
  – \( f_{ij} \) is a nonnegative continuous variable representing the flow on arc \((i, j)\).

\[
\begin{align*}
\min \quad & \sum_{(i,j) \in A} (d_{ij} x_{ij} + c_{ij} f_{ij}) \\
\text{s.t.} \quad & \sum_{(i,j) \in A} f_{ij} - \sum_{(j,i) \in A} f_{ji} = b_i \quad \forall i \in V \\
& f_{ij} \leq M x_{ij} \quad \forall (i, j) \in A \\
& x \in \{0, 1\}^{|A|}, \quad f \in \mathbb{R}_+^{|A|}.
\end{align*}
\]

(5)

If the network is capacitated we also need: \( f_{ij} \leq u_{ij}, \forall (i, j) \in A \).

Note that if the network is capacitated, we can model dependency constraints (5) as well as capacity constraints as follows: \( f_{ij} \leq u_{ij} x_{ij} \quad \forall (i, j) \in A \).
3 Modeling a Restricted Set of Values

- We may want a variable $x$ to only take on values in the set $\{a_1, \ldots, a_m\}$
- We introduce $m$ binary variables $y_j$, $j = 1, \ldots, m$ and the following constraints:

$$
\begin{align*}
    x &= \sum_{j=1}^{m} a_j y_j \\
    \sum_{j=1}^{m} y_j &= 1 \\
    y &\in \{0,1\}^m.
\end{align*}
$$

- The set of variables $y_j$, $j = 1, \ldots, m$ is called a *special ordered set* (SOS) of variables.

4 Piecewise Linear Cost Functions

- We can use binary variables to model arbitrary piecewise linear cost functions.
- The function is specified by ordered pairs $(a_i, f(a_i))$ and we wish to evaluate it at point $x$.
- We can use a binary variable $y_i$, which indicates whether $a_i \leq x \leq a_{i+1}$.
- To evaluate the function, we will take linear combinations $\sum_{i=1}^{k} \lambda_i f(a_i)$ of the given function values.
- This works iff the only two nonzero $\lambda$’s are the ones corresponding to the endpoints of the interval in which $x$ lies... otherwise the description is not unique!
- Note that piecewise linear functions that are convex (concave) can be minimized (maximized) by linear programming because the slope of the segments are increasing (decreasing)... If $f(x)$ is *convex* piecewise linear we can represent it as $f(x) = \max_{i=1,\ldots,k-1} \{a_i^T x + b_i\}$:

$$
\min_{i=1,\ldots,k-1} \max \{a_i^T x + b_i\} \iff \min t : \ t \geq a_i^T x + b_i \ (i = 1, \ldots, k-1).
$$

If $f(x)$ is *concave* piecewise linear we can represent it as $f(x) = \min_{i=1,\ldots,k-1} \{a_i^T x + b_i\}$:

$$
\max_{i=1,\ldots,k-1} \min \{a_i^T x + b_i\} \iff \max t : \ t \leq a_i^T x + b_i \ (i = 1, \ldots, k-1).
$$

- Clearly, this does not work for *mini-min* and *maxi-max* problems!
• General piecewise linear functions are neither convex nor concave, so binary variables are needed to select the correct segment for a given value of $x$.

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{k} \lambda_i f(a_i) \\
\text{s.t.} & \quad \sum_{i=1}^{k} \lambda_i = 1 \\
& \quad \lambda_1 \leq y_1 \\
& \quad \lambda_i \leq y_{i-1} + y_i \quad (i = 2 \ldots k - 1) \\
& \quad \lambda_k \leq y_{k-1} \\
& \quad \sum_{i=1}^{k-1} y_i = 1 \\
& \quad y \in \{0,1\}^{k-1}, \quad \lambda \in \mathbb{R}^k.
\end{align*}
\]

Why making such a fuss of piecewise linear functions? The point is that an arbitrary continuous function of one variable can be approximated by a piecewise linear function with the quality of the approximation being controlled by the size of the linear segments. So, if we are given a separable function $f(x_1, \ldots, x_p) = \sum_{j=1}^{p} f(x_j)$, we know a “trick”!

## 5 Modeling Disjunctive Constraints

• We are given two constraints $a^T x \geq b$ and $c^T x \geq d$ with nonnegative coefficients. Instead of insisting both constraints be satisfied, we want at least one of the two constraints to be satisfied. To model this, we define a binary variable $y$ and impose the following:

\[
\begin{align*}
& a^T x \geq by \\
& c^T x \geq (1-y)d \\
& y \in \{0,1\}.
\end{align*}
\]

• More generally, we can impose that at least $k$ out of $m$ constraints be satisfied with the following:

\[
\begin{align*}
& (a_i)^T x \geq b_i y_i \\
& \sum_{i=1}^{m} y_i \geq k \\
& y \in \{0,1\}^m.
\end{align*}
\]
• Even more generally, suppose $P_i^i = \{ x \in \mathbb{R}_+^p : A_i^i x \leq b_i^i, x \leq d \}, i = 1 \ldots m$. Then

$\exists w : A_i^i x \leq b_i^i + w, \forall x \leq d, i = 1 \ldots m$. So, $\exists x$ contained in at least $k$ of the $P_i^i$ iff the following set is nonempty:

$$
A_i^i x \leq b_i^i + w(1 - y_i) \quad (i = 1 \ldots m)
\sum_{i=1}^m y_i \geq k
x \leq d
y \in \{0, 1\}^m, x \in \mathbb{R}_+^p.
$$

This follows because $y_i = 1$ yields the constraint $A_i^i x \leq b_i^i$, while $y_i = 0$ yields the redundant constraint $A_i^i x \leq b_i^i + w$.

• When $k = 1$ this is an alternative formulation:

$$
A_i^i x^i \leq b_i^i y_i \quad (i = 1 \ldots m)
\sum_{i=1}^m y_i = 1
\sum_{i=1}^m x^i = x
y \in \{0, 1\}^m, x^i \in \mathbb{R}_+^p \quad (i = 1 \ldots m).
$$

First, given that $x \in \mathbb{R}_+^p \cup_{i=1}^m P_i^i$, suppose w.l.o.g. that $x \in P_1^1$. Then a solution to the above model is $y_1 = 1, y_i = 0$ otherwise; $x^1 = x$, and $x^i = 0$ otherwise. On the other hand, suppose the above model has a solution, and, w.l.o.g., suppose $y_1 = 1, y_i = 0$ otherwise. Then we obtain $x = x^1$, and $x^i = 0$ otherwise. Thus, $x \in P_1^1$ and $x \in \mathbb{R}_+^p \cup_{i=1}^m P_i^i \neq \emptyset$.

Example: A scheduling problem

Disjunctive constraints arise naturally in scheduling problems where several jobs have to be processed on a machine and where the order in which they are processed is not specified. Thus we obtain disjunctive constraints of the type “either job $k$ precedes job $j$ on machine $i$ or vice versa”.

Suppose there are $n$ jobs and $m$ machines and each job must be processed on each machine. For each job, the machine order is fixed, that is, job $j$ must first be processed on machine $j(1)$ and then on machine $j(2)$, and so on. A machine can only process one job at a time, and once a job is started on any machine it must be processed to completion. The objective is to minimize the sum of the completion times of all the jobs. The data that specify an instance of the problem are:
1. \( m, n \) and \( p_{ij} \) for \( i = 1 \ldots m \) and \( j = 1 \ldots n \), which is the processing time of job \( j \) on machine \( i \).

2. the machine order \( j(1), j(2), \ldots, j(m) \) for each job \( j \).

To model this problem we use two types of variables:

- \( t_{ij} \) is a nonnegative continuous variable representing the start time of job \( j \) on machine \( i \).
- \( x_{ijk} \) is a binary variable taking value 1 if job \( j \) precedes job \( k \) on machine \( i \) (0 otherwise), where \( j < k \).

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} t_{j(m),j} \\
\text{s.t.} & \quad t_{j(r+1),j} \geq t_{j(r),j} + p_{j(r),j} \quad (r = 1 \ldots m - 1; j = 1 \ldots n) \\
& \quad t_{ij} - t_{ik} \leq -p_{ij} + M(1 - x_{ijk}) \quad (i = 1 \ldots m; 1 \leq j < k \leq n) \\
& \quad t_{ik} - t_{ij} \leq -p_{ik} + Mx_{ijk} \quad (i = 1 \ldots m; 1 \leq j < k \leq n) \\
& \quad x \in \{0,1\}^{m(n^2)}, \quad t \in \mathbb{R}^{mn}_+. 
\end{align*}
\]

6 Indicator Variables

We have already seen this trick here and there, but let’s look at the details. An **indicator variable** is a binary variable that indicates whether or not a constraint is satisfied. Given a constraint \( \sum_{j \in \mathcal{N}} a_j x_j \leq b \), let \( M \) and \( m \) denote respectively an UB and a LB on the value of \( \sum_{j \in \mathcal{N}} a_j x_j - b \), and \( \epsilon \) the tolerance.

- \( (\delta = 1 \implies \sum_{j \in \mathcal{N}} a_j x_j \leq b) \iff \sum_{j \in \mathcal{N}} a_j x_j \leq b + M(1 - \delta) \)
- \( (\sum_{j \in \mathcal{N}} a_j x_j \leq b \implies \delta = 1) \iff \sum_{j \in \mathcal{N}} a_j x_j \geq b + \epsilon + \delta(m - \epsilon) \)
- \( (\delta = 1 \implies \sum_{j \in \mathcal{N}} a_j x_j \geq b) \iff \sum_{j \in \mathcal{N}} a_j x_j \geq b + m(1 - \delta) \)
- \( (\sum_{j \in \mathcal{N}} a_j x_j \geq b \implies \delta = 1) \iff \sum_{j \in \mathcal{N}} a_j x_j \leq b - \epsilon + \delta(M + \epsilon) \)

**Exercise: Production Planning**

An engineering plant can produce 5 types of products: \( p_1, p_2, \ldots, p_5 \) by using two production processes: grinding and drilling. Each product requires the following number of hours for each process, and contributes the following (in hundreds of euros) to the net total profit.

- Max the total profit


Table 1: Production Planning Data

<table>
<thead>
<tr>
<th></th>
<th>p_1</th>
<th>p_2</th>
<th>p_3</th>
<th>p_4</th>
<th>p_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grind</td>
<td>12</td>
<td>20</td>
<td>0</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>Drill</td>
<td>10</td>
<td>8</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Profit</td>
<td>55</td>
<td>60</td>
<td>35</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

- Each unit of each product takes 20 man-hours for final assembly.
- The factory has 3 grinding machines and 2 drilling machines.
- The factory works a 6 day week with two shifts of 8 hours/day. Eight workers are employed in assembly, each working one shift per day.
- If we manufacture \( p_1 \) or \( p_2 \) (or both), then at least one of \( p_3, p_4, p_5 \) must also be manufactured.

We can model this as a MILP:

\[
\begin{align*}
\text{max} & \quad 55x_1 + 60x_2 + 35x_3 + 40x_4 + 20x_5 \\
\text{s.t.} & \quad 12x_1 + 20x_2 + 25x_4 + 15x_5 \leq 288 \\
& \quad 10x_1 + 8x_2 + 16x_3 \leq 192 \\
& \quad 20x_1 + 20x_2 + 20x_3 + 20x_4 + 20x_5 \leq 384 \\
& \quad x_i \leq Mz_i \quad (i = 1 \ldots 5) \\
& \quad z_1 + z_2 - 2\delta \leq 0 \\
& \quad z_3 + z_4 + z_5 - \delta \geq 0 \\
& \quad x \geq 0, \quad z \in \{0,1\}^5, \quad \delta \in \{0,1\}.
\end{align*}
\]

“Good” IP Models

In IP formulating a “good” model is of crucial importance to solving the model. A model consists of variables, objective function, and constraints. When addressing a model, the first question to ask is: “what are the variables?” Once we have defined the variables we can give an implicit representation of the problem:

\[
\max \{ c^Tx : x \in S \subset \mathbb{Z}_+^n \}
\]

where \( S \) is the set of feasible points in \( \mathbb{Z}_+^n \). We say that

\[
\max \{ c^Tx : Ax \leq b, \ x \in \mathbb{Z}_+^n \}
\]

is a valid formulation if \( S = \{ x \in \mathbb{Z}_+^n : Ax \leq b \} \). The point is: there are many choices for \((A,b)\)! Which one to choose?
Example

\[ S = \{(0000), (1000), (0100), (0010), (0001), (0110), (0101), (0011) \subset \{0, 1\}^4 \} \]

1. \[ S = \{x \in \mathbb{B}^4 : 93x_1 + 49x_2 + 37x_3 + 29x_4 \leq 111\} \]
2. \[ S = \{x \in \mathbb{B}^4 : 2x_1 + x_2 + x_3 + x_4 \leq 2\} \]
3. \[ S = \{x \in \mathbb{B}^4 : 2x_1 + x_2 + x_3 + x_4 \leq 2 \]

\[ \begin{align*}
    &x_1 + x_2 \leq 1 \\
    &x_1 + x_3 \leq 1 \\
    &x_1 + x_4 \leq 1
\end{align*} \]

Which one is the best?

Most IP algorithms require an UB (assuming max problem) on the value of the objective function, and the efficiency of the algorithm is very dependent on the sharpness of the bound. An UB is obtained, for example, relaxing the integrality constraints (aka continuous relaxation), so we solve

\[ z_{LP} = \max\{c^T x : Ax \leq b, x \in \mathbb{R}_+^n\} \]

since \( S \subset P = \{x : Ax \leq b, x \in \mathbb{R}_+^n\} \). Now, given two valid formulations defined by \((A^i, b^i), i = 1, 2\) let \( P^i = \{x : A^i x \leq b^i, x \in \mathbb{R}_+^n\} \) and \( z_{LP}^i = \max\{c^T x : x \in P^i\} \). Clearly, \( P^1 \subset P^2 \implies z_{LP}^1 \leq z_{LP}^2 \).

Also, it is instinctive to believe that computation time increases as the number of constraints increases. Yet, trying to find a formulation with a small number of constraints is often a bad strategy! In fact, one of the main algorithmic approaches involves a systematic addition of constraints, known as cutting planes or valid inequalities.

To understand how to generate valid inequalities for integer programs, it is first necessary to understand valid inequalities for polyhedra (or linear programs). So, the first question is: When is the inequality \( \pi^T x \leq \pi_0 \) valid for \( P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\} \)?

- \( \pi^T x \leq \pi_0 \) is valid for \( P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\} \neq \emptyset \) iff \( \pi^T x \leq \pi_0 \) is satisfied by all points in \( P \).

- More mathy...

- \( \pi^T x \leq \pi_0 \) is valid for \( P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\} \neq \emptyset \) iff \( \exists u \geq 0 \) such that \( u^T A \geq \pi \) and \( u^T b \leq \pi_0 \).
• Proof: The linear program \( \Pi = \max \{ \pi^T x \mid x \in P \} \) is feasible because \( P \neq \emptyset \) and bounded because \( \Pi \leq \pi_0 \). Therefore the dual \( D = \min \{ u^T b \mid u \geq 0, u^T A \geq \pi \} \) is also feasible and bounded. A basic optimal solution \( u \) proves the claim.

• Given two valid inequalities \( \pi^T x \leq \pi_0 \) and \( \mu^T x \leq \mu_0 \), we say \( \pi^T x \leq \pi_0 \) dominates \( \mu^T x \leq \mu_0 \) iff \( \exists \alpha > 0 \) s.t. \( \pi \geq \alpha \mu \) and \( \pi_0 \leq \alpha \mu_0 \) (if \( (\pi^T, \pi_0) = (\alpha \mu^T, \alpha \mu_0) \) they are equivalent).

• If \( \pi^T x \leq \pi_0 \) dominates \( \mu^T x \leq \mu_0 \) \( \implies \) \( \{ x \in \mathbb{R}^n_+ \mid \pi^T x \leq \pi_0 \} \subseteq \{ x \in \mathbb{R}^n_+ \mid \mu^T x \leq \mu_0 \} \).

• A valid inequality \( \mu^T x \leq \mu_0 \) is redundant, if there exist \( k \geq 1 \) valid inequalities \( \pi_i^T x \leq \pi_0^i, i = 1 \ldots k \), and weights \( \alpha_i > 0, i = 1 \ldots k \) such that \( \sum_{i=1}^k \alpha_i \pi_i^T x \leq \sum_{i=1}^k \alpha_i \pi_i^0 \) dominates \( \mu^T x \leq \mu_0 \).

Geometrically, we can see that there must be an infinite number of formulations, so how can we choose between them? The geometry again helps us to find the answer.

7 Graph Problems

7.1 A(n) (in)Famous Problem: TSP

TSP stands for Traveling Salesman Problem. Here we consider the asymmetric version (ATSP) which is defined on a directed graph. (The so-called symmetric or standard TSP is defined on an undirected graph). We are given a directed graph \( G = (V, A) \), where \( V = \{1, \ldots, n\} \) represents the set of cities that a salesman must visit. The arcs \( A \) represent ordered pairs of cities with direct travel possibility, and \( c_{ij} \) is the travel cost for arc \( (i, j) \). The salesman wishes to visit each of \( n \) cities exactly once and then return to his starting point. Find the order in which he should make his tour so as to do it with minimum cost.

To model this problem we can use a binary variable for each arc \( x_{ij} \) taking value one if the salesman goes directly from town \( i \) to town \( j \), and \( x_{ij} = 0 \) otherwise.

\[
\min \sum_{(ij) \in A} c_{ij} x_{ij} \quad (6)
\]

s.t. \[
\sum_{(ij) \in \delta^+(i)} x_{ij} = 1 \quad (i = 1, \ldots, n) \quad (7)
\]
\[
\sum_{(ij) \in \delta^-(j)} x_{ij} = 1 \quad (j = 1, \ldots, n) \quad (8)
\]
\[
\sum_{(ij) \in A(S)} x_{ij} \leq |S| - 1 \quad \forall S \subset V, \ 2 \leq |S| \leq n - 2 \quad (9)
\]
\[
x \in \{0, 1\}^{|A|}. \quad (10)
\]

Constraints (9) are called subtour elimination constraints. An alternative to (9) are the so-called cut-set constraints:

\[
\sum_{(ij) \in \delta^+(S)} x_{ij} \geq 1 \quad \forall S \subset V, \ 2 \leq |S| \leq n - 2. \quad (11)
\]
Cut-set constraints can also be written as follows:

$$\sum_{(ij) \in \delta^+(S)} x_{ij} \geq 1 \quad \forall S \subset V, \ r \in S$$

for an arbitrary \( r \in V \).

The two formulations (6)–(10) and (6)–(8), (11), (10) are equivalent (i.e., they provide the same lower bound), and are due to Dantzig, Fulkerson & Johnson, hence we call them DFJ. A key feature of DFJ is that constraints (9) (as well as (11)) are exponential in number, which makes cutting-planes methods necessary.

On the other hand, a wide variety of compact formulations exist, i.e., formulations with a polynomial number of both variables and constraints. All of them start by setting \( V = \{1, \ldots, n\} \) and viewing node 1 as a “depot”, which the salesman must leave at the start of the tour and return at the end of the tour. All of them can be used for the asymmetric TSP as well as for the symmetric TSP.

We start giving the (compact) formulation due to Miller, Tucker & Zemlin (MTZ). This formulation is based on the following observation: if the salesman travels from \( i \) to \( j \), then the position of node \( j \) is one more than that of node \( i \). So, for \( i = 2 \ldots n \) let \( u_i \) be a continuous variable representing the position of node \( i \) in the tour. (The depot can be thought of as being at position 0 and \( n \)). The MTZ formulation looks as follows:

$$\begin{align*}
\min & \quad \sum_{(ij) \in A} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{(ij) \in \delta^+(i)} x_{ij} = 1 \quad (i = 1, \ldots, n) \\
& \quad \sum_{(ij) \in \delta^-(j)} x_{ij} = 1 \quad (j = 1, \ldots, n) \\
& \quad u_i - u_j + (n-1)x_{ij} \leq n-2 \quad (2 \leq i, j \leq n; \ i \neq j) \quad (12) \\
& \quad 1 \leq u_i \leq n-1 \quad (2 \leq i \leq n) \quad (13) \\
& \quad x \in \{0,1\}^{|A|}.
\end{align*}$$

The MTZ formulation is compact, having only \( O(n^2) \) variables and \( O(n^2) \) constraints. Unfortunately, Padberg & Sung showed that its LP relaxation yields an extremely weak lower bound, much weaker than that of DFJ formulation.

Another compact formulation is due to Gavish & Groves and is called single-commodity flow (SCF). The idea is that the salesman carries \( n-1 \) units of a (single) commodity when he leaves the depot (i.e., node 1), and delivers one unit of this commodity to each other node:
\[
\min \sum_{(ij) \in A} c_{ij} x_{ij}
\]
\[
s.t. \quad \sum_{(ij) \in \delta^+(i)} x_{ij} = 1 \quad (i = 1, \ldots, n)
\]
\[
\sum_{(ij) \in \delta^-(j)} x_{ij} = 1 \quad (j = 1, \ldots, n)
\]
\[
\sum_{j=1}^n g_{ji} - \sum_{j=2}^n g_{ij} = 1 \quad (2 \leq i \leq n)
\]
\[
0 \leq g_{ij} \leq (n-1)x_{ij} \quad (1 \leq i, j \leq n; \ i \neq j)
\]
\[
x \in \{0,1\}^{|A|}.
\]

where continuous variables \(g_{ij}\) represent the amount of the commodity (if any) passing directly from node \(i\) to node \(j\). The constraints (14) ensure that one unit of the commodity is delivered to each non-depot node. The bounds (15) ensure that the commodity can only flow along edges that are in the tour. The SCF formulation has \(O(n^2)\) variables and \(O(n)\) constraints. The associated LB is intermediate between the DFJ and MTZ bounds.

Finally, we give a multicommodity flow (MCF) formulation by Claus. Here the salesman carries \(n-1\) commodities, one unit of each for each customer:

\[
\min \sum_{(ij) \in A} c_{ij} x_{ij}
\]
\[
s.t. \quad \sum_{(ij) \in \delta^+(i)} x_{ij} = 1 \quad (i = 1, \ldots, n)
\]
\[
\sum_{(ij) \in \delta^-(j)} x_{ij} = 1 \quad (j = 1, \ldots, n)
\]
\[
\sum_{j=1}^n f_{ki} - \sum_{j=2}^n f_{kj} = 0 \quad 2 \leq k \leq n; \ (2 \leq i \leq n); \ i \neq k
\]
\[
0 \leq f_{kj} \leq x_{ij} \quad (1 \leq i, j \leq n; \ i \neq j); \ 2 \leq k \leq n
\]
\[
\sum_{i=2}^n f_{ik} = 1 \quad 2 \leq k \leq n
\]
\[
\sum_{i=1}^n f_{ik} = 1 \quad 2 \leq k \leq n
\]
\[
x \in \{0,1\}^{|A|}.
\]

where the continuous variables \(f_{ij}^k\) represent the amount of the \(k\)-th commodity (if any) passing directly from node \(i\) to node \(j\). The constraints (17) state that a commodity cannot flow along an edge unless that edge belongs to the tour. The constraints (18), (19) impose that each commodity leaves the depot and arrives at its destination. The constraints (16) ensure that, when a commodity arrives at a node that is not its final destination, then it also leaves the node. The MCF formulation has \(O(n^3)\) variables and \(O(n^3)\) constraints. The associated LB is equal to the DFJ bound.

### 7.2 Stable Set

In graph theory, \textit{stable set} or \textit{independent set} is a (sub)set of vertices in a graph \(G = (V,E)\), no two of which are adjacent. That is, it is a (sub)set \(I \subseteq V\) of vertices such that for every
two vertices in $I$, there is no edge in $G$ connecting the two. Equivalently, each edge in the graph has at most one endpoint in $I$. The size of the stable set is the number of vertices it contains. A maximal stable set is either a stable set such that adding any other vertex to the set forces the set to contain an edge (or the set of all vertices of the empty graph). In the maximum stable set problem each vertex $i \in V$ is given a weight $p_i$. We wish to find a stable set of maximum weight, clearly if all vertices are given the same weight this corresponds to a stable set of maximum cardinality. The problem is also called vertex packing or node packing. The following is a valid formulation for the maximum stable set problem. Let $x_i$ be a binary variable associated to a vertex $i \in V$, the variable takes on value 1 if the vertex belongs to the stables set, 0 otherwise:

$$\begin{align*}
\text{max} & \quad \sum_{i \in V} p_i x_i \\
\text{s.t.} & \quad x_i + x_j \leq 1 \quad \forall (i, j) \in E \\
& \quad x \in \{0, 1\}^{|V|}.
\end{align*}$$

The maximum set packing problem with incidence matrix $A \in \{0, 1\}^{m \times n}$ and cost vector $c$, is equivalent to the stable set problem on the intersection graph $G(A) = (V, E)$ with vertices $V = \{1 \ldots n\}$, weights $p_j = c_j, j \in V$ and edges $E = \{(i, j) : a_{hi} = a_{hj} \text{ for some } h \in \{1 \ldots m\}\}$.

Also, the maximum stable set problem is equivalent to the maximum clique problem on the complement graph. A clique in an undirected graph $G = (V, E)$ is a subset of the vertex set $K \subseteq V$, such that for every two vertices in $K$, there exists an edge connecting the two. This is equivalent to saying that the subgraph induced by $K$ is complete. Every clique corresponds to a stable set (of the same size) in the complement graph $\bar{G} = (V, \bar{E})$.

Finally, the maximum stable set problem is equivalent to the minimum vertex cover on the same graph. A vertex cover (aka node cover) of a graph is a (sub)set of vertices $C \subseteq V$ such that each edge of the graph is incident to at least one vertex of the set $C$. A (sub)set of vertices $C$ is a vertex cover, if and only if its complement $I = V \setminus C$ is an independent set. Note that the above formulation can be strenthened using clique inequalities.

### 7.3 Vertex Colouring

We are given an undirected graph $G = (V, E)$, and we want to assign a colour to each vertex in $V$ such that:

- two adjacent vertices get a different colour each
- minimize the number of colours used

Let $n = |V|$. We use binary variables $y_j, j = 1 \ldots n$, taking value 1 if colour $j$ is used, 0 otherwise; and binary variables $x_{ij} = 1$ taking value one if colour $j$ is used for vertex $i \in V$. 

17
\[
\begin{align*}
\text{min} & \quad \sum_{j=1}^{n} y_j \\
\text{s.t.} & \quad x_{ij} + x_{hj} \leq y_j \quad \forall (i, h) \in E, \ j = 1, \ldots, n \quad (21) \\
& \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in V \quad (22) \\
& \quad y \in \{0, 1\}^n, \ x \in \{0, 1\}^{n^2}.
\end{align*}
\]

### 7.4 Network Flow Problems

The following are “easy” problems with TU constraint matrices.

#### 7.4.1 Minimum Cost Network Flow

Given a digraph \( G = (V, A) \) with arc capacities \( h_{ij} \) for all \((ij) \in A\), demands \( b_i \) at each node \( i \in V \), and unit flow costs \( c_{ij} \) for all \((ij) \in A\), we want to find a feasible flow that satisfies all the demands at minimum cost:

\[
\begin{align*}
\text{min} & \quad \sum_{(ij) \in A} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = b_i \quad (i \in V) \\
& \quad 0 \leq x_{ij} \leq h_{ij} \quad (i, j) \in A.
\end{align*}
\]

A generalization of this is the so called Min-Cost \textit{Multi-commodity Network Flow problem}. Again, we are given a network represented by a digraph \( G = (V, A) \), with arc capacities \( h_{ij} \) for all \((ij) \in A\), and unit flow costs \( c_{ij} \) for all \((ij) \in A\). Also, we are given \( K \) commodities, each one represented by a \textit{triple} \((s_k, t_k, b_k)\), i.e., source-sink-demand. We want to find a feasible flow for all \( K \) commodities that satisfies all the demands at minimum cost:

\[
\begin{align*}
\text{min} & \quad \sum_{k=1}^{K} \sum_{(ij) \in A} c_{ij} x_{ij}^k \\
\text{s.t.} & \quad \sum_{(ij) \in \delta^+(i)} x_{ij}^k - \sum_{(ji) \in \delta^-(i)} x_{ji}^k = b_i^k \quad (i \in V, \ k = 1 \ldots K) \\
& \quad \sum_{k=1}^{K} x_{ij}^k \leq h_{ij} \quad (i, j) \in A \\
& \quad x_{ij}^k \geq 0 \quad (i, j) \in A, \ k = 1 \ldots K.
\end{align*}
\]

Note that:

- LP is big: \( km \) variables, \( kn + m \) constraints, \( km \) non-negativity constraints. The size of constraint matrix is \( km(kn + m) = k^2 nm + km^2 \). So, this is a computationally challenging problem!
• Constraint matrix is not TU.
• Optimal solution to a multi-commodity flow LP might be fractional.
• Finding a feasible integer solution is \( \mathcal{NP} \)-hard.

### 7.4.2 Shortest Path

Given a digraph \( G = (V, A) \), two distinguished nodes \( s, t \in V \), and nonnegative arc costs \( c_{ij} \) for all \( (ij) \in A \), find a minimum cost \( s-t \) path:

\[
\begin{align*}
\min & \quad \sum_{(ij) \in A} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = 1 \quad (i = s) \\
& \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = 0 \quad (i \in V \setminus \{s, t\}) \\
& \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = -1 \quad (i = t) \\
& \quad x_{ij} \geq 0 \quad (i,j) \in A.
\end{align*}
\]

A \( \mathcal{NP} \)-hard version is the so called **Constrained Shortest Path Problem**:

\[
\begin{align*}
\min & \quad \sum_{(ij) \in A} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = 1 \quad (i = s) \\
& \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = 0 \quad (i \in V \setminus \{s, t\}) \\
& \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = -1 \quad (i = t) \\
& \quad \sum_{(ij) \in A} t_{ij} x_{ij} \leq T \\
& \quad x_{ij} \in \{0, 1\} \quad (i,j) \in A.
\end{align*}
\]

### 7.4.3 Maximum Flow

Given a digraph \( G = (V, A) \), two distinguished nodes \( s, t \in V \), and nonnegative arc capacities \( h_{ij} \) for all \( (ij) \in A \), find a maximum flow from \( s \) to \( t \). Adding a backward arc from \( t \) to \( s \), the maximum \( s-t \) flow problem can be formulated as:

\[
\begin{align*}
\max & \quad x_{ts} \\
\text{s.t.} & \quad \sum_{(ij) \in \delta^+(i)} x_{ij} - \sum_{(ji) \in \delta^-(i)} x_{ji} = 0 \quad (i \in V) \\
& \quad 0 \leq x_{ij} \leq h_{ij} \quad (i,j) \in A.
\end{align*}
\]
Taking the dual we get:

\[
\min \sum_{(ij) \in A} h_{ij} w_{ij} \\
\text{s.t. } u_i - u_j + w_{ij} \geq 0 \quad (i, j) \in A \\
\quad u_t - u_s \geq 1.
\]

which corresponds to the minimum \(s-t\) cut problem:

\[
\min \left\{ \sum_{(ij) \in \delta^+(X)} h_{ij} : s \in X \subset V \setminus \{t\} \right\}.
\]

### 7.4.4 Transportation

Given a digraph \(G = (V, A)\), the set of nodes is partitioned into two subsets \(S = \{1, \ldots, m\}\) and \(D = \{1, \ldots, n\}\), respectively the set of node-sources and the set of node-demands. Each source \(i \in S\) has a quantity \(s_i\) to send, and each destination \(j \in D\) has a quantity \(d_j\) to receive. The cost of shipping one unit of material from source \(i\) to destination \(j\) is \(c_{ij}\). We look for a shipping from sources to destinations with minimum cost:

\[
\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t. } \sum_{j=1}^{n} x_{ij} = s_i \quad (i = 1 \ldots m) \\
\quad \sum_{i=1}^{m} x_{ij} = d_j \quad (j = 1 \ldots n) \\
\quad x_{ij} \geq 0 \quad (i, j) \in A.
\]

Note that the problem is feasible iff \(\sum_{i \in S} s_i = \sum_{j \in D} d_j\). Also, note that if \(m = n\) and \(s_i = d_j = 1\) for all \(i, j\), the problem corresponds to the Assignment problem, which is indeed another “easy” problem.

### 7.5 Minimum Spanning Tree

We are given an undirected graph \(G = (V, E)\), a subset of edges \(T \subseteq E\) is a spanning tree iff any two of the following conditions hold:

1. connected
2. acyclic
3. \(n - 1\) edges
We use conditions 2 and 3 and look for a minimum cost spanning tree in $G$:

\[
\begin{align*}
\min & \quad \sum_{e \in E} w_e x_e \\
\text{s.t.} & \quad \sum_{e \in E} x_e = n - 1 \\
& \quad \sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall S \subset V \\
& \quad x_e \in \{0, 1\}^{|E|}.
\end{align*}
\]

(23)

Note that:

- MST is an “easy” problem, but the constraint matrix is not TU and the number of constraints is exponential.
- This formulation corresponds to the integral hull of MST incidence vectors.
- Another valid formulation can be obtained by substituting subtour elimination (23) constraints with the following connectivity constraints:

\[
\sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \subset V.
\]

(24)

For a di-graph $G = (A, V)$ the problem is known as shortest spanning arborescence with root $r$ (SSA-$r$), and has the following formulation:

\[
\begin{align*}
\min & \quad \sum_{(ij) \in A} w_{ij} x_{ij} \\
& \quad \sum_{(ij) \in \delta^{-}(j)} x_{ij} = 1 \quad \forall j \in V, j \neq r \\
& \quad \sum_{(ir) \in \delta^{-}(r)} x_{ir} = 0 \\
& \quad \sum_{(ij) \in \delta^{+}(S)} x_{ij} \geq 1 \quad \forall S \subset V, \ r \in S \\
& \quad x_{ij} \in \{0, 1\}^{|A|}.
\end{align*}
\]

(25)

Constraints (25) can be substituted by the following cardinality constraints:

\[
\begin{align*}
\text{s.t.} & \quad \sum_{(ij) \in A} x_{ij} = n - 1
\end{align*}
\]

(27)

(SSA-$r$) is an “easy” problem and can be solved in $O(n^2)$ time.
7.6 Maximum Matching

We are given an undirected graph \( G = (V, E) \), a subset of edges \( M \subseteq E \) is a matching in \( G \) iff all edges in \( M \) are non-incident. We look for a maximum weight matching in \( G \):

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} w_e x_e \\
\text{s.t.} & \quad \sum_{e \in \delta(i)} x_e \leq 1 \quad \forall i \in V \\
& \quad x_e \in \{0, 1\}^{|E|}. 
\end{align*}
\]

(28)

Note that:

- Matching is an “easy” problem, but the formulation above with degree constraints (28), although valid, does not describe the integral hull of matching incidence vectors.

- In fact, we need to add the following odd-set inequalities:

\[
\sum_{e \in E(S)} x_e \leq \frac{|S| - 1}{2} \quad \forall S \subseteq V, \ |S| \geq 3 \text{ odd}
\]

(29)

- The formulation with (28) and (29) describes the integral hull of matching incidence vectors.

- So, matching is another “easy” problem where the constraint matrix is not TU and the number of constraints is exponential.

Note that:

- Maximum Matching and Minimum Vertex Cover form a weak dual pair.

- König Theorem: Strong duality holds if and only if the graph is bipartite, indeed, if the graph is bipartite, the constraint matrix is TU.

- The (linear) assignment problem corresponds to a minimum cost perfect matching problem on a bipartite graph.

8 Hard feasibility problems

Example: Satisfability Problem (SAT)

This is a \( \mathcal{NP} \)-complete feasibility problem.

We are given:

- a finite set \( N = \{1, \ldots, n\} \) (aka the literals)
• $m$ pairs of subsets of $N$, $C_i = (C_i^+, C_i^-)$ (aka the *clauses*)
• an instance is feasible iff the set:

$$\{ x \in \mathbb{B}^n \mid \sum_{j \in C_i^+} x_j + \sum_{j \in C_i^-} (1 - x_j) \geq 1 \quad i = 1, \ldots, m \}$$

is nonempty.

**Example: Partition Problem**

This is a another $NP$-complete feasibility problem. We are given:
• a finite set $N = \{1, \ldots, n\}$ of items with weights $(a_1, a_2, \ldots, a_n)$
• an integer $b$
• an instance is feasible iff the set:

$$\{ x \in \mathbb{B}^n \mid \sum_{j=1}^n a_j x_j = b \}$$

is nonempty.
• The problem is still $NP$-complete if $b = \frac{\sum_{j=1}^n a_j}{2}$

9 **The Combinatorial Explosion: Bang!**

Assignment, 0-1 Knapsack, Set Covering, TSP, are all *combinatorial* problems, in the sense that the feasible solution is some subset of a finite set. Thus, in principle, they could be solved by *enumeration*:
• Assignment: all possible permutations of $\{1, \ldots, n\} \implies n!$
• 0-1 KP and Set Covering: all possible subsets of $\{1, \ldots, n\} \implies 2^n$
• TSP: all possible Hamiltonian tours $\implies (n - 1)!$

Using complete enumeration we can only hope to solve such problems for very small values of $n$. So we need sth more clever...