Modelling with binary variables I

Lorenzo Castelli, Università degli Studi di Trieste.

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Combinatorial optimisation

Let

$N = \{1, \ldots, n\}$ be a finite set

$c = \{c_1, \ldots, c_n\}$ be a $n-$vector

For $F \subseteq N$ define $c(F) = \sum_{j \in F} c_j$.
Suppose we are given a collection of subsets $\mathcal{F}$ of N.
The combinatorial optimisation problem is

$$\max\{c(F) : F \in \mathcal{F}\}.$$

A binary choice

Consider an event that may or may not occur, and suppose that it is part of the problem to decide between these two possibilities. To model such a dichotomy, we use a binary variable $\boldsymbol{x}$ and let

$$\boldsymbol{x} = \begin{cases} \mathbf{1} & \text{if the event occurs} \\ \mathbf{0} & \text{if the event does not occur} \end{cases}$$

The 0–1 Knapsack problem

Suppose there are n projects. The jth project, $j = 1, \ldots, n$ has a cost of a_j and a value of C_j. Each project is either done or not, that is, it is not possible to do a fraction of any of the projects. Also there is a budget of b available to fund the projects. The problem of choosing a subset of the projects to maximise the sum of the values while not exceeding the budget constraint is the 0–1 knapsack problem

$$\max \left\{ \sum_{j=1}^{n} C_j x_j : \sum_{j=1}^{n} a_j x_j \leq b, x \in \{0, 1\}^n \right\}$$

The 0–1 Knapsack Problem

Here the jth event is the jth project. This problem is called the knapsack problem because of the analogy to the hiker's problem of deciding what should be put in a knapsack, given a weight limitation on how much can be carried.

In general, problems of this sort may have several constraints. We then refer to the problem as the multidimensional knapsack problem.

The 0-1 Knapsack Problem – Example

Suppose you are planning a picnic. You've constructed a list of items you would like to carry with you on the picnic. Each item has a weight associated with it and your knapsack is limited to carrying no more than 15 pounds. You have also come up with a 1 to 10 rating for each item, which indicates how strongly you want to include the particular item in the knapsack for the picnic. This information is listed in the next table.

Item	Weight	Rating
Ant Repellent	1	2
Beer	3	9
Blanket	4	3
Bratwurst	3	8
Brownies	3	10
Frisbee	1	6
Salad	5	4
Watermelon	10	10

The 0-1 Knapsack Problem - Example

$$
\begin{aligned}
\max \quad & 2x_1 + 9x_2 + 3x_3 + 8x_4 + 10x_5 + 6x_6 + 4x_7 + 10x_8 \\
& x_1 + 3x_2 + 4x_3 + 3x_4 + 3x_5 + x_6 + 5x_7 + 10x_8 \leq 15
\end{aligned}
$$

$$x_j \geq 0 \text{ for } j = 1, \ldots, 8$$

The Assignment problem (I)

Suppose there are n people and m jobs, where $n \geq m$. Each job must be done by exactly one person; also, each person can do, at most, one job. The cost of person j doing job i is c_{ij}. The problem is to assign the people to the jobs so as to minimise the total cost of completing all of the jobs. To formulate this problem, which is known as the assignment problem, we introduce 0-1 variables $x_{ij}, i = 1, \ldots, m, j = 1, \ldots, n$ corresponding to the ijth event of assigning person j to job i.

The Assignment problem (II)

Since exactly one person must do job i, we have the constraints

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, \dots, m \tag{1}$$

Since each person can do no more than one job, we also have the constraints

$$\sum_{i=1}^{m} x_{ij} \leq 1 \quad \text{for } j = 1, \dots, n \tag{2}$$

It is now easy to check that if $x \in \{0, 1\}^{mn}$ satisfies (1) and (2), we obtain a feasible solution to the assignment problem. The objective function is

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

The Assignment problem – Example (I)

A company has 4 machines available for assignment to 4 tasks. Any machine can be assigned to any task, and each task requires processing by one machine. The time required to set up each machine for the processing of each task is given in the table below.

	Task 1	Task 2	Task 3	Task 4
Machine 1	13	4	7	6
Machine 2	1	11	5	4
Machine 3	6	7	2	8
Machine 4	1	3	5	9

The company wants to minimise the total setup time needed for the processing of all four tasks.

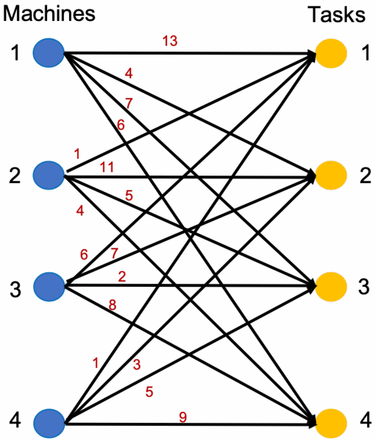
The Assignment problem - Example (II)

If we think of the setup times as costs and define

$$x_{ij} = \begin{cases} 1 & \text{if machine } i \text{ is assigned to process task } j \\ 0 & \text{if machine } i \text{ is not assigned to process task } j \end{cases}$$

where $i = 1, 2, 3, 4$ and $j = 1, 2, 3, 4$, then it is easily seen that what we have is a problem with 4 sources (representing the machines), 4 sinks (representing the tasks), a single unit of supply from each source (representing the availability of a machine), and a single unit of demand at each sink (representing the processing requirement of a task). This particular class of problems is called the assignment problems.

The Assignment problem - Example (III)



The Assignment problem – Example (IV)

$$
\begin{aligned}
\min 13x_{11} + 4x_{12} + 7x_{13} + 6x_{14} + \\
x_{21} + 11x_{22} + 5x_{23} + 4x_{24} + \\
6x_{31} + 7x_{32} + 2x_{33} + 8x_{34} + \\
x_{41} + 3x_{42} + 5x_{43} + 9x_{44}
\end{aligned}
$$

$$
\begin{aligned}
x_{11} + x_{12} + x_{13} + x_{14} &= 1 \\
x_{21} + x_{22} + x_{23} + x_{24} &= 1 \\
x_{31} + x_{32} + x_{33} + x_{34} &= 1 \\
x_{41} + x_{42} + x_{43} + x_{44} &= 1
\end{aligned}
$$

$$
\begin{aligned}
x_{11} + x_{21} + x_{31} + x_{41} &= 1 \\
x_{12} + x_{22} + x_{32} + x_{42} &= 1 \\
x_{13} + x_{23} + x_{33} + x_{43} &= 1 \\
x_{14} + x_{24} + x_{34} + x_{44} &= 1
\end{aligned}
$$

$x_{ij} \in \{0, 1\}, \text{ for } i = 1, \ldots, 4, j = 1, \ldots, 4$

The Matching problem

In the assignment problem the $m + n$ elements are partitioned into disjoint sets of jobs and people. But in other models of this type, we cannot assume such a partition. Suppose $2n$ students are to be assigned to n double rooms. Here each student must be assigned exactly one roommate. Let the ijth event, $i < j$, correspond to assigning students i and j to the same room; also suppose that there is a value of c_{ij} when students i and j are roommates. The problem

$$\left\{ \max \sum_{i=1}^{2n-1} \sum_{j=i+1}^{2n} c_{ij} x_{ij} : \sum_{k<i} x_{ki} + \sum_{j>i} x_{ij} = 1, i = 1, \ldots, 2n, x \in \{0,1\}^{n(2n-1)} \right\} \tag{3}$$

is known as the perfect matching problem. If the equality constraints in (3) are replaced by equal-to-or-less-than inequalities, then the problem is called the matching problem.

The Perfect Matching problem

We have an even number n of persons that need to me matched into pairs in order to perform a certain job. If person i is matched with person j there is a cost c_{ij}. A matching is a paring of persons, so that each individual is matched with exactly one other individual. The goal is to find a matching that minimises the total cost.

The Perfect Matching problem

We represent the set of people by an undirected graph $G = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of individuals, and the cost of edge $e = \{i, j\}$ is c_e. If $\{i, j\} \notin \mathcal{E}$ this indicates that i and j cannot be matched. If we define

$$x_e = \begin{cases} 1 & \text{if edge } e = \{i, j\} \text{ is selected, i.e., persons } i \text{ and } j \text{ are matched} \\ 0 & \text{otherwise} \end{cases}$$

the perfect matching problem can be formulated as follows:

$$\begin{aligned} \min \sum_{e \in \mathcal{E}} c_e x_e & \\ \sum_{e \in \delta(\{i\})} x_e = 1, & \qquad \forall i \in \mathcal{N} \\ x_e \in \{0, 1\} & \qquad \forall e \in \mathcal{E} \end{aligned}$$

where $\delta(\{i\})$ is the set on edges incident to i.

The Perfect Matching problem

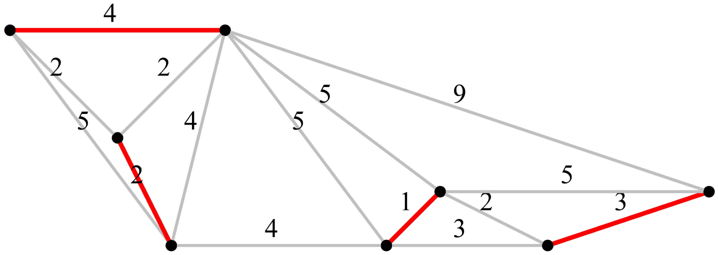


Figure: *The perfect matching problem, 8 nodes, 15 edges*

The Perfect Matching problem

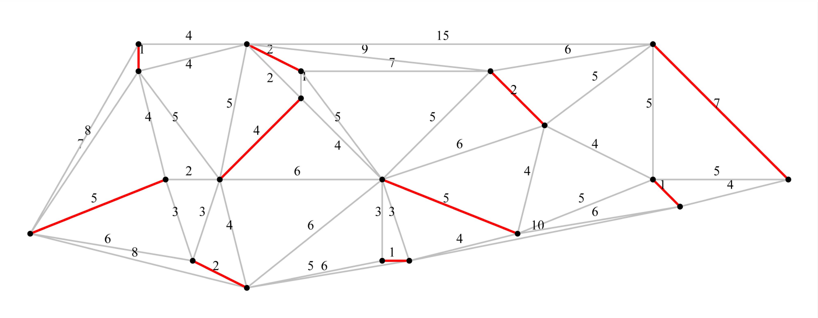


Figure: *The perfect matching problem, 20 nodes, 49 edges*

Maximum cardinality matching

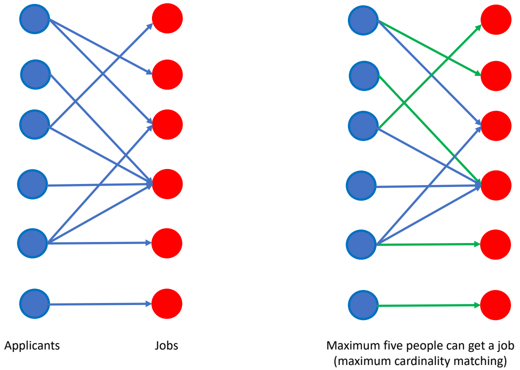


Figure: *The perfect matching does not always exist*

The Set-Covering problem

Let $\boldsymbol{M} = \{\mathbf{1}, \ldots, \boldsymbol{m}\}$ be a finite set and let $\{\boldsymbol{M}\}$ for $\boldsymbol{j} \in \boldsymbol{N} = \{\mathbf{1}, \ldots, \boldsymbol{n}\}$ be a given collection of subsets of $\boldsymbol{M}$. For example, the collection might consist of all subsets of size $\boldsymbol{k}$, for some $\boldsymbol{k} \leq \boldsymbol{m}$. We say that $\boldsymbol{F} \subseteq \boldsymbol{N}$ covers $\boldsymbol{M}$ if $\bigcup_{\boldsymbol{j} \in \boldsymbol{F}} \boldsymbol{M_j} = \boldsymbol{M}$. The set-covering problem is defined as

$$\min\{\boldsymbol{c}(\boldsymbol{F}) : \boldsymbol{F} \in \mathcal{F}\} \quad \text{where}$$
$$\mathcal{F} = \{\boldsymbol{F} : \boldsymbol{F} \text{ covers } \boldsymbol{M}\}$$

Set-Covering - Example

A typical application concerns facility location. Suppose we are given a set of potential sites $N = \{1, \ldots, n\}$ for the location of fire stations. A station placed at j costs c_j. We are also given a set of communities $M = \{1, \ldots, m\}$ that have to be protected. The subset of communities that can be protected from a station located at j is M_j. For example, M_j might be the set of communities that can be reached from j in 10 minutes. Then the problem of choosing a minimum-cost set of locations for the fire stations such that each community can be reached from some fire station in 10 minutes is a set-covering problem.

Set-Covering - Formulation

Decision variables

$$x_j = \begin{cases} 1 & \text{if } j \in F \\ 0 & \text{if } j \notin F \end{cases}$$

Objective function

$$\min \sum_{j=1}^{n} c_j x_j$$

Set-Covering - Formulation

Constraints Let $\boldsymbol{A}$ be the $\boldsymbol{m} \times \boldsymbol{n}$ incidence matrix of the family $\{\boldsymbol{M_j}\}$ for $\boldsymbol{j} \in \boldsymbol{N}$, that is, for $\boldsymbol{i} \in \boldsymbol{M}$,

$$a_{ij} = \begin{cases} 1 & \text{if } i \in M_j \\ 0 & \text{if } i \notin M_j \end{cases}$$

Then $\boldsymbol{F}$ is a cover if and only if $\boldsymbol{x} \in \{0, 1\}^n$ satisfies

$$Ax \geq 1$$

where $\mathbf{1}$ is a $\boldsymbol{m}$—vector all of whose components are equal to 1. Alternatively,

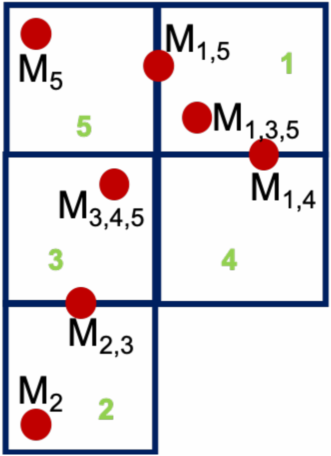
$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \text{for } i = 1, \ldots, m$$

Set-Covering - Example (I)

Let $M = \{1, 2, 3, 4, 5\}$,
$\{M\} = \{\{1, 3, 5\}, \{2, 3\}, \{1, 4\}, \{3, 4, 5\}, \{2\}, \{5\}, \{1, 5\}\}$,
$c = [3, 5, 1, 9, 2, 4, 1]$. Hence,

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

Set-Covering - Example (II)



Set-Covering - Example (III)

$$
\begin{array}{lllllllll}
\min & 3x_1 & +5x_2 & + x_3 & +9x_4 & + 2x_5 & +4x_6 & + x_7 & \\
(1) & x_1 & & + x_3 & & & & + x_7 & \geq 1 \\
(2) & & x_2 & & & + x_5 & & & \geq 1 \\
(3) & x_1 & +x_2 & & +x_4 & & & & \geq 1 \\
(4) & & & + x_3 & +x_4 & & & & \geq 1 \\
(5) & x_1 & & & +x_4 & & +x_6 & + x_7 & \geq 1
\end{array}
$$

$$x_j \in \{0, 1\}, j = 1, \ldots, 7$$

Set-Covering - Example 2

The director of a television channel for local information must organise the work of the teams of journalists and operators to cover m different off-site services. The editor-in-chief has prepared n possible activities that a each individual team can carry out, where an activity is the set of services that can be performed and involves a certain cost of remuneration for the team, including travel costs and any overtime hours. The director must decide which of the activities to do in order to pay as little as possible with the guarantee that each of the services is “covered” by at least one team.

The Set-Packing problem

We say that $F \subseteq N$ is a packing with respect to M if $M_j \cap M_k = \emptyset$ for all $j, k \in F, j \neq k$. In the set-packing problem

$$\mathcal{F} = \{F : F \text{ is a packing with respect to } M\}$$

In the set-packing problem c_j is the weight or value of M_j and we see a maximum-weight packing:

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \text{for } i = 1, \dots, m$$

$$x_j \in \{0, 1\} \quad \text{for } j = 1, \dots, n$$

The Set-Packing problem - Example

The famous shop window designer Ulivetto Laziali was called to set up the shop window of the most important florist of Trieste. With the m flowers, of different shape and colour, that the florist provided him, Ulivetto produces n different sketches of floral arrangements and for each of them also provides a score of "compositional beauty". He therefore decides to set up a set of floral arrangements in the shop window that maximises the overall score (defined as the sum of the scores of the individual compositions created). The problem is clearly a packing problem as it cannot create two compositions containing the same flower.

The Set-Partitioning problem

If $F \subseteq N$ is both a covering and a packing, then F is said to be a partition of M. Hence, the set-partitioning problem can be formulated as:

$$\begin{aligned} \text{max (or min)} \ \sum_{j=1}^{n} c_j x_j & \\ \sum_{j=1}^{n} a_{ij} x_j = 1 & \quad \text{for } i = 1, \ldots, m \\ x_j \in \{0, 1\} & \quad \text{for } j = 1, \ldots, n \end{aligned}$$

Note that an assignment problem with m jobs and m people is a set-partitioning problem in which $M = \{1, \ldots, m, m+1, \ldots, 2m\}$ and M_j for $j = 1, \ldots, m^2$ is a subset of M consisting of one job and one person.

The Set-Partitioning problem - Example

The manager of a construction site in Vicenza for the new high-speed line must contract out m different works. The procurement office has prepared several possible contract assignments, each of which is made up of a subset of the works which, for reasons of efficiency, it is good that they are performed by the same company. For each contract assignment is also defined the cost. The problem is to decide which contracts to award so that all the works are carried out and the cost of the contracts is minimum. This problem can easily be formulated as a partition problem.