Mathematical Programming with Applications to Economics

Debasis Mishra[1]

April 16, 2015

[1]Economics and Planning Unit, Indian Statistical Institute, 7 Shahid Jit Singh Marg, New Delhi 110016, India, E-mail: dmishra@isid.ac.in

Contents

Chapter 1

Basic Graph Theory

1.1 What is a Graph?

Let $N = \{1, \ldots, n\}$ be a finite set. Let E be a collection of ordered or unordered pairs of distinct [1] elements from N. A **graph** G is defined by (N, E). The elements of N are called **vertices** or **nodes** of graph G. The elements of E are called **edges** of graph G. If E consists of ordered pairs of vertices, then the graph is called a **directed graph** (digraph). When we say graph, we refer to undirected graph, i.e., E consists of unordered pairs of vertices. To avoid confusion, we write an edge in an undirected graph as $\{i, j\}$ and an edge in a directed graph as (i, j).

Figure 1.1 gives three examples of graphs. The rightmost graph in the figure is a directed graph. In all the graphs $N = \{1, 2, 3, 4\}$. In the leftmost graph in Figure 1.1, $E = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$. In the directed graph, $E = \{(1, 2), (1, 3), (2, 3), (3, 2), (3, 4), (4, 1)\}$.

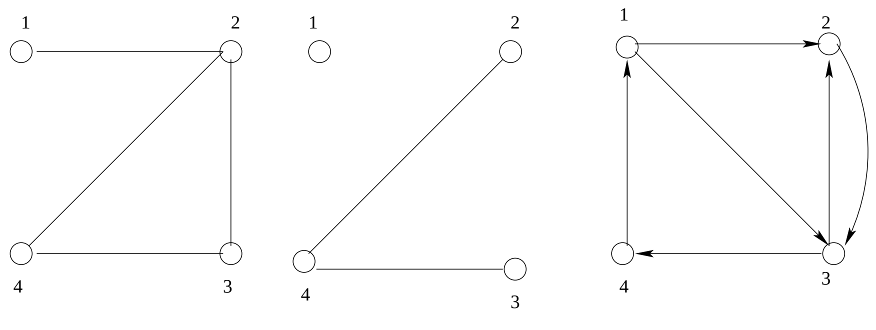


Figure 1.1: Examples of graph

[1]In standard graph theory, we do not require this distinct restriction.

Often, we associate weights to edges of the graph or digraph. These weights can represent capacity, length, cost etc. of an edge. Formally, a weight is a mapping from set of edges to real numbers, $w : E \to \mathbb{R}$. Notice that weight of an edge can be zero or negative also. We will learn of some economic applications where this makes sense. If a weight system is given for a (di)graph, then we write (N, E, w) as the (di)graph.

1.1.1 Modeling Using Graphs: Examples

Example 1: Housing/Job Market

Consider a market of houses (or jobs). Let there be $H = \{a, b, c, d\}$ houses on a street. Suppose $B = \{1, 2, 3, 4\}$ be the set of potential buyers, each of whom want to buy exactly one house. Every buyer $i \in B$ is interested in $\emptyset \neq H_i \subseteq H$ set of houses. This can be modeled using a graph.

Consider a graph with the following set of vertices: $N = H \cup B$ (note that $H \cap B = \emptyset$). The *only* edges of the graph are of the following form: for every $i \in B$ and every $j \in H_i$, there is an edge between i and j. Graphs of this kind are called **bipartite** graphs, i.e., a graph whose vertex set can be partitioned into two non-empty sets and the edges are only between vertices which lie in separate parts of the partition.

Figure 1.2 is a bipartite graph of this example. Here, $H_1 = \{a\}, H_2 = \{a, c\}, H_3 = \{b, d\}, H_4 = \{c\}$. Is it possible to allocate a unique house to every buyer?

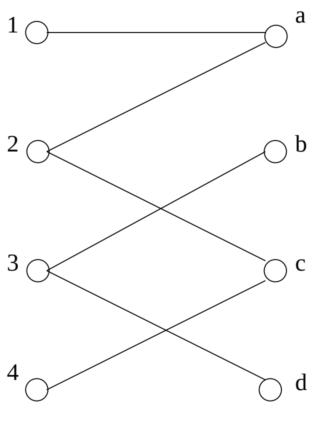


Figure 1.2: A bipartite graph

If every buyer associates a value for every house, then it can be used as a weight of the graph. Formally, if there is an edge (i, j) then $w(i, j)$ denotes the value of buyer $i \in B$ for house $j \in H_i$.

Example 2: Coauthor/Social Networking Model

Consider a model with researchers (or agents in Facebook site). Each researcher wants to collaborate with some set of other researchers. But a collaboration is made only if both agents (researchers) put substantial effort. The effort level of agent i for edge (i, j) is given by $w(i, j)$. This situation can be modeled as a directed graph with weight of edge (i, j) being $w(i, j)$.

Example 3: Transportation Networks

Consider a reservoir located in a state. The water from the reservoir needs to be supplied to various cities. It can be supplied directly from the reservoir or via another cities. The cost of supplying water from city i to city j is given and so is the cost of supplying directly from reservoir to a city. What is the best way to connect the cities to the reservoir?

The situation can be modeled using directed or undirected graphs, depending on whether the cost matrix is asymmetric or symmetric. The set of nodes in the graph is the set of cities and the reservoir. The set of edges is the set of edges from reservoir to the cities and all possible edges between cities. The edges can be directed or undirected. For example, if the cities are located at different altitudes, then cost of transporting from i to j may be different from that from j to i, in which case we model it as a directed graph, else as an undirected graph.

1.2 Definitions of (Undirected) Graphs

If $\{i, j\} \in E$, then i and j are called **end points** of this edge. The **degree** of a vertex is the number of edges for which that vertex is an end point. So, for every $i \in N$, we have $deg(i) = |\{j \in N : \{i, j\} \in E\}|$. In Figure 1.1, degree of vertex 2 is 3. Here is a simple lemma about degree of a vertex.

Lemma **1** *The number of vertices of odd degree is even.*

Proof: Let O be the set of vertices of odd degree. Notice that if we take the sum of the degrees of all vertices, we will count the number of edges exactly twice. Hence, $\sum_{i \in N} deg(i) =$

$2|E|$. Now, $\sum_{i \in N} deg(i) = \sum_{i \in O} deg(i) + \sum_{i \in N \setminus O} deg(i)$. Hence, we can write,

$$\sum_{i \in O} deg(i) = 2|E| - \sum_{i \in N \setminus O} deg(i).$$

Now, right side of the above equation is even. This is because $2|E|$ is even and for every $i \in N \setminus O$, $deg(i)$ is even by definition. Hence, left side of the above equation $\sum_{i \in O} deg(i)$ is even. But for every $i \in O$, $deg(i)$ is odd by definition. Hence, $|O|$ must be even. ■

A **path** is a sequence of *distinct* vertices $(i^1, \ldots, i^k)$ such that $\{i^j, i^{j+1}\} \in E$ for all $1 \leq j < k$. The path $(i^1, \ldots, i^k)$ is called a path from i^1 to i^k. A graph is **connected** if there is a path between every pair of vertices. The middle graph in Figure 1.1 is not connected.

A **cycle** is a sequence of vertices $(i^1, \ldots, i^k, i^{k+1})$ with $k > 2$ such that $\{i^j, i^{j+1}\} \in E$ for all $1 \leq j \leq k$, $(i^1, \ldots, i^k)$ is a path, and $i^1 = i^{k+1}$. In the leftmost graph in Figure 1.1, a path is $(1, 2, 3)$ and a cycle is $(2, 3, 4, 2)$.

A graph $G' = (N', E')$ is a **subgraph** of graph $G = (N, E)$ if $\emptyset \neq N' \subseteq N$, $E' \subseteq E$, and for every $\{i, j\} \in E'$ we have $i, j \in N'$. A connected acyclic (that does not contain a cycle) graph is called a **tree**. Graphs in Figure 1.1 are not trees, but the second and third graph in Figure 1.3 are trees. A graph may contain several trees (i.e. connected acyclic subgraphs). The **spanning tree** of a connected graph $G \equiv (N, E)$ is a subgraph (N, E') which is a tree. Note that $E' \subseteq E$ and the set of vertices in a spanning tree is the same as the original graph - in that sense, a spanning tree *spans* the original graph. By definition, every tree (N', E') is a spanning tree of itself.

Figure 1.3 shows a connected graph (which is not a tree) and two of its spanning trees.

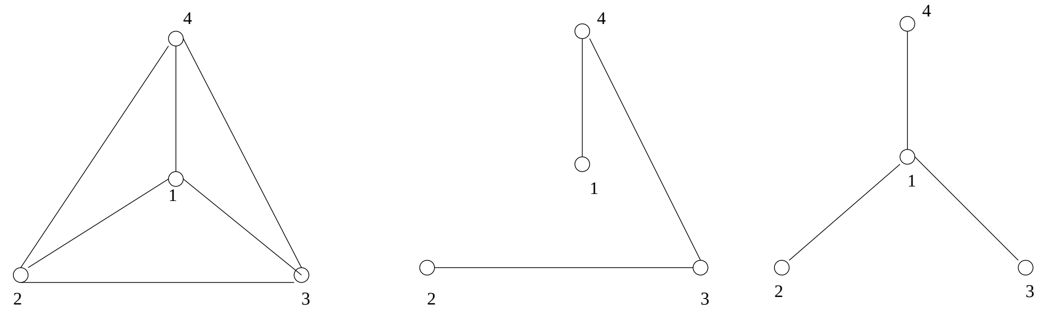


Figure 1.3: Spanning trees of a graph

A subgraph $G' = (N', E')$ of a graph $G = (N, E)$ is a **component** of G if G' is connected and there does not exist vertices $i \in N'$ and $j \in N$ such that $\{i, j\} \in E \setminus E'$. Hence, G' is a maximally connected subgraph of G.

Clearly, any graph can be partitioned into its components. A connected graph has only one component, which is the same graph.

1.2.1 Properties of Trees and Spanning Trees

We now prove some properties of trees and spanning trees.

PROPOSITION **1** *Every tree $G' = (N', E')$, where G' is a subgraph of a graph $G = (N, E)$, satisfies the following properties.*

1. *There is a unique path from i to j in G' for every $i, j \in N'$.*
2. *If there is an edge $\{i, j\} \in E \setminus E'$ with $i, j \in N'$, adding $\{i, j\}$ to E' creates a cycle.*
3. *By removing an edge from E' disconnects the tree. In particular, if an edge $\{i, j\}$ is removed, then it creates two components one containing i and the other containing j.*
4. *Every tree with at least two vertices has at least two vertices of degree one.*
5. $|E| = |N| - 1$.

Proof:

1. Suppose there are at least two paths from i to j. Let these two paths be $P_1 = (i, i^1, \ldots, i^k, j)$ and $P_2 = (i, j^1, \ldots, j^q, j)$. Then, consider the following sequence of vertices: $(i, i^1, \ldots, i^k, j, j^q, \ldots, j^1, i)$. This sequence of vertices is a cycle or contains a cycle if both paths share edges, contradicting the fact that G' is a tree.
2. Consider an edge $\{i, j\} \in E \setminus E'$. In graph G', there was a unique path from i to j. The edge $\{i, j\}$ introduces another path. This means the graph $G'' = (N', E' \cup \{i, j\})$ is not a tree (from the claim above). Since G'' is connected, it must contain a cycle.
3. Let $\{i, j\} \in E'$ be the edge removed from G'. By the first claim, there is a unique path from i to j in G'. Since there is an edge between i and j, this unique path is the edge $\{i, j\}$. This means by removing this edge we do not have a path from i to j, and hence the graph is no more connected.

   Now, in the new graph, consider all the vertices N_i to which i is connected and all the vertices N_j to which j is connected - note $i \in N_i$ and $j \in N_j$. Clearly, if we pick $i' \in N_i$ and $j' \in N_j$, then there is no path between i and j. This is because, if there was such a path, then it would define a path without edge $\{i, j\}$ in G' but there is a path involving edge $\{i,j\}$ bet ween i' and j' in G'. This contradicts the fact that there is a unique path between i' and j' in G'. Hence, N_i and N_j along with the corresponding edges among them define two components. It is also clear that we cannot have any more components since each vertex is either connected to i or j.

4. We do this using induction on number of vertices. If there are two vertices, the claim is obvious. Consider a tree with n vertices. Suppose the claim is true for any tree with $< n$ vertices. Now, consider any edge $\{i,j\}$ in the tree. By (1), the unique path between i and j is this edge $\{i,j\}$. Now, remove this edge from the tree. By (3), we disconnect the tree into trees which has smaller number of vertices. Each of these trees have either a single vertex or have two vertices with degree one (by induction). By connecting edge $\{i,j\}$, we can increase the degree of one of the vertices in each of these trees. Hence, there is at least two vertices with degree one in the original graph.

5. For $|N'| = 2$, it is obvious. Suppose the claim holds for every $|N'| = m$. Now, consider a tree with $(m+1)$ vertices. By the previous claim, there is a vertex i that has degree 1. Let the edge for which i is an endpoint be $\{i,j\}$. By removing i, we get the subgraph $(N' \setminus \{i\}, E' \setminus \{i,j\})$, which is a tree. By induction, number of edges of this tree is $m-1$. Since, we have removed one edge from the original tree, the number of edges in the original tree (of a graph with $(m+1)$ vertices) is m.

■

We prove two more important, but straightforward, lemmas.

Lemma **2** *Let $G = (N,E)$ be a graph and $G' = (N,E')$ be a subgraph of G such that $|E'| = |N| - 1$. If G' has no cycles then it is a spanning tree.*

Proof: Consider a cycle-free graph $G' = (N,E')$ and let $G^1,\ldots,G^q$ be the components of G' with number of vertices in component G^j being n_j for $1 \leq j \leq q$. Since every component in a cycle-free graph is a tree, by Proposition 1 the number of edges in component G^j is $n_j - 1$ for $1 \leq j \leq q$. Since the components have no vertices and edges in common, the total number of edges in components $G^1,\ldots,G^q$ is

$$(n_1 - 1) + \ldots + (n_q - 1) = (n_1 + n_2 + \ldots + n_q) - q = |N| - q.$$

By our assumption in the claim, the number of edges in G' is $|N| - 1$. Hence, $q = 1$, i.e., the graph G' is a component, and hence a spanning tree. ■

LEMMA **3** *Let $G = (N, E)$ be a graph and $G' = (N, E')$ be a subgraph of G such that $|E'| = |N| - 1$. If G' is connected, then it is a spanning tree.*

Proof: We show that G' has no cycles, and this will show that G' is a spanning tree. We do the proof by induction on $|N|$. The claim holds for $|N| = 2$ and $|N| = 3$ trivially. Consider $|N| = n > 3$. Suppose the claim holds for all graphs with $|N| < n$. In graph $G' = (N, E')$, there must be a vertex with degree 1. Else, every vertex has degree at least two (it cannot have degree zero since it is connected). In that case, the total degree of all vertices is $2|E| \geq 2n$ or $|E| = n-1 \geq n$, which is a contradiction. Let this vertex be i and let $\{i, j\}$ be the unique edge for which i is an endpoint. Consider the graph $G'' = (N \setminus \{i\}, E' \setminus \{\{i, j\}\})$. Clearly, G'' is connected and number of edges in G'' is one less than $n - 1$. By our induction hypothesis, G'' has no cycles. Hence, G' cannot have any cycle. ■

We can summarize these results as follows.

THEOREM **1** *Let $G = (N, E)$ be a graph and $G' = (N, E')$ be a subgraph of G. Then the following statements are equivalent.*

1. *G' is a spanning tree of G.*
2. *G' has no cycles and connected.*
3. *G' has no cycles and $|E'| = |N| - 1$.*
4. *G' is connected and $|E'| = |N| - 1$.*
5. *There is a unique path between every pair of nodes in G'.*

Note here that if $E' = E$ (i.e., $G' = G$) in the statement of Theorem 1 above, then we get a characterization of a tree.

1.3 The Minimum Cost Spanning Tree Problem

Consider a graph $G = (N, E, w)$, i.e., a weighted graph. Assume G to be connected. Imagine the weights to be costs of traversing an edge. So, $w \in \mathbb{R}_+^{|E|}$. Using these weights, we can compute the cost of a tree (or path or cycle) by summing the weights of edges on the tree (or path or cycle). The **minimum cost spanning tree (MCST) problem** is to find a spanning tree of minimum cost in graph G. Figure 1.4 shows a weighted graph. In this figure, one can imagine one of the vertices as "source" (of water) and other vertices to be cities. The weights on edges may represent cost of supplying water from one city to another.

In that case, the MCST problem is to find a minimum cost arrangement (spanning tree) to supply water to cities.

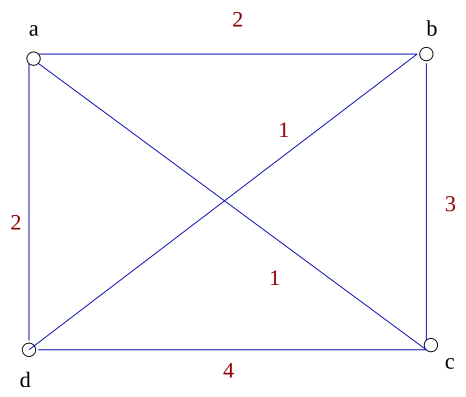


Figure 1.4: The minimum cost spanning tree problem

Of course, one way to find an MCST of a graph is to enumerate all possible spanning trees of a graph and compare their costs. If the original graph itself is a tree, then of course, there is nothing to worry as that graph is the unique spanning tree of itself. But, in general, there may be way too many spanning trees of a graph. Consider a **complete graph** - a graph where for every pair of vertices i and j, there is an edge $\{i, j\}$. It is known that the number of spanning trees in a complete graph with n vertices is n^{n-2} - this is called the **Cayley's formula** [2]. Note that according to Cayley's formula, for 10 vertices the number of spanning trees is about 10^8, a high number. Ideally, we would like an *algorithm* which does not require us to do so many computations. For instance, as the number of vertices grow, the amount of computations should not grow exponentially with the number of vertices.

1.3.1 Greedy Algorithms for MCST

There are many *greedy* algorithms that find an MCST and achieves so in a reasonable amount of computation (and avoiding explicit enumeration of all the spanning trees). We give a generic algorithm, and show one specific algorithm that falls in this generic class.

The generic greedy algorithm grows an MCST one edge at a time. The algorithm manages a subset A of edges that is always a subset of *some* MCST. At each step of the algorithm, an edge $\{i, j\}$ is added to A such that $A \cup \{\{i, j\}\}$ is a subset of *some* MCST. We call such an edge a **safe edge** for A (since it can be safely added to A without destroying the invariant).

[2] Many elegant proofs of Cayley's formula are known. You are encouraged to look at some of the proofs.

In Figure 1.4, A can be taken to be $\{\{b,d\},\{a,d\}\}$, which is a subset of an MCST. A safe edge for A is $\{a,c\}$.

Here is the formal procedure:

1. Set $A = \emptyset$.
2. If $|A| \neq |N| - 1$, then find an edge $\{i,j\}$ that is safe for A. Set $A \leftarrow A \cup \{\{i,j\}\}$.
3. If If $|A| = |N| - 1$, then (N,A) is a spanning tree. Return (N,A) as the MCST. Else, repeat from Step 2.

Remark: After Step 1, the invariant (that in every step we maintain a set of edges which belong to some MCST) is trivially satisfied. Also, if (N,A) is not a spanning tree but a subset of an MCST, then there must exist an edge which is safe.

The question is how to identify safe edges. We discuss one such rule. For this, we provide some definitions. A **cut** in a graph $G = (N,E)$ is a partition of set of vertices $(V, N \setminus V)$ with $V \neq N$ and $V \neq \emptyset$. An edge $\{i,j\}$ **crosses** a cut $(V, N \setminus V)$ if $i \in V$ and $j \in N \setminus V$. We say a cut $(V, N \setminus V)$ **respects** a set of edges A if no edge from A crosses the cut. A **light edge** crossing a cut $(V, N \setminus V)$ is an edge that has the minimum weight among all the edges crossing the cut $(V, N \setminus V)$.

Figure 1.5 shows a graph and two of its cuts. The first cut is $(\{1,2,3\},\{4,5,6\})$. The following set of edges respect this cut $\{\{1,2\},\{1,3\}\}$. Also, the set of edges $\{\{4,5\},\{5,6\}\}$ and the set of edges $\{\{1,3\},\{4,5\}\}$ respect this cut. Edges $\{1,4\},\{2,6\},\{3,4\}$ cross this cut.

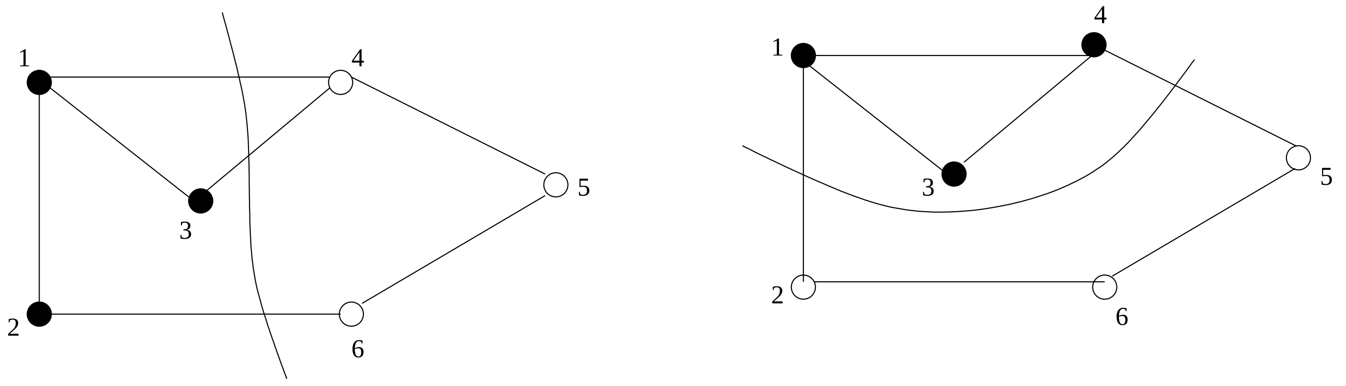


Figure 1.5: Cuts in a graph

The following theorem says how a light edge of an appropriate cut is a safe edge.

THEOREM **2** *Let $G = (N, E, w)$ be a connected graph. Let $A \subset T \subseteq E$ be a subset of edges such that (N, T) is an MCST of G. Let $(V, N \setminus V)$ be any cut of G that respects A and let $\{i, j\}$ be a light edge crossing $(V, N \setminus V)$. Then edge $\{i, j\}$ is a safe edge for A.*

Proof: Let A be a subset of edges of MCST (N, T). If $\{i, j\} \in T$, then we are done. So, we consider the case when $\{i, j\} \notin T$. Since (N, T) is a spanning tree, adding edge $\{i, j\}$ to T creates a cycle (Proposition 1). Hence, the sequence of vertices in the set of edges $T \cup \{\{i, j\}\}$ contains a cycle between vertex i and j - the unique cycle it creates consists of the unique path from i to j in MCST (N, T) and the edge $\{i, j\}$. This cycle must cross the cut $(V, N \setminus V)$ at least twice - once at $\{i, j\}$ and the other at some edge $\{a, b\} \neq \{i, j\}$ such that $a \in V$, $b \in N \setminus V$ which crosses the cut $(V, N \setminus V)$. Note that $\{a, b\} \in T$. If we remove edge $\{a, b\}$, then this cycle is broken and we have no cycle in the graph $G'' = (N, (T \cup \{\{i, j\}\}) \setminus \{\{a, b\}\})$. By Proposition 1, there are $|N| - 1$ edges in (N, T). Hence, G'' also has $|N| - 1$ edges. By Lemma 2, G'' is a spanning tree.

Let $T' = (T \cup \{\{i, j\}\}) \setminus \{\{a, b\}\}$. Now, the difference of edge weights of T and T' is equal to $w(\{a, b\}) - w(\{i, j\})$. Since (N, T) is an MCST, we know that $w(\{a, b\}) - w(\{i, j\}) \leq 0$. Since $\{i, j\}$ is a light edge of cut $(V, N \setminus V)$ and $\{a, b\}$ crosses this cut, we have $w(\{a, b\}) \geq w(\{i, j\})$. Hence, $w(\{a, b\}) = w(\{i, j\})$. Hence (N, T') is an MCST.

This proves that $(A \cup \{\{i, j\}\}) \subseteq T'$. Hence, $\{i, j\}$ is safe for A. ■

The above theorem almost suggests an algorithm to compute an MCST. We now describe this algorithm. This algorithm is called the **Dijkstra-Jarnik-Prim (DJP) algorithm** or just **Prim's** algorithm - named after the authors who formulated the algorithm independently. To describe the algorithm, denote by $V(A)$ the set of vertices which are endpoints of edges in A.

1. Set $A = \emptyset$.

2. Choose any vertex $i \in N$ and consider the cut $(\{i\}, N \setminus \{i\})$. Let $\{i, j\}$ be a light edge of this cut. Then set $A \leftarrow A \cup \{\{i, j\}\}$.

3. If A contains $|N| - 1$ edges then return (N, A) as the MCST and stop. Else, go to Step 4.

4. Consider the cut $(V(A), N \setminus V(A))$. Let $\{i, j\}$ be a light edge of this cut.

5. Set $A \leftarrow A \cup \{\{i, j\}\}$ and repeat from Step 3.

This algorithm produces an MCST. To see this, by Theorem 2, in every step of the algorithm, we add a safe edge, and hence it terminates with an MCST.

We apply this algorithm to the example in Figure 1.4. In the first iteration of the algorithm, we choose vertex a and consider the cut $(\{a\}, \{b, c, d\})$. A light edge of this cut is $\{a, c\}$. So, we set $A = \{\{a, c\}\}$. Then, we consider the cut $(\{a, c\}, \{b, d\})$. A light edge of this cut is $\{a, d\}$. Now, we set $A = \{\{a, c\}, \{a, d\}\}$. Then, we consider the cut $(\{a, c, d\}, \{b\})$. A light edge of this cut is $\{b, d\}$. Since $(N, \{a, c\}, \{a, d\}, \{b, d\})$ is a spanning tree, we stop. The total weight of this spanning tree is $1 + 2 + 1 = 4$, which gives the minimum weight over all spanning trees. Hence, it is an MCST.

1.3.2 Other Algorithms for MCST

We give below, informally, three more algorithms for finding an MCST of a weighted graph $G = (N, E, w)$. We leave the correctness of these algorithms as an exercise. We illustrate the algorithms by the weighted graph in Figure 1.6.

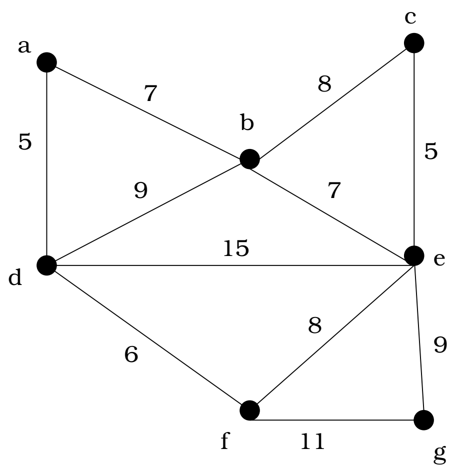
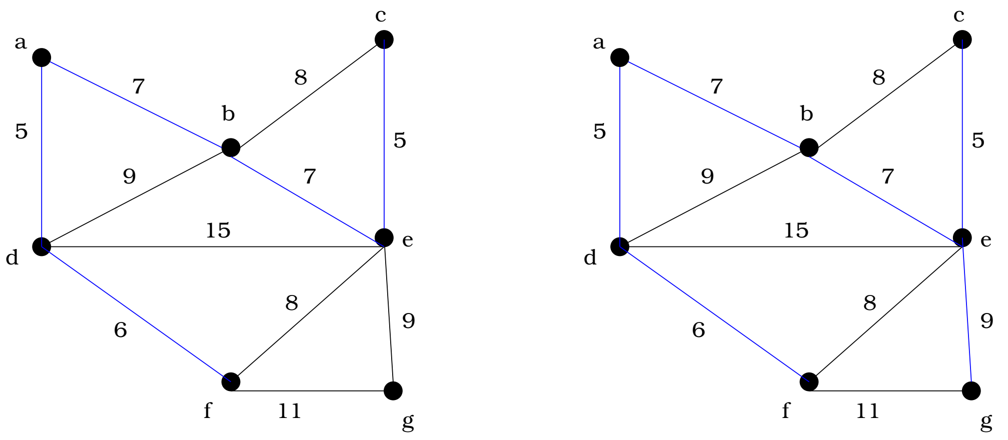


Figure 1.6: Illustration of algorithms to find an MCST

1. **Kruskal's Algorithm.** Kruskal's algorithm maintains two types of sets in each stage t of the algorithm: (1) E^t - the set of added edges till stage t and (2) D^t - the set of discarded edges till stage t. Initially, E^1 consists any one of the smallest weight edges in E and D^1 is empty. In each stage t, one of the cheapest edges $e \in E \setminus (E^{t-1} \cup D^{t-1})$ is chosen. If $E^{t-1} \cup \{e\}$ forms a cycle, then $D^t := D^{t-1} \cup \{e\}$ and $E^t := E^{t-1}$. If $E^{t-1} \cup \{e\}$ does not form a cycle, then $E^t := E^{t-1} \cup \{e\}$ and $D^t := D^{t-1}$. Then, repeat from stage $(t + 1)$. The algorithm terminates at stage t if $|E^t| = |N| - 1$.

Example. Kruskal's algorithm is illustrated in Figures 1.7(a) and 1.7(b) for the example in Figure 1.6. In Figure 1.7(a) all the edges have been added one by one and after this, the next candidate edge to be added is $\{b, c\}$ or $\{e, f\}$, and both of them create cycles. So, they are skipped. Out of the next two candidate edges $\{b, d\}$ and $\{e, g\}$, $\{b, d\}$ forms a cycle, and hence, $\{e, g\}$ is chosen to complete the MCST in Figure 1.7(b).



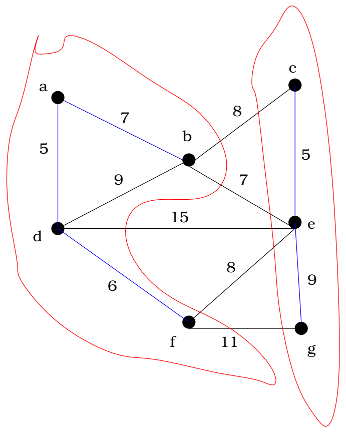
(a) (b)

Figure 1.7: Illustration of Kruskal's algorithm

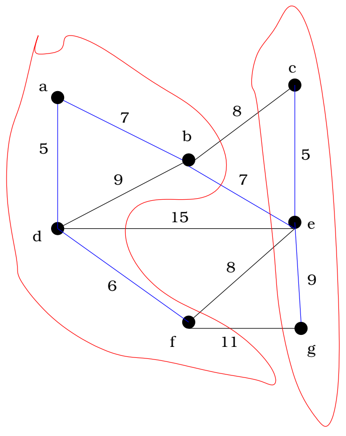
2. **Boruvka's Algorithm.** [3] This algorithm first picks one of the smallest weight edges from every vertex. Note that such a set of edges cannot form a cycle (argue why). If the number of such edges is $|N| - 1$, then the algorithm stops by producing a tree. Else, it must produce several components. Now, a new graph is constructed. Each component is now treated as a new vertex. There might be more than one edge between two components. In that case, one of the smallest weight edges is kept and all the other edges are discarded. Now, one of the smallest weight edges from each vertex is selected, and the step is repeated. The algorithm stops when a tree is chosen in some step. Then, the algorithm backtracks to find the corresponding spanning tree.

   Example. Boruvka's algorithm is illustrated in Figures 1.8(a) and 1.8(b) for the example in Figure 1.6. In Figure 1.8(a), the first set of smallest weight edges for each vertex has been identified, and they form two components. In Figure 1.8(b), these two components are treated like two vertices and the smallest weight edge ($\{b, e\}$) between these two components (vertices) is chosen to complete the MCST.

[3]Interestingly, this algorithm was rediscovered by many, including the famous Choquet of Choquet integral fame.



(a)



(b)

Figure 1.8: Illustration of Boruvka's algorithm

3. **Reverse Delete Algorithm.** In this algorithm one edge at a time is deleted from the original graph G till exactly $|N|-1$ edges remain. First, one of the highest weight edges in E is removed. Next, of the remaining edges, one of the highest weight edges is removed if it does not disconnect the (remaining edges) graph. The process is repeated till we are left with exactly $|N|-1$ edges.

   Example. The reverse delete algorithm is illustrated in Figures 1.9(a) and 1.9(b) for the example in Figure 1.6. Figure 1.9(a) shows successive deletion of high weight edges. After this, the next candidate for deletion is $\{e, g\}$, but this will result in disconnection of vertex g. Hence, this edge is skipped. Then edges $\{b, c\}$ and $\{e, f\}$ are deleted, which results in the tree in Figure 1.9(b). After this, deleting any further edge will lead to disconnection of the graph. Hence, the algorithm stops.

1.4 Application: The Minimum Cost Spanning Tree Game

In this section, we define a cooperative game corresponding to the MCST problem. To do so, we first define the notion of a cooperative game and a well known stability condition for such games.

1.4.1 Cooperative Games

A cooperative game consists of a set of agents and a cost that every subset of agents can incur. Let N be the set of agents. A subset $S \subseteq N$ of agents is called a coalition. Let Ω

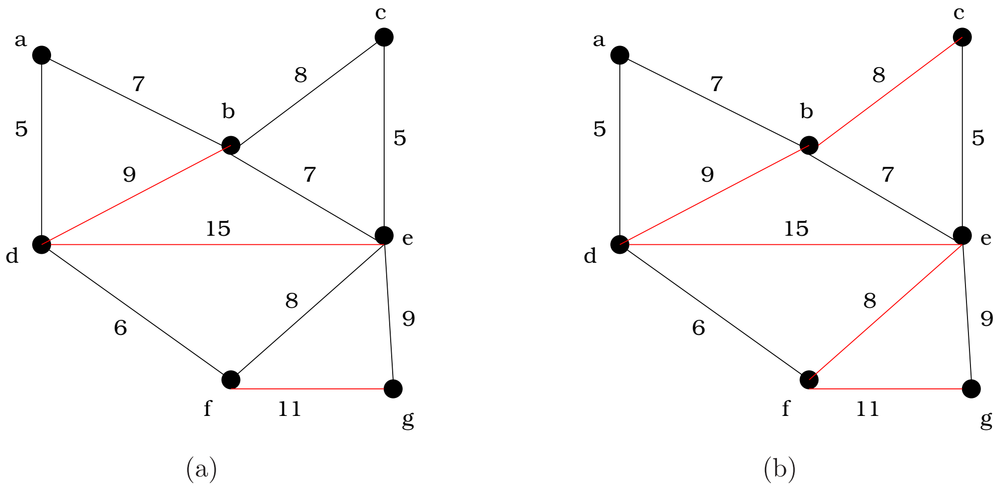


Figure 1.9: Illustration of reverse delete algorithm

be the set of all coalitions. The main objective of a cooperative game is to resolve *conflicts* between coalition to sustain the grand coalition N.

A cooperative game is a tuple (N, c) where N is a finite set of agents and c is a characteristic function defined over the set of coalitions Ω, i.e., $c : \Omega \to \mathbb{R}$. The number $c(S)$ can be thought to be the cost incurred by coalition S when they cooperate [4].

The problem is to divide the total cost $c(N)$ amongst the agents in N when they cooperate. If the division of $c(N)$ is not done properly, then coalition of agents may not want to join the grand coalition. A primary objective of cooperative game theory is to divide the cost of cooperation among the agents such that the grand coalition can be sustained.

A cost vector x assigns every player a cost share in a game (N, c). The core of a cooperative game (N, c) is the set of cost vectors which satisfies a stability condition.

$$Core(N,c) = \{x \in \mathbb{R}^{|N|} : \sum_{i \in N} x_i = c(N), \ \sum_{i \in S} x_i \leq c(S) \ \forall \ S \subsetneq N\}$$

Every cost vector in the core is such that it distributes the total cost $c(N)$ amongst agents in N and no coalition of agents can be better off by forming their independent coalition.

There can be many cost vectors in a core or there may be none. For example, look at the following game with $N = \{1, 2\}$. Let $c(12) = 5$ and $c(1) = c(2) = 2$. Core conditions tell us $x_1 \leq 2$ and $x_2 \leq 2$ but $x_1 + x_2 = 5$. But there are certain class of games which have non-empty core (more on this later). We discuss one such game.

[4]Cooperative games can be defined with value functions also, in which case notations will change, but ideas remain the same.

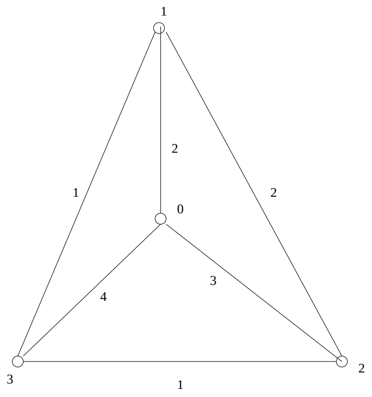


Figure 1.10: An MCST game

1.4.2 The Minimum Cost Spanning Tree Game

The minimum cost spanning tree game (MCST game) is defined by a set of agents $N = \{1, \ldots, n\}$ and a **source** agent 0 to whom all the agents in N need to be connected. The underlying graph is $(N \cup \{0\}, E, c)$ where $E = \{\{i, j\} : i, j \in N \cup \{0\}, i \neq j\}$ and $c(i, j)$ denotes the cost of edge $\{i, j\}$. For any $S \subseteq N$, let $S^+ = S \cup \{0\}$. When a coalition of agents S connect to the source, they form an MCST using edges between themseleves. Let $c(S)$ be the total cost of an MCST when agents in S form an MCST with the source. Thus, (N, c) defines a cooperative game.

An example is given in Figure 1.10. Here $N = \{1, 2, 3\}$ and $c(123) = 4, c(12) = 4, c(13) = 3, c(23) = 4, c(1) = 2, c(2) = 3, c(3) = 4$. It can be verified that $x_1 = 2, x_2 = 1 = x_3$ is in the core. The next theorem shows that this is always the case.

For any MCST (N^+, T), let $\{p(i), i\}$ be the last edge in the unique path from 0 to agent i. Define $x_i = c(p(i), i)$ for all $i \in N$. Call x the Bird cost allocation - named after the inventor of this allocation.

Figure 1.11 gives an example with 5 agents (the edges not shown have very high cost). The MCST is shown with red edges in Figure 1.11. To compute Bird allocation of agent 1, we observe that the last edge in the unique path from 0 to 1 in the MCST is $\{0, 1\}$, which has cost 5. Hence, cost allocation of agent 1 is 5. Consider agent 2 now. The unique path from 0 to 2 in the MCST has edges, $\{0, 3\}$ followed by $\{3, 2\}$. Hence, the last edge in this path is $\{3, 2\}$, which has a cost of 3. Hence, cost allocation of agent 2 is 3. Similarly, we can compute cost allocations of agents 3,4, and 5 as 4, 6, and 3 respectively.

There may be more than one Bird allocation. Figure 1.12 illustrates that. There are two MCSTs in Figure 1.12 - one involving edges $\{0, 1\}$ and $\{1, 2\}$ and the other involving edges $\{0, 2\}$ and $\{1, 2\}$. The Bird allocation corresponding to the first MCST is: agent 1's cost

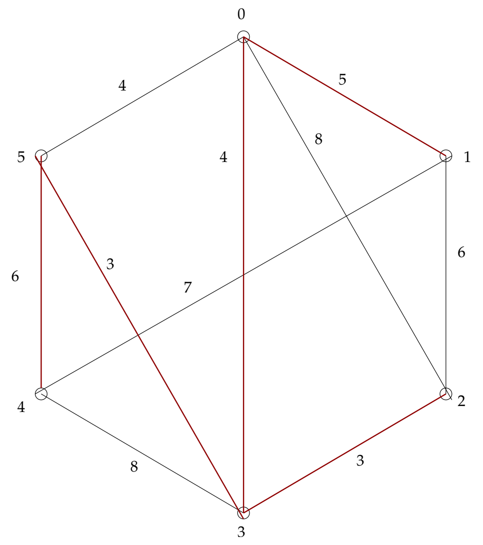


Figure 1.11: Bird allocation

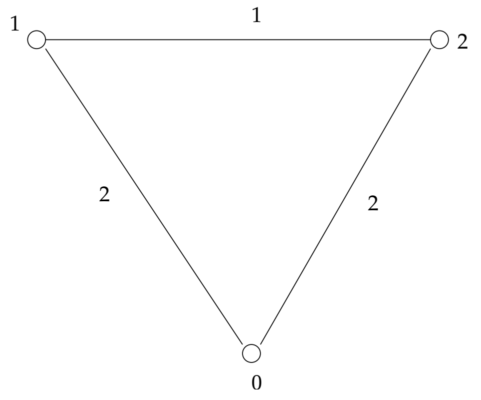


Figure 1.12: More than one Bird allocation

allocation is 2 and that of agent 2 is 1. The Bird allocation corresponding to the second MCST is: agent 2's cost allocation is 2 and that of agent 1 is 1.

Theorem **3** *Any Bird allocation is in the core of the MCST game.*

Proof: For any Bird allocation x, by definition $\sum_{i \in N} x_i = c(N)$. Consider any coalition $S \subsetneq N$. Assume for contradiction $c(S) < \sum_{i \in S} x_i$.

Let (N^+, T) be an MCST for which the Bird allocation x is defined. Delete for every $i \in S$, the edge $e_i = \{p(i), i\}$ (last edge in the unique path from 0 to i) from the MCST (N^+, T). Let this new graph be $(N^+, \hat{T})$. Then, consider the MCST corresponding to nodes S^+ (which only use edges having endpoints in S^+). Such an MCST has $|S|$ edges by Proposition 1. Add the edges of this tree to $(N^+, \hat{T})$. Let this graph be (N^+, T'). Note that (N^+, T')

has the same number ($|N|$) edges as the MCST (N^+, T) - $|N|$ edges since the original graph has $|N|+1$ nodes. We show that (N^+, T') is connected. It is enough to show that there is a path from source 0 to every vertex $i \in N$. We consider two cases.

Case 1: Consider any vertex $i \in S$. We have a path from 0 to i in (N^+, T') by the MCST corresponding S^+.

Case 2: Consider any vertex $i \notin S$. Consider the path in (N^+, T) from 0 to i. Let k be the last vertex in this path such that $k \in S$. So, all the vertices from k to i are not in S in the path from 0 to i. If no such k exists, then the path from 0 to i still exists in (N^+, T'). Else, we know from Case 1, there is a path from 0 to k in (N^+, T'). Take this path and go along the path from k to i in (N^+, T). This defines a path from 0 to i in (N^+, T').

This shows that (N^+, T') is connected and has $|N|$ edges. By Lemma 3, (N^+, T') is a spanning tree.

Now, the new spanning tree has cost $c(N) - \sum_{i \in S} x_i + c(S) < c(N)$ by assumption. This violates the fact that the original tree is an MCST. ■

If the number of Bird allocations is more than one, each of them belongs to the core. Moreover, any convex combination of these Bird allocations is also in the core (this is easy to verify, and left as an exercise). There are members of core which are not necessarily a Bird allocation. Such core allocations have been studied extensively, and have better properties than the Bird allocation.

1.5 Hall's Marriage Theorem

Formally, a graph $G = (N, E)$ is **bipartite** if the set of vertices N can be partitioned into two sets H and B such that for every $\{i, j\} \in E$ we have $i \in H$ and $j \in B$. An equivalent way to state this is that a graph $G = (N, E)$ is bipartite if there is a cut (H, B) of G such that every edge in E crosses this cut. Bipartite graphs possess many interesting properties. Of course, not every graph is bipartite. Figure 1.13(a) shows a graph which is not bipartite, but the graph in Figures 1.13(b) and 1.13(c) are bipartite. For Figure 1.13(b), take $H = \{1, 3, 5\}$ and $B = \{2, 4, 6\}$, and notice that every edge $\{i, j\}$ has $i \in H$ and $j \in B$. For Figure 1.13(c), take $H = \{1, 3, 4\}$ and $B = \{2, 5, 6\}$, and notice that every edge $\{i, j\}$ has $i \in H$ and $j \in B$.

The graph in Figure 1.13(c) is a tree. We show that a tree is always a bipartite graph.

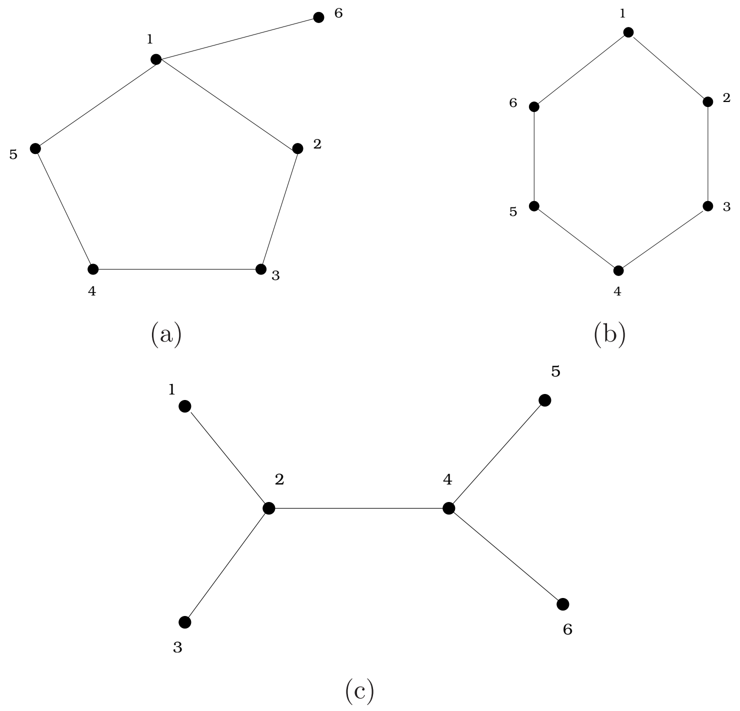


Figure 1.13: Bipartite and non-bipartite graphs

Lemma 4 *A tree is a bipartite graph.*

Proof: We prove the claim by induction on number of vertices. The claim is trivially true for two vertices. Suppose the claim is true for all trees with $n-1$ vertices and consider a tree $G = (N, E)$ with $|N| = n$ vertices. By Proposition 1, there is a vertex $i \in N$, which has degree one. Let $\{i, j\}$ be the unique edge of G with i as one of the end points. Consider the graph $G' = (N \setminus \{i\}, E \setminus \{i, j\})$. By definition, G' is a tree and has $n-1$ vertices. By our induction hypothesis, G' is a bipartite graph. Let the parts of G' be H and B respectively with $H \cup B = N \setminus \{i\}$. Suppose $j \in H$. Then, H and $B \cup \{i\}$ creates a cut of G such that every edge of G crosses this cut. Hence, G is also bipartite. ■

The graph in Figure 1.13(b) is a cycle with six number of edges but the graph in Figure 1.13(a) contains a cycle with five number of edges. As it turns out a graph containing a cycle with odd number of edges cannot be bipartite and, conversely, every cycle in a bipartite graph must have even number of edges - the proofs of these facts is left as an exercise.

Consider an economy with a finite set of houses H and a finite set of buyers B with $|B| \leq |H|$. We want to find if houses and buyers can matched in a compatible manner. For every buyer $i \in B$, a set of houses $\emptyset \neq H_i$ are compatible. Every buyer wants only one house from his compatible set of houses. We say buyer i likes house j if and only if

$j \in H_i$. This can be represented as a bipartite graph with vertex set $H \cup B$ and edge set $E = \{\{i,j\} : i \in B, j \in H_i\}$. A bipartite graph is denoted as $G = (H \cup B, \{H_i\}_{i \in B})$.

We say two edges $\{i,j\}$, $\{i',j'\}$ in a graph are disjoint if i, j, i', j' are all distinct, i.e., the endpoints of the edges are distinct. A set of edges are disjoint if every pair of edges in that set are disjoint. A **matching** in graph G is a set of edges which are disjoint. The number of edges in a matching is called the **size** of the matching. We ask whether there exists a matching with $|B|$ edges, i.e., where all the buyers are matched. Notice that since $|B| \leq |H|$, $|B|$ is the maximum possible size of matching in this graph. In a bipartite graph where $|B| = |H|$, a matching with $|B|$ number of edges is called a **perfect matching**.

Figure 1.14 shows a bipartite graph with a matching: $\{\{1,b\},\{2,a\},\{3,d\},\{4,c\}\}$.

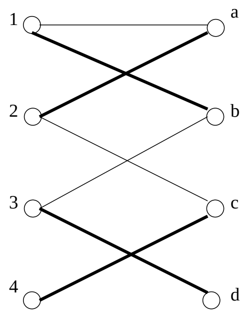


Figure 1.14: A bipartite graph with a matching

We now analyze a bipartite graph $G \equiv (H \cup B, E \equiv \{H_i\}_{i \in B})$. Clearly, a matching where all buyers are matched will not always exist. Consider the case where $|B| \geq 2$ and for every $i \in B$, we have $H_i = \{j\}$ for some $j \in H$, i.e, every buyer likes the same house - j. No matching where all the buyers are matched is possible in this case.

In general, if we take a subset $S \subseteq B$ of buyers and take the set of houses that buyers in S like: $D(S) = \cup_{i \in S} H_i$, then $|S| \leq |D(S)|$ is necessary for a matching where all buyers are matched to exist. Else, number of houses who buyers in S like, are less, and so some buyer cannot be matched. For example, if we pick a set of 5 buyers who like only a set of 3 houses, then we cannot match some buyers.

Hall's marriage theorem states that this condition is also sufficient.

Theorem 4 *A matching with $|B|$ edges in a bipartite graph $G = (H \cup B, \{H_i\}_{i \in B})$ exists if and only if for every $\emptyset \neq S \subseteq B$, we have $|S| \leq |D(S)|$, where $D(S) = \cup_{i \in S} H_i$.*

Before proving Theorem 4, let us consider an alternate interpretation of this theorem. We have already defined the degree of a vertex - it is the number of edges for which it is an endpoint. Now, consider degrees of subsets of vertices. For any graph $G \equiv (N, E)$, the

degree of a subset of vertices $S \subsetneq N$ is the number of *distinct* vertices in $N \setminus S$ with which a vertex in S has an edge. Denote the degree of a subset of vertices S as $deg(S)$.

Now, consider a bipartite graph $G \equiv (H \cup B, \{H_i\}_{i \in B})$. Notice that $deg(S) = D(S)$ for all $\emptyset \neq S \subseteq B$. Then, the condition in Theorem 4 says that for every $\emptyset \neq S \subseteq B$, $deg(S) \geq |S|$.

We now give some examples where the condition of Theorem 4 is violated and satisfied. Figure 1.15(a) shows a bipartite graph which violates the condition of Theorem 4 - here, buyers $\{b, c, d\}$ demand only $\{1, 2\}$. However, the condition of Theorem 4 is satisfied in Figure 1.15(b). A perfect matching in this bipartite graph is $\{a, 3\}, \{b, 4\}, \{c, 2\}, \{d, 1\}$.

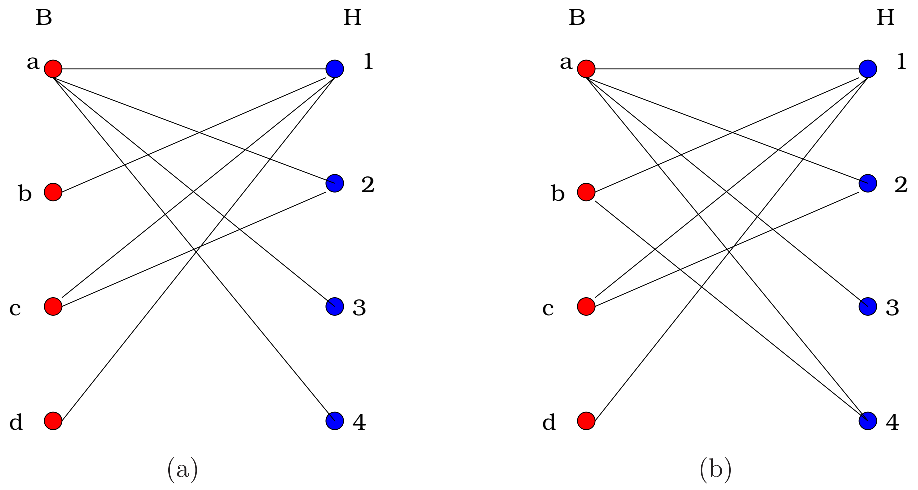


Figure 1.15: Illustration of Hall's Marriage Theorem

Proof: Suppose a matching with $|B|$ edges exists in G. Then, we have $|B|$ disjoint edges. Denote this set of edges as M. By definition, every edge in M has a unique buyer and a unique house as endpoints, and for every $\{i, j\} \in M$ we have $j \in H_i$. Now, for any set of buyers $\emptyset \neq S \subseteq B$, we define $M(S) = \{j \in H : \{i, j\} \in M, i \in S\}$ - the set of houses matched to buyers in S in matching M. We know that $|S| = |M(S)|$. By definition $M(S) \subseteq D(S)$. Hence $|S| \leq |D(S)|$.

Suppose for every $\emptyset \neq S \subseteq B$, we have $|S| \leq |D(S)|$. We use induction to prove that a matching with $|B|$ edges exists. If $|B| = 1$, then we just match her to one of the houses she likes (by our condition she must like at least one house). Suppose a matching with $|B|$ edges exists for any graph with less than $l + 1$ buyers. We will show that a matching with $|B|$ edges exists for any graph with $|B| = l + 1$ buyers.

There are two cases to consider.

Case 1. Suppose $|S| < |D(S)|$ for every $\emptyset \neq S \subsetneq B$ (notice proper subset). Then choose an arbitrary buyer $i \in B$ and any $j \in H_i$, and consider the edge $\{i, j\}$. Now consider the bipartite graph $G' = (H \setminus \{j\} \cup B \setminus \{i\}, \{H_k \setminus \{j\}\}_{k \in B \setminus \{i\}})$. Now, G' is a graph with l buyers. Since we have removed one house and one buyer from G to form G' and since $|S| \leq |D(S)| - 1$ for all $\emptyset \neq S \subsetneq B$, we will satisfy the condition in the theorem for graph G'. By induction assumption, a matching exists in graph G' with $|B| - 1$ edges. This matching along with edge $\{i, j\}$ forms a matching of graph G with $|B|$ edges.

Case 2. For some $\emptyset \neq S \subsetneq B$, we have $|S| = |D(S)|$. By definition $|S| < |B|$, and hence by induction we have a matching in the graph $G' = (S \cup D(S), \{H_i\}_{i \in S})$ with $|S|$ edges. Now consider the graph $G'' = ((H \setminus D(S)) \cup (B \setminus S), \{H_i \setminus D(S)\}_{i \in B \setminus S})$. We will show that the condition in the theorem holds in graph G''.

Consider any $\emptyset \neq T \subseteq (B \setminus S)$. Define $D'(T) = D(T) \setminus D(S)$. We have to show that $|T| \leq |D'(T)|$. We know that $|(T \cup S)| \leq |D(T \cup S)|$. We can write $|D(T \cup S)| = |D(S)| + |(D(T)| \setminus |D(S))| = |D(S)| + |D'(T)|$. Hence, $|(T \cup S)| = |T| + |S| \leq |D(S)| + |D'(T)|$. But $|S| = |D(S)|$. Hence, $|T| \leq |D'(T)|$. Hence, the condition in the theorem holds for graph G''. By definition $|(B \setminus S)| < |B|$. So, we apply the induction assumption to find a matching in G'' with $|(B \setminus S)|$ edges. Clearly, the matchings of G' and G'' do not have common edges, and they can be combined to get a matching of G with $|B|$ edges. ■

Remark: Hall's marriage theorem tells you when a matching can exist in a bipartite graph. It is silent on the problem of finding a matching when it exists. We will study other results about existence and feasibility later.

1.5.1 Application: Competitive Equilibrium with Indivisible Objects

Let M be a set of m indivisible objects and N be a set of n agents. Each agent can consume at most one object. The valuation of agent $i \in N$ for an object $j \in M$ is $v_{ij} \geq 0$. There is a dummy object, denoted by 0, whose valuation for all the agents is zero. Unlike objects in M, the dummy object can be assigned to as many agents as required. The interpretation of the dummy object is that any agent who is not assigned an object in M is assigned the dummy object. Denote $M \cup \{0\}$ as M^0. An allocation is a map $\mu : N \to M^0$, such that for all $i, k \in N$, with $\mu(i), \mu(k) \in M$, we have $\mu(i) \neq \mu(k)$, i.e., agents are assigned unique objects if they are assigned to non-dummy objects.

We allow payments. In particular, the seller announces a price vector $p \in \mathbb{R}^{m+1}_{+}$, where $p_0 = 0$ (if an agent is assigned a dummy object, the interpretation is that he is not assigned, and hence, pays nothing). If an agent i is assigned object j at price vector p, then his net payoff is $v_{ij} - p_j$. We will need the following notion of demand sets. The **demand set** of agent i at price vector p is the set of all objects in M^0 that maximizes his net payoff, i.e., $D_i(p) = \{j \in M^0 : v_{ij} - p_j \geq v_{ik} - p_k \ \forall \ k \in M^0\}$.

We give an example to illustrate ideas. Let $N = \{1, 2, 3\}$ and $M = \{1, 2, 3, 4\}$. The valuations of agents are shown in Table 1.1 - valuation for the dummy object is always zero. Consider a price p as follows: $p(1) = 2, p(2) = 0, p(3) = 3, p_4 = 4$ - by definition, $p(0) = 0$. Then, $D_1(p) = \{1, 2\}$, $D_2(p) = \{2\}$, $D_3(p) = \{2\}$.

	1	2	3	4
$v_{1\cdot}$	4	2	5	3
$v_{2\cdot}$	2	5	3	4
$v_{3\cdot}$	1	4	3	2

Table 1.1: Matching example

Definition 1 *An allocation μ and a price vector p is a* **competitive equilibrium** *if (a) for every $i \in N$, $\mu(i) \in D_i(p)$ and (b) for every $j \in M$, if $\mu(i) \neq j$ for all $i \in N$, then $p_j = 0$. A price vector is called a* **competitive equilibrium price** *vector if there is an allocation μ such that (μ, p) is a competitive equilibrium.*

Condition (a) requires that each agent must be assigned an object that maximizes his net payoff. Condition (b) requires that unassigned objects must have zero price. While condition (a) is simply requiring that demand must be met, condition (b) is requiring that supply should not be more than enough (think of an object with positive price being *supplied*, and then condition (b) is saying that supplied objects must be sold).

In the example in Table 1.1, the price vector $p(1) = 2, p(2) = 0, p(3) = 3, p_4 = 4$, gives $D_1(p) = \{1, 2\}$, $D_2(p) = \{2\}$, $D_3(p) = \{2\}$. Then, it is clear that both conditions for competitive equilibrium cannot be satisfied - objects 3 and 4 are not demanded and cannot be assigned, yet their prices are positive, and objects 1 and 2 are demanded by too many agents. On the other hand, consider the price vector $p(1) = 0, p(2) = 1, p(3) = p(4) = 0$. Here, $D_1(p) = \{3\}, D_2(p) = \{2, 4\}, D_3(p) = \{2, 3\}$. Assigning agent 1 to object 3, agent 2 to object 4, and agent 3 to object 2 makes p a competitive equilibrium price vector.

Suppose we observe only demand sets of agents and the price vector. When can we say that the observed price vector is a competitive equilibrium price vector? One possibility is to check if there is an allocation that can make it a competitive equilibrium. Hall's marriage theorem allows us to verify this using conditions on demand sets only.

To define these conditions, we need some notation. At any price vector p, let $M^+(p)$ be the set of objects with positive price, i.e., $M^+(p) := \{j \in M : p_j > 0\}$. For every subset of objects $S \subseteq M$, define the **demanders** of S at a price vector p as $U(S,p) := \{i \in N : D_i(p) \cap S \neq \emptyset\}$ - these are agents who demand at least one object from S. Similarly, for every subset of objects $S \subseteq M$, define the **exclusive demanders** of S at a price vector p as $O(S,p) := \{i \in N : D_i(p) \subseteq S\}$ - these are agents who demand objects from S only.

Definition **2** *A set of objects $S \subseteq M$ is* **overdemanded** *at price vector p if $|S| < |O(S,p)|$. A set of objects $S \subseteq M^+(p)$ is* **underdemanded** *at price vector p if $|S| > |U(S,p)|$.*

If a set of objects S is overdemanded, then there are too many agents exclusively demanding objects from S, and these exclusive demanders cannot be assigned objects from their demand sets. Similarly, if a set of objects S is underdemanded, then there are too few agents demanding these objects, and these objects cannot be assigned even though they have positive price.

In the example in Table 1.1, the price vector $p(1) = 2, p(2) = 0, p(3) = 3, p_4 = 4$, gives $D_1(p) = \{1,2\}$, $D_2(p) = \{2\}$, $D_3(p) = \{2\}$. Then for objects $\{1,2\}$, agents $\{1,2,3\}$ are exclusive demanders. So, objects $\{1,2\}$ are overdemanded. On the other hand, object 3 and object 4, and the set of objects $\{3,4\}$ and $\{1,3,4\}$ are underdemanded.

So, clearly, at a competitive equilibrium price vector no overdemanded or underdemanded set of objects must exist. We use Hall's marriage theorem to show that these conditions are also sufficient.

Theorem **5** *A price vector is a competitive equilibrium price vector if and only if no set of objects is overdemanded and no set of objects is underdemanded.*

Proof: As argued earlier, if p is a competitive equilibrium price vector, then no set of objects can be overdemanded or underdemanded. For the converse, suppose that p is a price vector where for every $S \subseteq M$, we have $|S| \geq |O(S,p)|$ and for every $S \subseteq M^+(p)$, we have $|S| \leq |U(S,p)|$.

Let $N' := O(M,p)$ be the set of exclusive demanders of M. Since M is not overdemanded, $|N'| \leq |M|$. Now, consider the bipartite graph with set of vertices $N' \cup M$ and edge $\{i,j\}$ exists if and only if $j \in D_i(p)$. For any set of agents $T \subseteq N'$, let $D(T,p)$ be the set of objects demanded by agents in T. Note that agents in T are exclusive demanders of $D(T,p)$. Since $D(T,p)$ is not overdemanded, $|D(T,p)| \geq |T|$. Hence, by Hall's marriage theorem, there is a matching μ^1 of this bipartite graph which matches all the agents in N'. Note that μ^1 may not match all the agents. We can create a matching μ^2 by assigning all the agents outside N^1 to the dummy object and it will satisfy, $\mu^2(i) \in D_i(p)$ for all $i \in N$. This shows that there exists at least one allocation such that for all $i \in N$, the match of i is belongs to

$D_i(p)$. Now, choose an allocation μ among all such allocations that maximizes the number of objects assigned from $M^+(p)$.

Let $S^0 = \{j \in M^+(p) : \mu(i) \neq j \; \forall \; i \in N\}$ - these are unassigned positive price objects in μ. We argue that $S^0 = \emptyset$. Assume for contradiction that S^0 is not empty. Then, let T^0 be the demanders of S^0 at p. Clearly, no agent in T^0 can be assigned an object with zero price (including the dummy object) in μ - if such an agent i exists in T^0, we can assign him to an object in S^0 instead of $\mu(i)$ and this will increase the number of objects being assigned from $M^+(p)$. Now, let S^1 be the set of objects assigned to agents in T^0 in μ. Note that $S^1 \subseteq M^+(p)$ and $S^1 \cap S^0 = \emptyset$. Let T^1 be the set of demanders of S^1 and let S^2 be the set of objects assigned to agents in T^1 in μ. No object in S^2 can have zero price - if it does, we can create a sequence of (object, agent) pairs starting from an object in S^0 and ending at an agent in T^1, which gives rise to a new allocation with more objects in $M^+(p)$ matched than μ, giving us a contradiction (think carefully on how you will build this sequence). Now, let T^2 be the demanders of objects from S^2. We repeat this sequence of sets of objects from $M^+(p)$ and their demanders, $(S^0, T^0, S^1, T^1, \ldots)$, till we discover no new agents. Since the number of objects is finite, this will eventually terminate. Let $(S^0, S^1, \ldots, S^k)$ be the set of objects discovered in this process. Then, the set of agents discovered must be $(T^0, T^1, \ldots, T^{k-1})$. By assumption, agents in T^j ($j \in \{0, 1, \ldots, k-1\}$) are assigned to objects in S^{j+1}. So, $|(T^0 \cup T^1 \cup \ldots \cup T^{k-1})| = |(S^1 \cup S^2 \cup \ldots \cup S^k)|$. But the set of demanders of $S^0 \cup S^1 \cup \ldots \cup S^k$ are $T^0 \cup T^1 \cup \ldots \cup T^{k-1}$. Let $S := S^0 \cup S^1 \cup \ldots \cup S^k$, and by definition $S \subseteq M^+(p)$. Since S is not underdemanded, $|S| \leq |U(S, p)| = |(T^0 \cup T^1 \cup \ldots \cup T^{k-1})|$. This implies that $|S^0| \leq 0$, which is not possible since S^0 is non-empty, a contradiction. ■

1.6 Maximum Matching in Bipartite Graphs

We saw in the last section that matching all buyers in a bipartite matching problem requires a combinatorial condition hold. In this section, we ask the question - what is the maximum number of matchings that is possible in a bipartite graph? We will also discuss an algorithm to compute such a maximum matching.

1.6.1 M-Augmenting Path

We start with the notion of augmenting path in an arbitrary undirected graph. To remind, in a graph $G = (N, E)$, a matching $M \subseteq E$ is a set of disjoint edges in G. Here, one can think of nodes in G to be students, the set of edges to be set of possible pairings of students.

The problem of finding roommates for students can be thought to be a problem of finding a matching (of maximum size) in G. Figures 1.16 and 1.17 show two matchings in a graph - dark edges represent a matching.

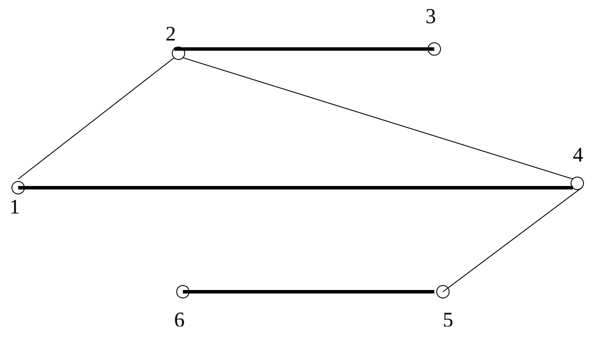


Figure 1.16: A matching in a graph

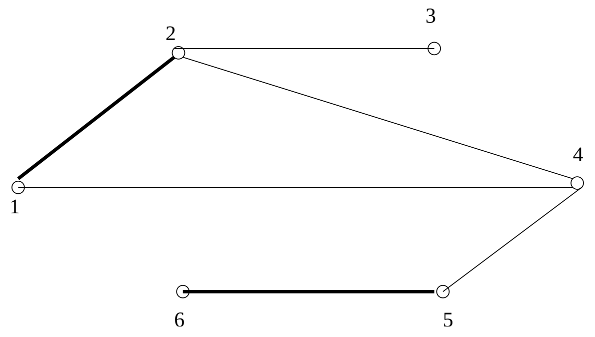


Figure 1.17: A matching in a graph

Before we introduce the definition of an augmenting path, we introduce some terminology. The **length** of a path (cycle) is the number of edges in a path (cycle). Given a graph $G = (N, E)$, a set of vertices $S \subseteq N$ is **covered** by a set of edges $X \subseteq E$ if every vertex in S is an endpoint of some edge in X. If $(i_1, i_2, \ldots, i_k)$ is a path, then i_1 and i_k are called endpoints of this path.

Definition **3** *Let M be a matching in a graph $G = (N, E)$. A path P (with non-zero length) in G is called M-***augmenting** *if its endpoints are not covered by M and its edges are alternatingly in and out of M.*

Note that an M-augmenting path need not contain all the edges in M. Indeed, it may contain just one edge that is not in M. Suppose an M-augmenting path contains k edges from $E \setminus M$. Note that $k \geq 1$ since endpoints of an M-augmenting path are not covered by

M. Then, there are exactly $k-1$ edges from M in this path. If $k=1$, then we have no edges from M in this path. So, an M-augmenting path has odd $(2k-1)$ number of edges, and the number of edges in M is less than the number of edges out of M in an M-augmenting path.

Figures 1.18 and 1.19 show two matchings and their respective M-augmenting paths. The endpoints of the M-augmenting paths are shown in dark, the edges not in the matching are shown with dashed edges. The edges in the matchings are shown in dark edges.

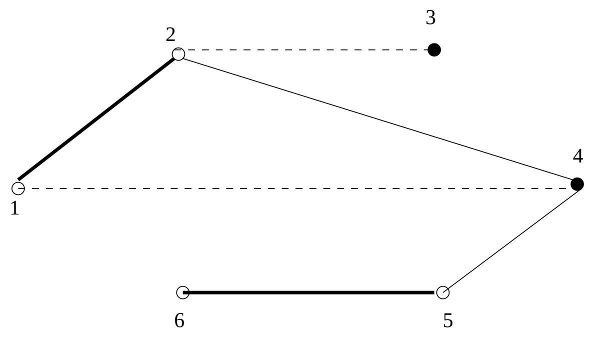


Figure 1.18: An M-augmenting path for a matching

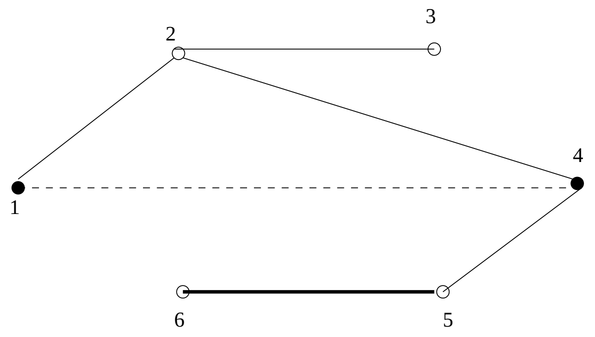


Figure 1.19: An M-augmenting path for a matching

Definition 4 *A matching M in graph $G = (N, E)$ is* **maximum** *if there does not exist another matching M' in G such that $|M'| > |M|$.*

It can be verified that the matching in Figure 1.16 is a maximum matching. There is an obvious connection between maximum matchings and augmenting paths. For example, notice the maximum matching M in Figure 1.16. We cannot seem to find an M-augmenting path for this matching. On the other hand, observe that the matching in Figure 1.17 is not a maximum matching (the matching in Figure 1.16 has more edges), and Figure 1.18 shows an augmenting path of this matching. This observation is formalized in the theorem below.

THEOREM **6** *Let $G = (N, E)$ be a graph and M be a matching in G. The matching M is a maximum matching if and only if there exists no M-augmenting paths.*

Proof: Suppose M is a maximum matching. Assume for contradiction that P is an M-augmenting path. Let E^P be the set of edges in P. Now define, $M' = (E^P \setminus M) \cup (M \setminus E^P)$. By definition of an augmenting path, $E^P \setminus M$ contains more edges than $E^P \cap M$. Hence, M' contains more edges than M. Also, by definition of an augmenting path, the edges in $E^P \setminus M$ are disjoint. Since M is a matching, the set of edges in $(M \setminus E^P)$ are disjoint. Also, by the definition of the augmenting path (ends of an augmenting path are not covered in M), we have that the edges in $(E^P \setminus M)$ and edges in $(M \setminus E^P)$ cannot share an endpoint. Hence, M' is a set of disjoint edges, i.e., a matching with size larger than M. This is a contradiction.

Now, suppose that there exists no M-augmenting path. Assume for contradiction that M is not a maximum matching and there is another matching M' larger than M. Consider the graph $G' = (N, M \cup M')$. Hence, every vertex of graph G' has degree in $\{0, 1, 2\}$. Now, partition G' into components. Each component has to be either an isolated vertex or a path or a cycle. Note that every cycle must contain equal number of edges from M and M'. Since the number of edges in M' is larger than that in M, there must exist a component of G' which is a path and which contains more edges from M' than from M. Such a path forms an M-augmenting path. This is a contradiction. ■

Theorem 6 suggests a simple algorithm for finding a maximum matching. The algorithm starts from some arbitrary matching, may be the empty one. Then, it searches for an augmenting path of this matching. If there is none, then we have found a maximum matching, else the augmenting path gives us a matching larger than the current matching, and we repeat. Hence, as long as we can find an augmenting path for a matching, we can find a maximum matching.

1.6.2 Algorithm for Maximum Matching in Bipartite Graphs

We describe a simple algorithm to find a maximum matching in bipartite graphs. We have already laid the foundation for such an algorithm earlier in Theorem 6, where we proved that any matching is either a maximum matching or there exists an augmenting path of that matching which gives a larger matching than the existing one. We use this fact.

The algorithm starts from an arbitrary matching and searches for an augmenting path of that matching. Let M be any matching of bipartite graph $G = (N, E)$ and $N = B \cup L$ such that for every $\{i, j\} \in E$ we have $i \in B$ and $j \in L$. Given the matching M, we construct a directed graph G^M from G as follows:

- The set of vertices of G^M is N.
- For every $\{i,j\} \in M$ with $i \in B$ and $j \in L$, we create the edge (i,j) in graph G^M, i.e., edge from i to j.
- For every $\{i,j\} \notin M$ with $i \in B$ and $j \in L$, we create the edge (j,i) in graph G^M, i.e., edge from j to i.

Consider the bipartite graph in Figure 1.20 (left one) and the matching M shown with dark edges. For the matching M, the corresponding directed graph G^M is shown on the right in Figure 1.20.

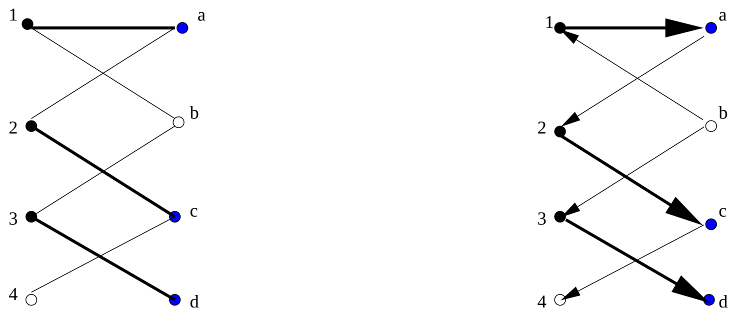


Figure 1.20: A bipartite graph with a matching

Let B^M be the set of vertices in B *not* covered by M and L^M be the set of vertices in L *not* covered by M. Note that every vertex in B^M has no outgoing edge and every vertex in L^M has no incoming edge.

We first prove a useful lemma. For every directed path in G^M the corresponding path in G is the path obtained by removing the directions of the edges in the path of G^M.

LEMMA **5** *A path in G is an M-augmenting path if and only if it is the corresponding path of a directed path in G^M which starts from a vertex in L^M and ends at a vertex in B^M.*

Proof: Consider a directed path P in G^M which starts from a vertex in L^M and ends at vertex in B^M. By definition, the endpoints of P are not covered by M. Since edges from L to B are not in M and edges from B to L are in M in G^M, alternating edges in P is in and out of M. Hence, the corresponding path in G is an M-augmenting path.

For the converse, consider an M-augmenting path in G and let P be this path in G^M with edges appropriately oriented. Note that endpoints of P are not covered by M. Hence, the starting point of P is in L^M and the end point of P is in B^M - if the starting point belonged to B^M, then there will be no outgoing edge and if the end point belonged to L^M, then there will be no incoming edge. This shows that P is a directed path in G^M which starts from a vertex in L^M and ends at a vertex in B^M. ■

Hence, to find an augmenting path of a matching, we need to find a specific type of path in the corresponding directed graph. Consider the matching M shown in Figure 1.20 and the directed graph G^M. There is only one vertex in L^M - $\{b\}$. The directed path $(b, 1, a, 2, c, 4)$ is a path in G^M which starts at L^M and ends at B^M (see Figure 1.21). The corresponding path in G gives an M-augmenting path. The new matching from this augmenting path assigns: $\{1, b\}, \{2, a\}, \{4, c\}, \{3, d\}$ (see Figure 1.21). It is now easy to see that this is indeed a maximum matching (if it was not, then we would have continued in the algorithm to find an augmenting path of this matching).

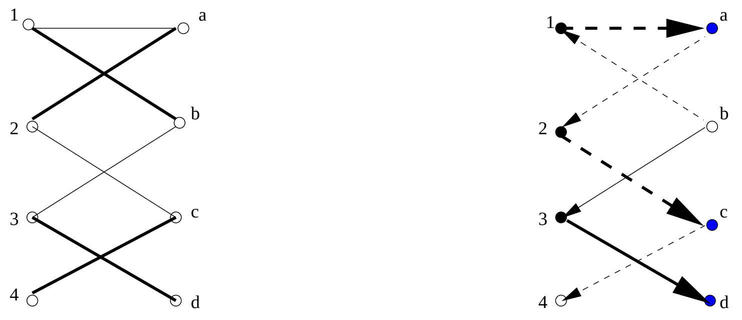


Figure 1.21: A bipartite graph with a matching

1.6.3 Minimum Vertex Cover and Maximum Matching

The **size** of a maximum matching in a graph is the number of edges in the maximum matching. We define vertex cover now and show its relation to matching. In particular, we show that the minimum vertex cover and the maximum matching of a bipartite graph have the same size.

Definition 5 *Given a graph $G = (N, E)$, a set of vertices $C \subseteq N$ is called a* **vertex cover** *of G if every edge in E has at least one end point in C. Further, C is called a* **minimum vertex cover** *of G if there does not exist another vertex cover C' of G such that $|C'| < |C|$.*

Clearly, the set of all vertices in a graph consists of a vertex cover. But this may not be a minimum vertex cover. We give some examples in Figure 1.22. Figure 1.22 shows two vertex covers of the same graph - vertex covers are shown with black vertices. The first one is not a minimum vertex cover but the second one is.

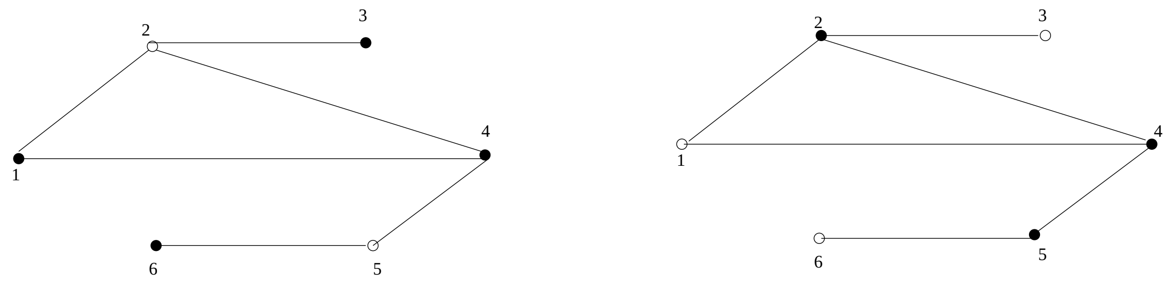


Figure 1.22: Vertex cover

An application of the vertex cover can be as follows. Suppose the graph represents a city: the vertices are squares and the edges represent streets. The city plans to deploy security office (or medical store or emergency service or park) at squares to monitor streets. A security officer deployed at a square can monitor all streets which have an endpoint in that square. The minimum vertex cover problem finds the minimum set of squares where one needs to put a security officer to monitor all the streets.

Fix a graph G. Denote the size of maximum matching in G as $\mu(G)$ - this is also called the **matching number** of G. Denote the size of minimum cover in G as $\kappa(G)$ - this is also called the **vertex cover number** of G.

LEMMA **6** *For any graph G, $\mu(G) \leq \kappa(G)$.*

Proof: Any vertex cover contains at least one end point of every edge of a matching. Hence, consider the maximum matching. A vertex cover will contain at least one vertex from every edge of this matching - this follows from the fact that the edges of a matching are disjoint. This implies that for every graph G, $\mu(G) \leq \kappa(G)$. ■

Lemma 6 can hold with strict inequality in general graphs. Consider the graph in Figure 1.23. A minimum vertex cover, as shown with black vertices, has two vertices. A maximum matching, as shown with the dashed edge, has one edge.

But the relationship in Lemma 6 is equality in case of bipartite graphs as the following theorem, due to Koňig shows.

THEOREM **7 (Koňig's Theorem)** *Suppose $G = (N, E)$ is a bipartite graph. Then, $\mu(G) = \kappa(G)$.*

Figure 1.24 shows a bipartite graph and its maximum matching edges (in dark) and minimum vertex cover (in dark). For the bipartite graph in Figure 1.24 the matching number (and the vertex cover number) is two.

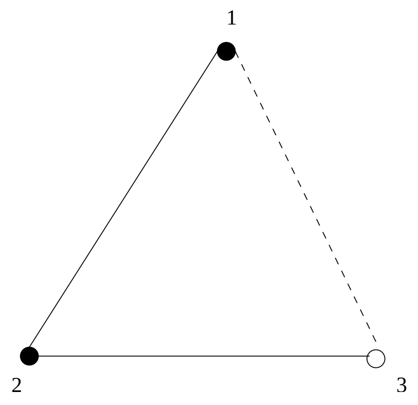


Figure 1.23: Matching number and vertex cover number

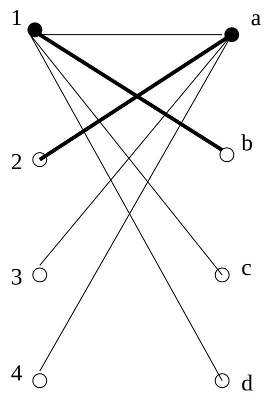


Figure 1.24: Matching number and vertex cover number in bipartite graphs

We will require the following useful result for proving Theorem 7. A graph may have multiple maximum matchings. The following result says that there is at least one vertex which is covered by every maximum matching if the graph is bipartite.

Lemma **7** *Suppose $G = (N, E)$ is a bipartite graph with $E \neq \emptyset$. Then, there exists a vertex in G which is covered by every maximum matching.*

Proof: Assume for contradiction that every vertex is not covered by some maximum matching. Consider any edge $\{i, j\} \in E$ - if the graph contains no edges, then there is nothing to prove. Suppose i is not covered by maximum matching M and j is not covered by maximum matching M'. Note that j must be covered by M - else adding $\{i, j\}$ to M gives another matching which is larger in size than M. Similarly, i must be covered by M'. Note that the edge $\{i, j\}$ is not in $(M \cup M')$.

Consider the graph $G' = (N, M \cup M')$. A component of G' must contain i. Since M' covers i, such a component will have alternating edges in and out of M and M'. Since i

is covered by M' and not by M, i must be an end point in this component. Further, this component must be a path - denote this path by P. Note that P contains alternating edges from M and M' (not in M). The other endpoint of P must be a vertex k which is covered by M - else, P defines an M-augmenting path, contradicting that M is a maximum matching by Theorem 6. This also implies that k is not covered by M' and P has even number of edges.

We argue that P does not contain j. Suppose P contains j. Since j is covered by M and not by M', j must be an endpoint of P. Since G is bipartite, let $N = B \cup L$ and every edge $\{u, v\} \in E$ is such that $u \in B$ and $v \in L$. Suppose $i \in B$. Since the number of edges in P is even, both the end points of P must be in B. This implies that $j \in B$. This contradicts the fact that $\{i, j\}$ is an edge in G.

So, we conclude that j is not in P. Consider the path P' formed by adding edge $\{i, j\}$ to P. This means j is an end point of P'. Note that j is not covered by M' and the other endpoint k of P' is also not covered by M'. We have alternating edges in and out of M' in P'. Hence, P' defines an M'-augmenting path. This is a contradiction by Theorem 6 since M' is a maximum matching. ■

Proof of Theorem 7

Proof: We use induction on number of vertices in G. The theorem is clearly true if G has one or two vertices. Suppose the theorem holds for any bipartite graph with less than n vertices. Let $G = (N, E)$ be a bipartite graph with n vertices. By Lemma 7, there must exist a vertex $i \in N$ such that every maximum matching of G must cover i. Let E^i be the set of edges in G for which i is an endpoint. Consider the graph $G' = (N \setminus \{i\}, E \setminus E^i)$. Note that G' is bipartite and contains one less vertex. Hence, $\mu(G') = \kappa(G')$.

We show that $\mu(G') = \mu(G) - 1$. By deleting the edge covering i in any maximum matching of G, we get a matching of G'. Hence, $\mu(G') \geq \mu(G) - 1$. Suppose $\mu(G') > \mu(G) - 1$. This means, $\mu(G') \geq \mu(G)$. Hence, there exists a maximum matching of G', which is also a matching of G, and has as many edges as the maximum matching matching of G. Such a maximum matching of G' must be a maximum matching of G as well and cannot cover i since i is not in G'. This is a contradiction since i is a vertex covered by every maximum matching of G.

This shows that $\mu(G') = \kappa(G') = \mu(G) - 1$. Consider the minimum vertex cover C of G' and add i to C. Clearly $C \cup \{i\}$ is a vertex cover of G and has $\kappa(G') + 1$ vertices. Hence, the minimum vertex cover of G must have no more than $\kappa(G') + 1 = \mu(G)$ vertices. This means $\kappa(G) \leq \mu(G)$. But we know from Lemma 6 that $\kappa(G) \geq \mu(G)$. Hence, $\kappa(G) = \mu(G)$. ■

Now, we show how to compute a minimum vertex cover from a maximum matching in a bipartite graph. The computation uses the ideas we used to find M-augmenting paths of bipartite graphs. For any bipartite graph $G = (B \cup L, E)$ and a maximum matching M, we first construct the directed bipartite graph G^M as follows: (a) the set of nodes of G^M is $B \cup L$ and (b) there is an edge from $i \in B$ to $j \in L$ if $\{i, j\} \in M$ and (c) there is an edge from $i \in L$ to $j \in B$ if $\{i, j\} \in E \setminus M$.

Consider the bipartite graph G in Figure 1.24. Figure 1.25 shows its maximum matching (in dark edges) M and the directed graph G^M.

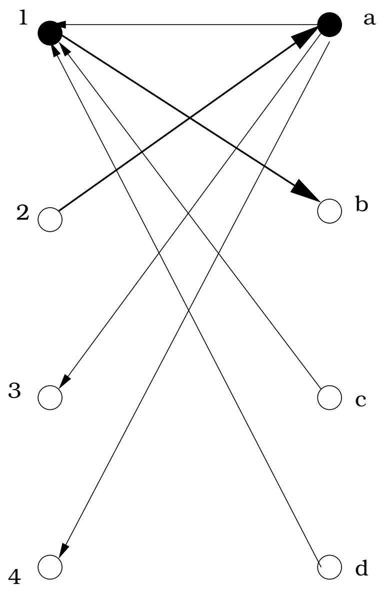


Figure 1.25: Minimum vertex cover from maximum matching

Let L^M be the set of vertices in L **not** covered by the maximum matching M and B^M be the set of vertices in B **not** covered by the maximum matching M. Formally,

$$L^M := \{i \in L : \{i, j\} \notin M \text{ for all } j \in B\}$$
$$B^M := \{i \in B : \{i, j\} \notin M \text{ for all } j \in L\}.$$

We first consider the set of vertices *reachable* from the set of vertices in L^M in G^M - a vertex $j \in G^M$ is reachable from a vertex $i \in L^M$ in the directed graph G^M if there is a directed path from i to j or $j \in L^M$. Call such a set of vertices R^M. Formally,

$$R^M := \{j \in B \cup L : \text{ there exists a } i \in L^M \text{ such that there is a path from } i \text{ to } j \text{ in } G^M\} \cup L^M.$$

In Figure 1.25, we have $L^M = \{c, d\}$ and $R^M = \{c, d, 1, b\}$. Note that $L^M \subseteq R^M$ by definition. In Figure 1.25, we have $1 \in R^M$ since $(c, 1) \in G^M$ and $c \in L^M$ and $b \in R^M$ since $c \in L^M$ and there is a path $(c, 1, b)$.

We now define a set of vertices C^M of G for the matching M as follows.

$$C^M := (L \setminus R^M) \cup (B \cap R^M).$$

In Figure 1.25, we have $C^M := \{1, a\}$, which is a minimum vertex cover. We show that this is true in general.

THEOREM 8 *Given a maximum matching M, the set of vertices C^M defines a minimum vertex cover of G.*

Proof: We do the proof in several steps.

STEP 1. We show that C^M defines a vertex cover of G. To see this, take any edge $\{i, j\}$ where $i \in B$ and $j \in L$. If $j \in L \setminus R^M$, then $j \in C^M$, and $\{i, j\}$ is covered. If $j \in R^M$, then there are two cases: (1) $\{i, j\} \in M$, in that case, the directed path to j must come from i, and hence $i \in R^M$; (2) $\{i, j\} \notin M$, then there is an edge from j to i in G^M, and hence $i \in R^M$. So, $j \in R^M$ implies $i \in B \cap R^M$, and hence $i \in C^M$.

STEP 2. We argue that $R^M \cap B^M = \emptyset$ - because if some vertex in $i \in R^M$ belongs to B^M, then it must be the last vertex in the path which starts from a vertex in L^M, and this will define an M-augmenting path, a contradiction since M is a maximum matching.

STEP 3. We show that C^M is disjoint from $B^M \cup L^M$. By definition, $L^M \subseteq R^M$. Hence, C^M is disjoint from L^M. By Step 2, C^M is disjoint from B^M. Hence, it is disjoint from $B^M \cup L^M$.

STEP 4. We conclude by showing that C^M is a minimum vertex cover. First, for every edge $\{i, j\} \in M$, either $i \notin C^M$ or $j \notin C^M$. To see this, suppose $i, j \in C^M$ with $i \in B$ and $j \in L$. But this means $i \in R^M$, which means $j \in R^M$, which contradicts the fact that $j \in C^M$. By Step 3, it must be that $|C^M| \leq |M|$. By Step 1, and Koňig's theorem (Theorem 7), C^M is a minimum vertex cover. ■

The problem of finding a maximum matching in general graphs is more involved and skipped. The analogue of Hall's marriage theorem in general graphs in called the **Tutte's theorem**. The size of the maximum matching in a general graph has an explicit formula, called the **Tutte-Berge formula**. These are topics that I encourage you to study on your own.

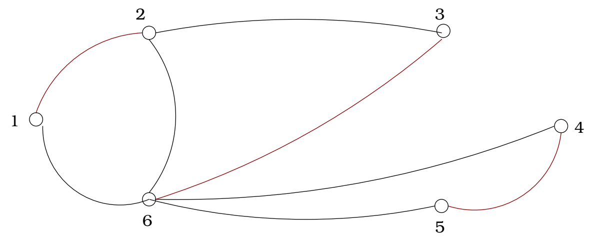


Figure 1.26: A minimum edge cover

1.6.4 Edge Covering

The problem of edge covering is analogous to the vertex cover problem.

DEFINITION **6** *Given a graph $G = (N, E)$, an* **edge cover** *of G is a subset of edges $S \subseteq E$ such that for every $i \in N$, there exists an edge in S with i as the endpoint.*

Clearly, if an edge cover exists, then no vertex can have degree zero. Hence, we will only consider graphs with vertices having positive degree. A **minimum edge cover** is an edge cover with the smallest possible number of edges.

The following example in Figure 1.26 shows a graph with a minimum edge cover - edges $\{1, 2\}, \{3, 6\}, \{4, 5\}$ define a minimum edge cover. Note that this also defines a maximum matching. Further, the sum of sizes of minimum edge cover and maximum matching is 6 (no. of nodes in this graph). This is true for every graph. For any graph G, we will denote the number of edges in a minimum edge cover as $\rho(G)$. To remind, $\mu(G)$ denotes the number of edges in a maximum matching of G.

THEOREM **9** *For any graph G with no vertex of degree zero, we have $\mu(G) + \rho(G) = n$.*

Proof: Consider any maximum matching M. Let V be the set of vertices covered by M, i.e., V is the set of endpoints of edges in M. Note that $|V| = 2|M|$ since M is a matching. First note that for any $i, j \in N \setminus V$, we have $\{i, j\} \notin E$. This is true since if $\{i, j\} \in E$ and $i, j \notin V$, $M \cup \{\{i, j\}\}$ will define a new matching, contradicting maximality of M.

Now, consider an edge cover S' by considering all the edges in M and taking one edge for every vertex in $N \setminus V$ - this is possible since each vertex has non-zero degree (also, note that each such edge must cover a unique vertex in $N \setminus V$ because of the fact we have shown in the first paragraph). The number of edges in S' is thus

$$|M| + |N \setminus V| = |M| + n - |V| = n - |M| = n - \mu(G).$$

Note that $|S'| \geq \rho(G)$. Hence,

$$\rho(G) + \mu(G) \leq n.$$

We now start by considering a minimum edge cover S of G. Notice that the subgraph (N, S) will not have any path with more than two edges - because if a path with more than two edges existed, then some edges in the middle can be deleted to get a smaller edge cover. Thus, (N, S) will not have any cycles. Hence, (N, S) can be broken down into components with each component being a tree and has a maximum path length of two - such graphs are called *star graphs*. Suppose we have k such components, and denote the components as $1, \ldots, k$. Suppose component j has e_j number of edges and $e_j + 1$ number of nodes. Then, $\sum_{j=1}^{k} e_j + k = n$. But $\sum_{j=1}^{k} e_j = \rho(G)$. Hence, $k = n - \rho(G)$. If we take one edge from each component, it will define a matching. Hence, the maximum matching must have at least $n - \rho(G)$ edges. So, we have $\mu(G) \geq n - \rho(G)$ or $\mu(G) + \rho(G) \geq n$.

These two arguments show that $\mu(G) + \rho(G) = n$. ■

An immediate corollary using Konig's theorem and Theorem 9 is the following.

Corollary **1** *For a bipartite graph G with no vertex of degree zero, $\kappa(G) + \rho(G) = n$.*

1.6.5 Independent Set

We now define a new notion for undirected graphs and show its relation to minimum vertex cover problem.

Definition **7** *Given a graph $G = (N, E)$, a subset of vertices $V \subseteq N$ is called an* **independent set** *of G if for every $i, j \in V$, we have $\{i, j\} \notin E$.*

A subset of vertices V is called a **maximum independent set** of G if it is an independent set of G and every other independent set V' of G satisfies $|V'| \leq |V|$.

The following example in Figure 1.27 shows a graph with a maximum independent set - nodes $\{1, 3, 4\}$ is a maximum independent set and nodes $\{2, 5, 6\}$ is a minimum vertex cover. Note that the sum of sizes of minimum vertex cover and maximum independent set is 6 (no. of nodes in this graph). This is true for every graph.

The size of the maximum independent set of G will be denoted by $\delta(G)$.

Theorem **10** *For any graph $G = (N, E)$, $\delta(G) + \kappa(G) = n$.*

Proof: Suppose V is a maximum independent set of G. By considering all the vertices in $N \setminus V$, we get a vertex cover of G. This follows from the definition of an independent set - there is no edge with both endpoints in V. As a result, $\kappa(G) \leq |N \setminus V| = n - |V| = n - \delta(G)$. Hence, $\kappa(G) + \delta(G) \leq n$.

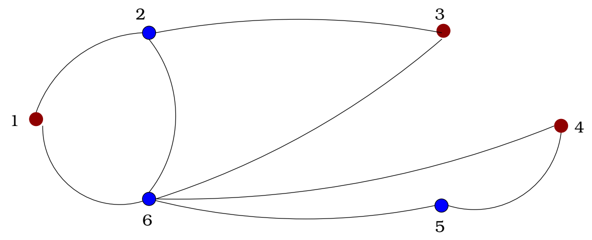


Figure 1.27: A maximum independent set

Suppose V' is a minimum vertex cover of G. The set $N \setminus V'$ must define an independent set of G. Otherwise, there is an edge $\{i, j\}$ with $i, j \in N \setminus V'$, and since V' is a vertex cover, we get a contradiction. Hence, $\delta(G) \geq |N \setminus V'| = n - \kappa(G)$. Hence, $\delta(G) + \kappa(G) \geq n$. This gives us $\kappa(G) + \delta(G) = n$. ■

Theorems 9, 10, and Konig's theorem lead us to the following theorem for bipartitie graphs.

Theorem **11 (Gallai's Theorem)** *Suppose $G = (N, E)$ is any graph with no vertex of degree zero. Then,*

$$\delta(G) + \kappa(G) = \mu(G) + \rho(G) = n.$$

If G is a bipartite graph, then

$$\kappa(G) + \rho(G) = \delta(G) + \mu(G) = n, \quad \kappa(G) = \mu(G), \quad \rho(G) = \delta(G).$$

1.7 Basic Directed Graph Definitions

A **directed graph** is defined by a triple $G = (N, E, w)$, where $N = \{1, \ldots, n\}$ is the set of n nodes, $E \subseteq \{(i, j) : i, j \in N\}$ is the set of edges (ordered pairs of nodes), and w is a vector of weights on edges with $w(i, j) \in \mathbb{R}$ denoting the weight or length of edge $(i, j) \in E$. Notice that the length of an edge is not restricted to be non-negative. A **complete** graph is a graph in which there is an edge between every pair of nodes.

A **path** is a sequence of *distinct* nodes $(i^1, \ldots, i^k)$ such that $(i^j, i^{j+1}) \in E$ for all $1 \leq j \leq k-1$. If $(i^1, \ldots, i^k)$ is a path, then we say that it is a path from i^1 to i^k. A graph is **strongly connected** if there is a path from every node $i \in N$ to every other node $j \in N \setminus \{i\}$.

A **cycle** is a sequence of nodes $(i^1, \ldots, i^k, i^{k+1})$ such that $(i^1, \ldots, i^k)$ is a path, $(i^k, i^{k+1}) \in E$, and $i^1 = i^{k+1}$. The length of a path or a cycle $P = (i^1, \ldots, i^k, i^{k+1})$ is the sum of the edge lengths in the path or cycle, and is denoted as $l(P) = w(i^1, i^2) + \ldots + w(i^k, i^{k+1})$. Suppose there is at least one path from node i to node j. Then, the **shortest path** from node i

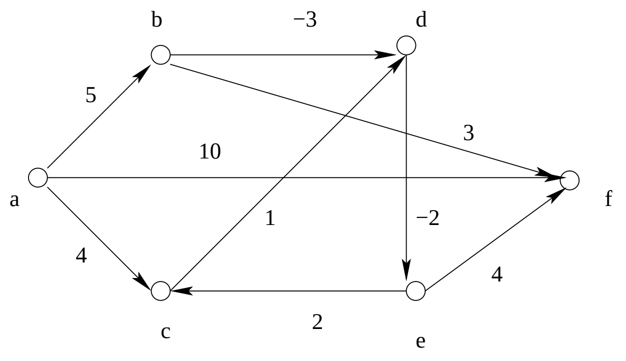


Figure 1.28: A Directed Graph

to node j is a path from i to j having the minimum length over all paths from i to j. We denote the length of the shortest path from i to j as $s(i, j)$.

Figure 1.28 shows a directed graph. A path from a to f is (a, b, d, e, f). A cycle in the graph is (c, d, e, c). The length of the path (a, b, d, e, f) is $5 + (-3) + (-2) + 4 = 4$. The length of the cycle (c, d, e, c) is $1 + (-2) + 2 = 1$. The possible paths from a to f with their corresponding lengths are:

- (a, f): 10.
- (a, b, f): $5 + 3 = 8$.
- (a, b, d, e, f): $5 + (-3) + (-2) + 4 = 4$.
- (a, c, d, e, f): $4 + 1 + (-2) + 4 = 7$.

Hence, $s(a, f) = 4$, and the shortest path from a to f is (a, b, d, e, f).

Here is a useful lemma.

LEMMA 8 *Suppose $(i, i^1, \ldots, i^k, j)$ is the shortest path from i to j in the digraph $G = (N, E, w)$. If G has no cycles of negative length, then for every $i^p \in \{i^1, \ldots, i^k\}$, $s(i, i^p) = l(i, i^1, \ldots, i^p)$ and $s(i^p, j) = l((i^p, \ldots, i^k, j)$.*

Proof: Let $P_1 = (i, i^1, \ldots, i^p)$ and $P_2 = (i^p, \ldots, i^k, j)$. By the definition of the shortest path, $s(i, i^p) \leq l(P_1)$. Assume for contradiction, $s(i, i^p) < l(P_1)$. This implies that there is some other path P_3 from i to i^p which is shorter than P_1. If P_3 does not involve any vertex from P_2, then P_3 and P_2 define a path from i to j, and $l(P_3) + l(P_2) < l(P_1) + l(P_2) = s(i, j)$. But this contradicts the fact that $(i, i^1, \ldots, i^k, j)$ is a shortest path. If P_3 involves a vertex from P_2, then P_3 and P_2 will include cycles, which have non-negative length. Removing these cycles from P_2 and P_3 will define a new path from i to j which has shorter length. This is again a contradiction.

A similar argument works if $s(i^p, j) < l(P_2)$. ■

1.7.1 Potentials

Definition **8** *A* **potential** *of a directed graph G is a function $p : N \to \mathbb{R}$ such that $p(j) - p(i) \leq w(i,j)$ for all $(i,j) \in E$.*

Figure 1.29 illustrates the idea of potentials. The idea of potentials is borrowed from Physics. Potential of a node can be interpreted as the *potential energy* at the node. The weight of an edge represents the gain in energy by going along the edge. The potential inequality is a feasible energy transfer constraint.

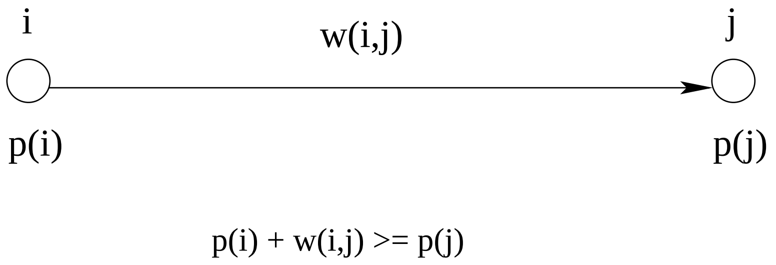


Figure 1.29: Idea of potentials

Notice that if p is a potential of graph G, so is $\{p(j) + \alpha\}_{j \in N}$ for all $\alpha \in \mathbb{R}$. Potentials do not always exist. Consider a directed graph with two nodes $\{1, 2\}$ and edge lengths $w(1, 2) = 3$ and $w(2, 1) = -4$. For this graph to have a potential, the following system of inequalities must have a solution:

$$p(1) - p(2) \leq w(2,1) = -4$$
$$p(2) - p(1) \leq w(1,2) = 3.$$

But this is not possible since adding them gives zero on the LHS and a negative number on the RHS.

The following theorem provides a necessary and sufficient condition for a directed graph to have a potential.

Theorem **12** *There exists a potential of directed graph $G = (N, E, w)$ if and only if G has no cycles of negative length.*

Proof: Suppose a potential p exists of graph G. Consider a cycle $(i^1, \ldots, i^k, i^1)$. By definition of a cycle, $(i^j, i^{j+1}) \in E$ for all $1 \leq j \leq k-1$ and $(i^k, i^1) \in E$. Hence, we can write

$$\begin{aligned} p(i^2) - p(i^1) &\leq w(i^1, i^2) \\ p(i^3) - p(i^2) &\leq w(i^2, i^3) \\ \ldots &\leq \ldots \\ \ldots &\leq \ldots \\ p(i^k) - p(i^{k-1}) &\leq w(i^{k-1}, i^k) \\ p(i^1) - p(i^k) &\leq w(i^k, i^1). \end{aligned}$$

Adding these inequalities, we get $w(i^1, i^2) + \ldots + w(i^{k-1}, i^k) + w(i^k, i^1) \geq 0$. The right had side of the inequality is the length of the cycle $(i^1, \ldots, i^k, i^1)$, which is shown to be non-negative.

Now, suppose every cycle in G has non-negative length. We construct another graph G' from G as follows. The set of vertices of G' is $N \cup \{0\}$, where 0 is a new (dummy) vertex. The set of edges of G' is $E \cup \{(0, j) : j \in N\}$, i.e., G' has all the edges of G and new edges from 0 to every vertex in N. The weights of new edges in G' are all zero, whereas weights of edges in G remain unchanged in G'. Clearly, there is a path from 0 to every vertex in G'. Observe that if G contains no cycle of negative length then G' contains no cycle of negative length. Figure 1.30 shows a directed graph and how the graph with the dummy vertex is created.

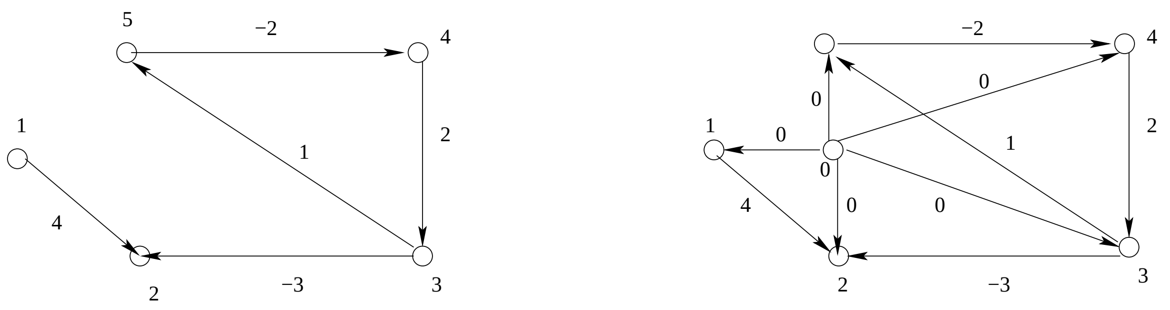


Figure 1.30: A directed graph and the new graph with the dummy vertex

We claim that $s(0, j)$ for all $j \in N$ defines a potential of graph G. Consider any $(i, j) \in E$. We consider two cases.

Case 1: The shortest path from 0 to i does not include vertex j. Now, by definition of shortest path $s(0, j) \leq s(0, i) + w(i, j)$. Hence, $s(0, j) - s(0, i) \leq w(i, j)$.

Case 2: The shortest path from 0 to i includes vertex j. In that case, $s(0,i) = s(0,j)+s(j,i)$ (by Lemma 8). Hence, $s(0,i)+w(i,j) = s(0,j)+s(j,i)+w(i,j)$. But the shortest path from j to i and then edge (i,j) creates a cycle, whose length is given by $s(j,i) + w(i,j)$. By our assumption, $s(j,i)+w(i,j) \geq 0$ (non-negative cycle length). Hence, $s(0,i)+w(i,j) \geq s(0,j)$ or $s(0,j) - s(0,i) \leq w(i,j)$.

In both cases, we have shown that $s(0,j) - s(0,i) \leq w(i,j)$. Hence $s(0,j)$ for all $j \in N$ defines a potential of graph G.

An alternate way to prove this part of the theorem is to construct G' slightly differently. Graph G' still contains a new dummy vertex but new edges are now from vertices in G to the dummy vertex 0. In such G', there is a path from every vertex in N to 0. Moreover, G' contains no cycle of negative length if G contains no cycle of negative length. We claim that $-s(j,0)$ for all $j \in N$ defines a potential for graph G. Consider any $(i,j) \in E$. We consider two cases.

Case 1: The shortest path from j to 0 does not include vertex i. Now, by definition of shortest path $s(i,0) \leq w(i,j) + s(j,0)$. Hence, $-s(j,0) - (-s(i,0)) \leq w(i,j)$.

Case 2: The shortest path from j to 0 includes vertex i. In that case, $s(j,0) = s(j,i)+s(i,0)$. Hence, $s(j,0) + w(i,j) = s(j,i) + w(i,j) + s(i,0)$. But $s(j,i) + w(i,j)$ is the length of cycle created by taking the shortest path from j to i and then taking the direct edge (i,j). By our assumption, $s(j,i) + w(i,j) \geq 0$. Hence, $s(j,0) + w(i,j) \geq s(i,0)$, which gives $-s(j,0) - (-s(i,0)) \leq w(i,j)$. ■

The proof of Theorem 12 shows a particular potential when it exists. It also shows an elegant way of verifying when a system of inequalities (of the potential form) have a solution. Consider the following system of inequalities. Inequalities of this form are called *difference inequalities*.

$$\begin{aligned}
x_1 - x_2 &\leq 2 \\
x_2 - x_4 &\leq 2 \\
x_3 - x_2 &\leq -1 \\
x_3 - x_4 &\leq -3 \\
x_4 - x_1 &\leq 0 \\
x_4 - x_3 &\leq 1.
\end{aligned}$$

To find if the above system of inequalities have a solution or not, we construct a graph with

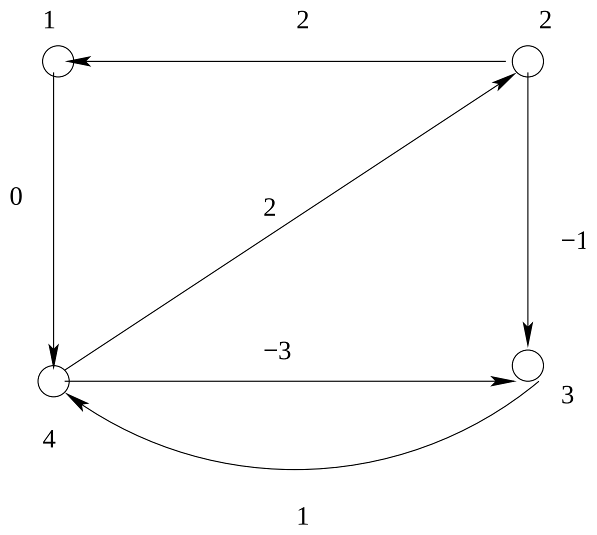


Figure 1.31: A graphical representation of difference inequalities

vertex set $\{1,2,3,4\}$ and an edge for every inequality with weights given by the right hand side of the inequalities. Figure 1.31 shows the graphical representation. Clearly, a solution to these difference inequalities correspond to potentials of this graph. It can be easily seen that the cycle $(3,4,3)$ in this graph has a length $(-3)+1=(-2)$. Hence, there exists no potential of this graph by Theorem 12.

From the proof of Theorem 12, we have established how to compute a potential when it exists. It suggests that if we have a vertex from which a path exists to every vertex or a vertex to which a path exists from every other vertex, then shortest such paths define a potential.

THEOREM **13** *Suppose $G=(N,E,w)$ is a directed graph with no cycle of negative length and i is a vertex in G such that there is a path from i to every other vertex in G. Then $p(j)=s(i,j)$ for all $j \in N \setminus \{i\}$ and $p(i)=0$ defines a potential of graph G. Similarly, if i is a vertex in G such that there is a path from every other vertex in G to i, then $p(j)=-s(j,i)$ for all $j \in N \setminus \{i\}$ and $p(i)=0$ defines a potential of graph G.*

Proof: The proof is similar to the second part of proof of Theorem 12 - the only difference being we do not need to construct the new graph G' and work on graph G directly. ■

Figure 1.32 gives an example of a complete directed graph. It can be verified that this graph does not have a cycle of negative length. Now, a set of potentials can be computed using Theorem 13. For example, fix vertex 2. One can compute $s(1,2)=3$ and $s(3,2)=4$. Hence, $(-3,0,-4)$ is a potential of this graph. One can also compute $s(2,1)=-2$ and

$s(2,3) = -3$, which gives $(-2, 0, -3)$ to be another potential of this graph.

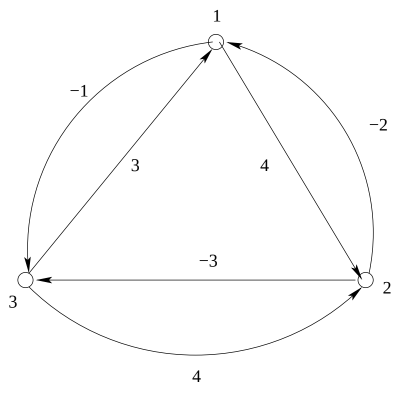


Figure 1.32: Potentials for a complete directed graph

1.8 Unique Potentials

We saw that given a digraph $G = (N, E, w)$, there may exist many potentials. Indeed, if p is a potential, then so is q, where $q(j) = p(j) + \alpha$ for all $j \in N$ and $\alpha \in \mathbb{R}$ is some constant.

There are other ways to construct new potentials from a given pair of potentials. We say a set of n-dimensional vectors $X \subseteq \mathbb{R}^n$ form a **lattice** if $x, y \in X$ implies $x \wedge y$, defined by $(x \wedge y)_i = \min(x_i, y_i)$ for all i, and $x \vee y$, defined by $(x \vee y)_i = \max(x_i, y_i)$ for all i both belong to X. We give some examples of lattices in $\mathbb{R}^2$. The whole of $\mathbb{R}^2$ is a lattice since if we take $x, y \in \mathbb{R}^2$, $x \wedge y$ and $x \vee y$ is also in $\mathbb{R}^2$. Similarly, $\mathbb{R}^2_+$ is a lattice. Any rectangle in $\mathbb{R}^2$ is also a lattice. However, a circular disk in $\mathbb{R}^2$ is not a lattice. To see this, consider the circular disk at origin of unit radius. Though $x = (1, 0)$ and $y = (0, 1)$ belong to this disk, $x \vee y = (1, 1)$ does not belong here.

The following lemma shows that the set of potentials of a digraph form a lattice.

Lemma 9 *The set of potentials of a digraph form a lattice.*

Proof: If a graph does not contain any potentials, then the lemma is true. If a graph contains a potential, consider two potentials p and q. Let $p'(i) = \min(p(i), q(i))$ for all $i \in N$. Consider any edge $(j, k) \in E$. Without loss of generality, let $p'(j) = p(j)$. Then $p'(k) - p'(j) = p'(k) - p(j) \leq p(k) - p(j) \leq w(j, k)$ (since p is a potential). This shows that p' is a potential.

Now, let $p''(i) = \max(p(i), q(i))$ for all $i \in N$. Consider any edge $(j, k) \in E$. Without loss of generality let $p''(k) = p(k)$. Then $p''(k) - p''(j) = p(k) - p''(j) \leq p(k) - p(j) \leq w(j, k)$ (since p is a potential). This shows that p'' is a potential. ■

This shows that there is more structure shown by potentials of a digraph. It is instructive to look at a complete graph with two vertices and edges of length 2 and -1. Potentials exist in this graph. Dashed regions in Figure 1.33 shows the set of potentials of this graph. Note that if edge lengths were 1 and -1, then this figure would have been the 45-degree line passing through the origin.

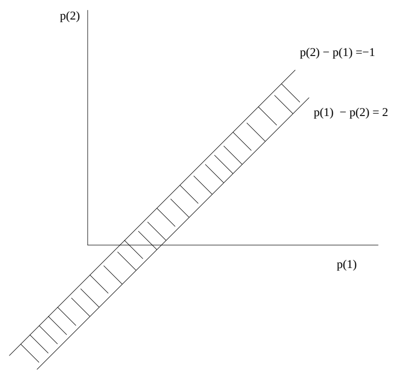


Figure 1.33: Potentials of a complete graph with two vertices

In this section, we investigate conditions under which a unique potential (up to a constant) may exist. We focus attention on strongly connected digraphs.

Definition 9 *We say a strongly connected digraph $G = (N, E, w)$ satisfies* **potential equivalence** *if for every pair of potentials p and q of G we have $p(j) - q(j) = p(k) - q(k)$ for all $j, k \in N$.*

Alternatively, it says that if a digraph G satisfies potential equivalence then for every pair of potentials p and q, there exists a constant $\alpha \in \mathbb{R}$ such that $p(j) = q(j) + \alpha$ for all $j \in N$. So, if we find one potential, then we can generate all the potentials of such a digraph by translating it by a constant.

Let us go back to the digraph with two nodes $N = \{1, 2\}$ and $w(1, 2) = 2, w(2, 1) = -1$. Consider two potentials p, q in this digraph as follows: $p(1) = 0, p(2) = 2$ and $q(1) = -1, q(2) = 0$. Note that $p(1) - q(1) = 1 \neq p(2) - q(2) = 2$. Hence, potential equivalence fails in this digraph. However, if we modify edge weights as $w(1, 2) = 1, w(2, 1) = -1$, then the p above is no longer a potential. Indeed, now we can verify that potential equivalence is satisfied. These insights are summarized in the following result.

Theorem 14 *Suppose $G = (N, E, w)$ is a strongly connected graph with no cycles of negative length. Then, G satisfies potential equivalence if and only if $s(j, k) + s(k, j) = 0$ for all $j, k \in N$.*

Proof: Suppose the digraph G satisfies potential equivalence. Then consider the potentials of G by taking shortest paths from j and k - denote them by p and q respectively. Note that p and q exist because G contains no cycles of negative length. Then we have

$$\begin{aligned} p(j) - q(j) &= p(k) - q(k) \\ \text{or } s(j,j) - s(k,j) &= s(j,k) - s(k,k), \end{aligned}$$

where $p(j) = s(j,j) = 0 = p(k) = s(k,k)$. So, we have $s(j,k) + s(k,j) = 0$.

Now, suppose $s(j,k) + s(k,j) = 0$ for all $j,k \in N$. Consider the shortest path from j to k. Let it be $(j, j^1, \ldots, j^q, k)$. Consider any potential p of G - a potential exists because G contains no cycle of negative length. We can write

$$\begin{aligned} s(j,k) &= w(j,j^1) + w(j^1,j^2) + \ldots + w(j^q,k) \\ &\geq p(j^1) - p(j) + p(j^2) - p(j^1) + \ldots + p(k) - p(j^q) \\ &= p(k) - p(j). \end{aligned}$$

Hence, we get $p(k) - p(j) \leq s(j,k)$. Similarly, $p(j) - p(k) \leq s(k,j)$ or $p(k) - p(j) \geq -s(k,j) = s(j,k)$. This gives, $p(k) - p(j) = s(j,k)$. Hence, for any two potentials p and q we have $p(k) - p(j) = q(k) - q(j) = s(j,k)$. ■

1.9 Application: Fair Pricing

Consider a market with $N = \{1, \ldots, n\}$ agents (candidates) and $M = \{1, \ldots, m\}$ indivisible goods (jobs). We assume $m = n$ (this is only for simplicity). Each agent needs to be assigned exactly one good, and no good can be assigned to more than one agent. If agent $i \in N$ is assigned to good $j \in M$, then he gets a value of v_{ij}. A price is a vector $p \in \mathbb{R}^m_+$, where $p(j)$ denotes the price of good j. Price is fixed by the market, and if agent i gets good j, he has to pay the price of good j. A **matching** μ is a bijective mapping $\mu : N \to M$, where $\mu(i) \in M$ denotes the good assigned to agent i. Also, $\mu^{-1}(j)$ denotes the agent assigned to good j in matching μ. Given a price vector p, a matching μ generates a net utility of $v_{i\mu(i)} - p(\mu(i))$ for agent i.

Definition **10** *A matching μ can be* **fairly priced** *if there exists p such that for every $i \in N$,*

$$v_{i\mu(i)} - p(\mu(i)) \geq v_{i\mu(j)} - p(\mu(j)) \qquad \forall\, j \in N \setminus \{i\}.$$

If such a p exists, then we say that μ is fairly priced by p.

Our idea of fairness is the idea of *envy-freeness*. If agent i gets object j at price $p(j)$ and agent i' gets object j' at price $p(j')$, agent i should not envy agent i''s matched object and price, i.e., he should get at least the payoff that he would get by getting object j' at $p(j')$.

Not all matchings can be fairly priced. Consider an example with two agents and two goods. The values are $v_{11} = 1, v_{12} = 2$ and $v_{21} = 2$ and $v_{22} = 1$. Now consider the matching μ: $\mu(1) = 1$ and $\mu(2) = 2$. This matching cannot be fairly priced. To see this, suppose p is a price vector such that μ is fairly priced by p. This implies that

$$v_{11} - p(1) \geq v_{12} - p(2)$$
$$v_{22} - p(2) \geq v_{21} - p(1).$$

Substituting the values, we get

$$p(1) - p(2) \leq -1$$
$$p(2) - p(1) \leq -1,$$

which is not possible. This begs the question whether there exists any matching which can be fairly priced.

Definition **11** *A matching μ is* **efficient** *if $\sum_{i \in N} v_{i\mu(i)} \geq \sum_{i \in N} v_{i\mu'(i)}$ for all other matchings μ'.*

Consider another example with 3 agents and 3 goods with valuations as shown in Table 1.2. The efficient matching in this example is: agent i gets good i for all $i \in \{1, 2, 3\}$.

	1	2	3
$v_{1\cdot}$	4	2	5
$v_{2\cdot}$	2	5	3
$v_{3\cdot}$	1	4	3

Table 1.2: Fair pricing example

Theorem **15** *A matching can be fairly priced if and only if it is efficient.*

The proof of this fact comes by interpreting the fairness conditions as potential inequalities. For every matching μ, we associate a complete digraph G^μ with set of nodes equal to the set of goods M. The digraph G^μ is a complete digraph, which means that there is an edge from every object to every other object. The weight of edge from node j to node k is given by

$$w(j,k) = v_{\mu^{-1}(k)k} - v_{\mu^{-1}(k)j}.$$

For the example in Table 1.2, we take μ to be the efficient matching, and construct G^μ. The edge lengths are:

$$\begin{aligned}
w(1,2) &= v_{22} - v_{21} = 3 \\
w(2,1) &= v_{11} - v_{12} = 2 \\
w(1,3) &= v_{33} - v_{31} = 2 \\
w(3,1) &= v_{11} - v_{13} = -1 \\
w(2,3) &= v_{33} - v_{32} = -1 \\
w(3,2) &= v_{22} - v_{23} = 2.
\end{aligned}$$

The digraph G^μ is shown in Figure 1.34.

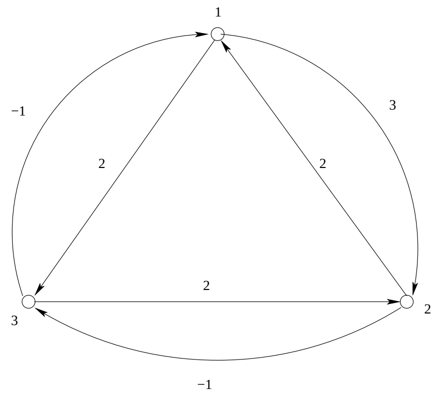


Figure 1.34: Digraph G^μ

It is easy to check then that μ can be fairly priced if there exists p such that for every $i \in N$

$$p(\mu(i)) - p(\mu(j)) \leq w(\mu(j), \mu(i)) \qquad \forall\, j \in N \setminus \{i\}.$$

Hence, μ can be fairly priced if and only if G^μ has no cycles of negative length.

Without loss of generality, reindex the objects such that $\mu(i) = i$ for all $i \in N$. Now, look at an arbitrary cycle $C = (i^1, i^2, \ldots, i^k, i^1)$ in G^μ. Denote by $R = N \setminus \{i^1, \ldots, i^k\}$. The length of the cycle C in G^μ is

$$\begin{aligned}
w(i^1,i^2)+w(i^2,i^3)+\ldots+w(i^k,i^1) &= [v_{i^2i^2}-v_{i^2i^1}]+[v_{i^3i^3}-v_{i^3i^2}]+\ldots+[v_{i^1i^1}-v_{i^1i^k}] \\
&= \sum_{r=1}^{k} v_{i^ri^r} - [v_{i^1i^k}+v_{i^2i^1}+v_{i^3i^2}+\ldots+v_{i^ki^{k-1}}] \\
&= \sum_{r=1}^{k} v_{i^ri^r} + \sum_{h\in R} v_{hh} - \sum_{h\in R} v_{hh} - [v_{i^1i^k}+v_{i^2i^1}+\ldots+v_{i^ki^{k-1}}]
\end{aligned}$$

Note that $\sum_{r=1}^{k} v_{i^ri^r} + \sum_{h\in R} v_{hh}$ is the total value of all agents in matching μ. Denote this as $V(\mu)$. Now, consider another matching μ': $\mu'(i)=\mu(i)=i$ if $i\in R$ and $\mu'(i^h)=i^{h-1}$ if $h\in\{2,3,\ldots,k\}$ and $\mu'(i^1)=i^k$. Note that μ' is the matching implicitly implied by the cycle, and $V(\mu')=\sum_{h\in R} v_{hh}+[v_{i^1i^k}+v_{i^2i^1}+v_{i^3i^2}+\ldots+v_{i^ki^{k-1}}]$. Hence, length of cycle C is $V(\mu)-V(\mu')$.

Now, suppose μ is efficient. Then, $V(\mu)\geq V(\mu')$. Hence, length of cycle C is non-negative, and μ can be fairly priced. For the converse, suppose μ can be fairly priced. Then, every cycle has non-negative length. Assume for contradiction that it is not efficient. Then, for some μ', we have $V(\mu)-V(\mu')<0$. But every μ' corresponds to a reassignment of objects, and thus corresponds to a cycle. To see this, let $C=\{i\in M:\mu^{-1}(i)\neq\mu'^{-1}(i)\}$. Without loss of generality, assume that $C=\{i^1,\ldots,i^k\}$, and suppose that $\mu'(i^r)=i^{r-1}$ for all $r\in\{2,\ldots,k\}$ and $\mu'(i^1)=i^k$. Then, the length of the cycle $(i^1,\ldots,i^k,i^1)$ is $V(\mu)-V(\mu')$. Since lengths of cycles are non-negative, this implies that $V(\mu)<V(\mu')$, which is a contradiction.

Let us revisit the example in Table 1.2. We can verify that if μ is efficient, then G^μ has no cycles of negative length. In this case, we can compute a price vector which fairly prices μ by taking shortest paths from any fixed node. For example, fix node 1, and set $p(1)=0$. Then, $p(2)=s(1,2)=3$ and $p(3)=s(1,3)=2$.

1.10 A Shortest Path Algorithm

In this section, we study an algorithm to compute a shortest path in a directed graph. We will be given a digraph $G=(N,E,w)$ with a *source* node $s\in N$. To avoid confusion, we denote the shortest path from s to every other node $u\in N$ as $\psi(s,u)$. We assume that the there is a path from the source node to every other vertex and weights of edges are non-negative. Under this assumption, we will discuss a *greedy* algorithm to compute the shortest path from s to all vertices in N. This algorithm is called the **Dijkstra's algortihm** and is based on the ideas of dynamic programming.

The algorithm goes in iterations. In each iteration, the algorithm maintains an **estimated shortest path (length)** to each node. We will denote this estimate for node u as $d(u)$. Initially, $d(s) = 0$ and $d(u) = \infty$ for all $u \neq s$.

The algorithm also maintains a set of nodes S in every stage - these are the nodes for which shortest paths have already been discovered. Initially, $S = \emptyset$. It then chooses a node $u \notin S$ with the lowest estimated shortest path among all nodes in $N \setminus S$, i.e., $d(u) \leq d(v)$ for all $v \in N \setminus S$. Then S is updated as $S \leftarrow S \cup \{u\}$. Further, the estimated shortest paths of all nodes v such that $(u, v) \in E$ is updated as follows:

$$d(v) \leftarrow \min(d(v), d(u) + w(u, v)).$$

This procedure of updating the estimated shortest path is called **relaxation** of edge (u, v). Figure 1.35 shows the process of relaxation of an edge (the numbers on nodes represent estimated shortest path and number on edges represent edge weights). In the left digraph of Figure 1.35 the estimated shortest path estimates decrease because of relaxation but on the right digraph it remains the same. This is a general property - relaxation never increases the shortest path estimates.

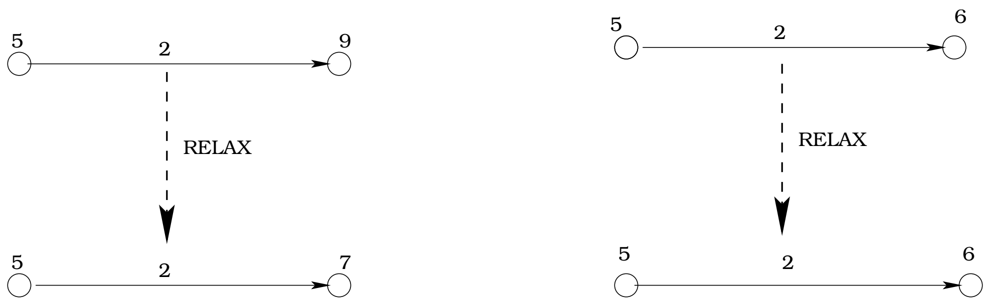


Figure 1.35: Relaxation

Once all the edges from the chosen node are relaxed, the procedure is repeated. The algorithm stops when $S = N$. The algorithm also maintains a **predecessor** for every node. Initially, all the nodes have NIL predecessor (i.e., no predecessors). If at any step, by the relaxation of an edge (u, v), the estimated shortest path decreases, then the predecessor of v is updated to u. At the end of the algorithm, the sequence of predecessors from any node u terminates at s and that defines the shortest path from s to u.

We illustrate the algorithm using an example first. The initialized estimated shortest paths are shown in Figure 1.36(a). We choose $S = \{s\}$ and relax (s, u) and (s, x) in Figure 1.36(b). Then, we add x to S.

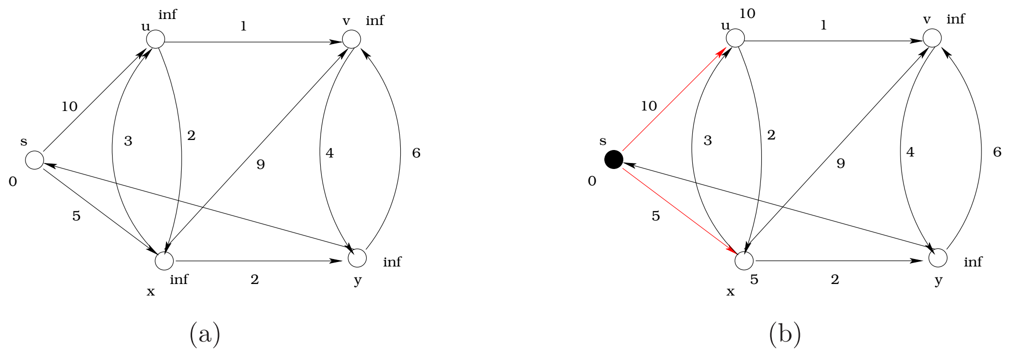


Figure 1.36: Illustration of Dijsktra's algorithm

Figure 1.37(a) shows $S = \{s, x\}$ and relaxation of edges $(x, u), (x, v)$, and (x, y). Then, we add y to S. Figure 1.37(b) shows $S = \{s, x, y\}$ and relaxation of edges (y, s) and (y, v). Then, we add u to S.

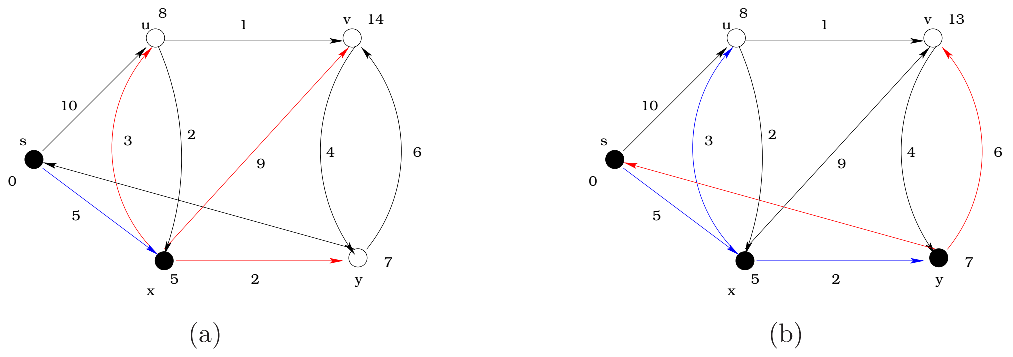


Figure 1.37: Illustration of Dijsktra's algorithm

Figure 1.38(a) shows $S = \{s, x, y, u\}$ and relaxation of edges (u, v) and (u, x). Then, we add v to S. Figure 1.38(b) shows $S = \{s, x, y, u, v\}$. At each step of the algorithm, we can maintain the edges responsible for shortest path estimates (shown in blue in the figures), and that gives the shortest path tree (rooted at s).

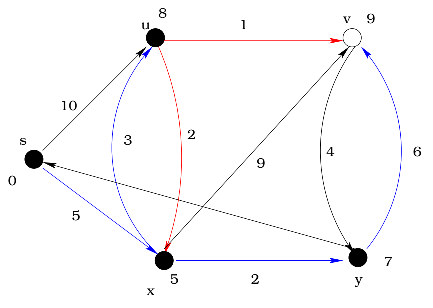
We now examine some properties of relaxation.

Lemma **10** *Suppose an edge (u, v) is relaxed, then $d(v) \leq d(u) + w(u, v)$.*

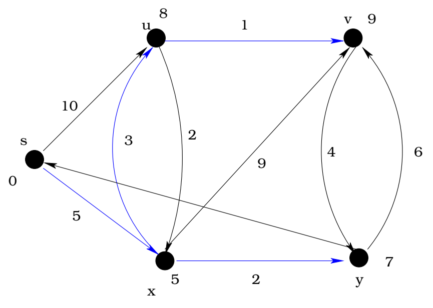
Proof: Follows from the definition of relaxation. ■

Lemma **11** *For every node $v \in N$, $d(v) \geq \psi(s, v)$.*

Proof: We show that this is true in every iteration. After initialization $d(v) = \infty$ for all $v \in N$. Hence, the claim is true for the first iteration. Assume for contradiction that in



(a)



(b)

Figure 1.38: Illustration of Dijsktra's algorithm

iteration t, the claim fails to be true for the first time. Then, it must have happened to a node $v \notin S$ in that iteration and because of relaxation of some edge (u, v). So, $d(v) < \psi(s, v)$ and (u, v) be the edge whose relaxation causes this inequality. Since this edge was relaxed, it must be $d(u) + w(u, v) = d(v) < \psi(s, v) \leq \psi(s, u) + w(u, v)$. Hence, $d(u) < \psi(s, u)$ and u was in S earlier. Since relaxing edge (u, v) does not change $d(u)$ value, this contradicts the fact that v is the first vertex for which a relaxation destroys the claimed inequality. ■

LEMMA **12** *Let (u, v) be an edge in the shortest path from s to v. If $d(u) = \psi(s, u)$ and after edge (u, v) is relaxed, we must have $d(v) = \psi(s, v)$.*

Proof: After relaxing edge (u, v), we must have $d(v) \leq d(u) + w(u, v) = \psi(s, u) + w(u, v) = \psi(s, v)$, where the last equality followed from the fact that the edge (u, v) is on the shortest path from s to v. By Lemma 11, the claim follows. ■

This leads to the correctness of Dijsktra's algorithm.

THEOREM **16** *The Dijsktra's algorithm finds the shortest paths from s to every other node.*

Proof: We go in the iteration of the Dijkstra's algorithm. Let u be the first vertex to be chosen in S for which $d(u) \neq \psi(s, u)$ - here $d(u)$ is final value of d at the end of the algorithm. By Lemma 11 $d(u) > \psi(s, u)$. Let S be the set of vertices selected so far in the algorithm. By assumption, for every vertex $v \in S$, $d(v) = \psi(s, v)$. Let P be the shortest path from s to u. Let x be the last vertex in P such that $x \in S$ - note that $s \in S$. Let the edge (x, y) be in the path P.

We first claim that $d(y) = \psi(s, y)$. Since $x \in S$, $d(x) = \psi(s, x)$. Also, edge (x, y) must have been relaxed and it belongs on the shortest path from s to y (since it belongs on the shortest path from s to u). By Lemma 12, $d(y) = \psi(s, y)$.

Now, since y occurs before u in the shortest path and all weights are non-negative, $\psi(s,y) \leq \psi(s,u)$. But $d(y) = \psi(s,y) \leq \psi(s,u) < d(u)$. But both vertices u and y are in $(N \setminus S)$ and u was chosen even though $d(u) > d(y)$. This is a contradiction. ■

1.11 Network Flows

We are given a directed graph $G = (N, E, c)$, where the weight function $c : E \rightarrow \mathbb{R}_{++}$ (positive) reflects the **capacity** of every edge. We are also given two specific vertices of this digraph: the **source** vertex, denoted by s, and the **terminal** vertex, denoted by t. Call such a digraph a **flow graph**. So, whenever we say $G = (N, E, c)$ is a flow graph, we imply that G is a digraph with a source vertex s and terminal vertex t.

A flow of a flow graph $G = (N, E, c)$ is a function $f : E \rightarrow \mathbb{R}_{+}$. Given a flow f for a flow graph $G = (N, E, c)$, we can find out for every vertex $i \in N$, the **outflow, inflow**, and **excess flow** as follows:

$$\begin{aligned}
\delta_i^-(f) &= \sum_{(i,j)\in E} f(i,j) \\
\delta_i^+(f) &= \sum_{(j,i)\in E} f(j,i) \\
\delta_i(f) &= \delta_i^+(f) - \delta_i^-(f).
\end{aligned}$$

A flow f is a **feasible flow** for a flow graph $G = (N, E, c)$ if

1. for every $(i,j) \in E$, $f(i,j) \leq c(i,j)$ and
2. for every $i \in N \setminus \{s,t\}$, $\delta_i(f) = 0$.

So, feasibility requires that every flow should not exceed the capacity and excess flow at a node must be zero.

Instead of defining flows on edges, it may be useful to define flows on $s-t$ paths (i.e., paths from s to t) and cycles. A fundamental result in network flows is that every feasible flow in a digraph can be decomposed into flows of $s-t$ paths and cycles. To state this formally, we need some notation. For every edges $e \in E$, let C^e and P^e be the set of cycles and $s-t$ paths using e. Let $\mathcal{C}$ and $\mathcal{P}$ be the set of all cycles and all $s-t$ paths in the digraph $G \equiv (N, E, c)$.

Theorem **17** *If f is a feasible flow of $G \equiv (N, E, c)$, then there exists $g : \mathcal{C} \to \mathbb{R}_+$ and $h : \mathcal{P} \to \mathbb{R}_+$ such that for every $e \in E$, we have*

$$f(e) = \sum_{C \in C^e} g(C) + \sum_{P \in P^e} h(P).$$

Proof: The proof is constructive. Consider any feasible flow f of $G \equiv (N, E, c)$. We will successively define new feasible flows for G. These new flows are derived by using the observation that by reducing the flow (a) either along an $s-t$ path or (b) along a cycle gives another feasible flow.

- $s-t$ Paths. Pick any $s-t$ path P. If $f(e) > 0$ for all $e \in P$, then

  $$h(P) = \min_{e \in P} f(e)$$
  $$f(e) := f(e) - h(P) \ \forall \ e \in P.$$

  Repeat this procedure till every $s-t$ path has at least one edge with zero flow. Note that the procedure produces a feasible flow in each iteration.

- Cycles. Once the $s-t$ path flows are determined, the only positive feasible flow remaining must be along cycles - by definition, no $s-t$ path can have positive flow. Pick any cycle C. If $f(e) > 0$ for all $e \in P$, then

  $$g(C) = \min_{e \in C} f(e)$$
  $$f(e) := f(e) - g(C) \ \forall \ e \in C.$$

  Repeat this procedure till every cycle has zero flow. Clearly, since the $s-t$ paths do not have positive flow, this procedure will produce a stage where all cycles have zero flow.

These two steps establish the claim of the theorem constructively. ■

To get an idea of the construction, consider the digraph in Figure 1.39 - capacities are infinite in this digraph.

Figures 1.40(a), 1.40(b), 1.41(a), 1.41(b), and 1.42 illustrate how the flows have been decomposed into flows along $s-t$ paths and flows along cycles. In each step the flow along an $s-t$ path is reduced (such a path is shown in Red in the figures) and that amount of flow is assigned to this path. Eventually (in Figure 1.42), only flows along two cycles remain, which are assigned to these cycles.

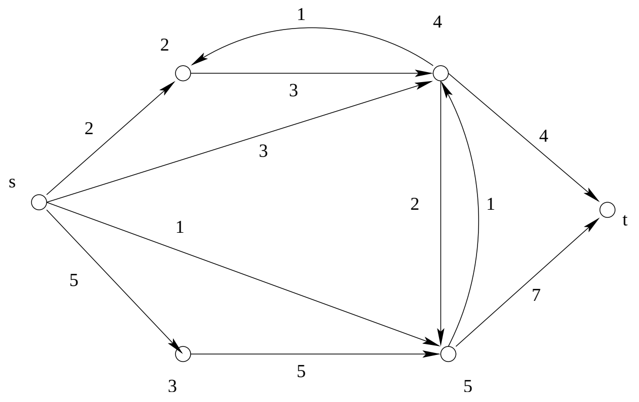


Figure 1.39: Feasible flow

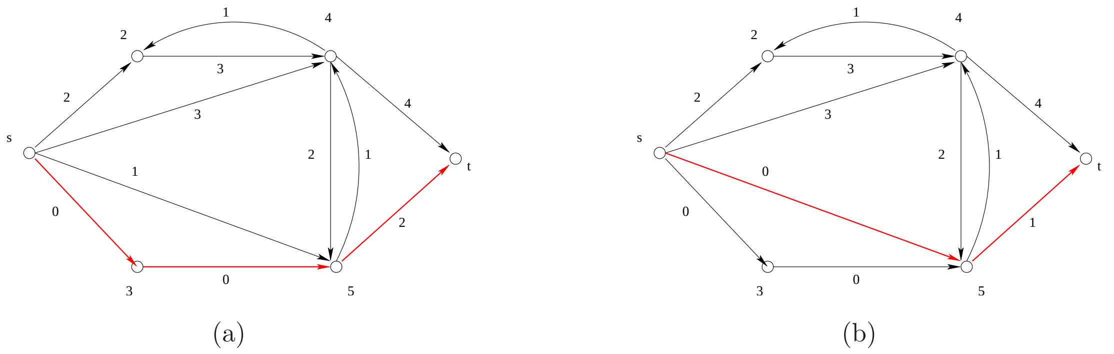


Figure 1.40: Flow decomposition: $s-3-5-t$ and $s-5-t$ paths

1.11.1 The Maximum Flow Problem

Definition **12** *The* **maximum flow problem** *is to find a feasible flow f of a flow graph $G=(N,E,c)$ such that for every feasible flow f' of G, we have*

$$\delta_t(f) \geq \delta_t(f').$$

The **value** of a feasible flow f in flow graph $G=(N,E,c)$ is given by $\nu(f)=\delta_t(f)$. So, the maximum flow problem tries to find a feasible flow that maximizes the value of the flow.

Figure 1.43 shows a flow graph with a feasible flow. On every edge, capacity followed by flow amount is written. It is easy to verify that this flow is feasible (but verify that this is not a maximum flow).

The maximum flow problem has many applications. One original application was railways. Suppose there are two cities, say Delhi and Mumbai, connected by several possible rail networks (i.e., routes which pass through various other cities). We want to determine what is the maximum traffic that can go from Delhi to Mumbai. The capacity of traffic from a

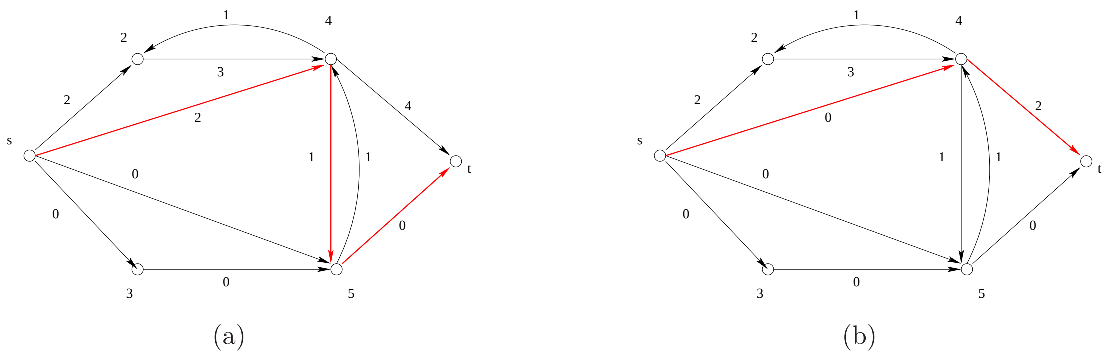


Figure 1.41: Flow decomposition: $s-4-5-t$ and $s-4-t$ paths

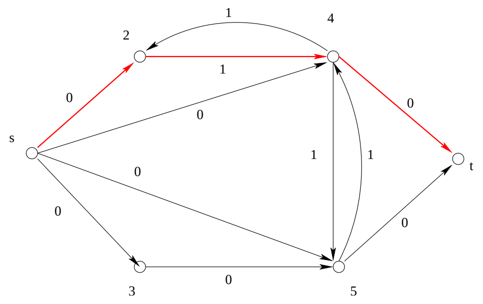


Figure 1.42: Flow decomposition: $s-2-4-t$ path and $(2,4,2)$ and $(4,5,4)$ cycles

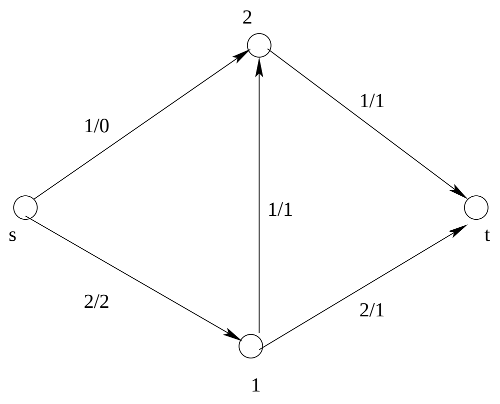


Figure 1.43: Feasible flow

city to another city is given (by the train services between these two cities). So, the solution of the problem is the solution to the max flow problem.

1.11.2 Analysis of the Maximum Flow Problem

We now analyze some properties of the maximum flow problem. The first property is immediate.

LEMMA **13** *Suppose f is a feasible flow of flow graph $G = (N, E, c)$ with s and t being the source and terminal vertices. Then, $\delta_s(f) + \delta_t(f) = 0$.*

Proof: Since $\delta_i(f) = 0$ for all $i \in N \setminus \{s, t\}$, we have $\sum_{i \in N} \delta_i(f) = \delta_s(f) + \delta_t(f)$. But we know that $\sum_{i \in N} \delta_i(f) = 0$ - the inflow of one node is the outflow of some other node. Hence, $\delta_s(f) + \delta_t(f) = 0$. ■

A consequence of this lemma is that the problem of finding a maximum flow can be equivalently formulated as minimizing the excess flow in the source.

The concept of a **cut** is important in analyzing the maximum flow problem. In flow graphs, a cut is similar to a cut in an undirected graph: a partitioning of the set of vertices. An (s, t)-cut of digraph $G = (N, E, c)$ is $(S, N \setminus S)$ such that $s \in S$ and $t \in N \setminus S$. For every such cut, $(S, N \setminus S)$, define $S^- := \{(i, j) \in E : i \in S, j \notin S\}$ and $S^+ = \{(i, j) : j \in S, i \notin S\}$. The capacity of a cut $(S, N \setminus S)$ in flow graph $G = (N, E, c)$ is defined as

$$\kappa(S) = \sum_{(i,j) \in S^-} c(i, j).$$

DEFINITION **13** *An (s, t)-cut $(S, N \setminus S)$ in flow graph $G = (N, E, c)$ is called a* **saturated cut** *for flow f if*

1. *$f(i, j) = c(i, j)$ for all $(i, j) \in S^-$ and*

2. *$f(i, j) = 0$ for all $(i, j) \in S^+$.*

Figure 1.44 shows a saturated cut: $(\{s, 2\}, \{1, t\})$.

The second part of the definition of saturated cut is equally important. Figure 1.45 shows a cut in the same graph which meets the first part of the definition (i.e. flow in the cut equals capacity) but fails the second part since $f(1, 2) \neq 0$.

LEMMA **14** *For any feasible flow f of a flow graph $G = (N, E, c)$ and any (s, t)-cut $(S, N \setminus S)$ of G*

1. *$\nu(f) \leq \kappa(S)$,*

2. *if $(S, N \setminus S)$ is saturated for flow f, $\nu(f) = \kappa(S)$.*

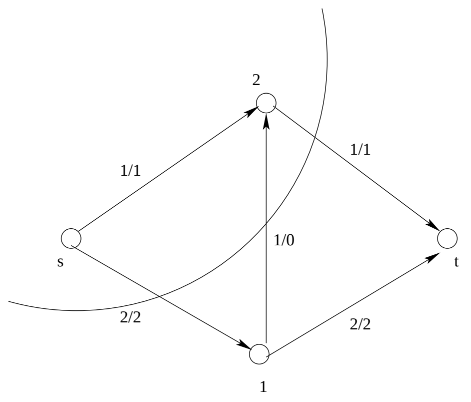


Figure 1.44: Saturated cut

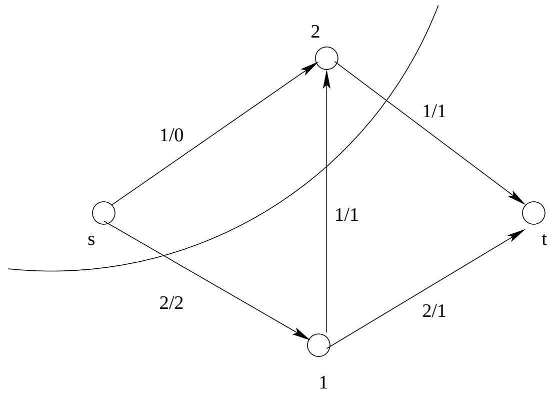


Figure 1.45: A cut which is not saturated

3. if $(S, N \setminus S)$ is saturated for flow f, then f is a maximum flow.

Proof: We prove (1) first. Using Lemma 13, we get

$$\begin{aligned}\nu(f) &= -\delta_s(f) = -\sum_{i \in S} \delta_i(f) \\ &= \sum_{(i,j) \in S^-} f(i,j) - \sum_{(i,j) \in S^+} f(i,j) \\ &\leq \sum_{(i,j) \in S^-} f(i,j) \\ &\leq \sum_{(i,j) \in S^-} c(i,j) \\ &= \kappa(S),\end{aligned}$$

where the inequality comes from feasibility of flow. Note that both the inequalities are equalities for saturated flows. So (2) follows. For (3), note that if $(S, N \setminus S)$ is saturated for flow f, then $\nu(f) = \kappa(S)$. For any feasible flow f', we know that $\nu(f') \leq \kappa(S) = \nu(f)$. Hence, (3) follows. ■

Lemma 14 says that if there is a (s,t)-cut which is saturated for flow f, then f is a maximum flow. However, if f is a maximum flow, then not every (s,t)-cut is saturated. Figure 1.46 shows a maximum flow and an (s,t)-cut which is not saturated.

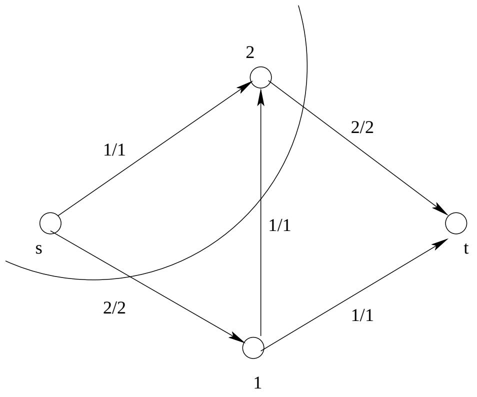


Figure 1.46: A maximum flow f does not imply every (s,t)-cut is saturated for f

1.11.3 The Residual Digraph of a Flow

We now proceed to identify some key properties which will help us identify a maximum flow. The first is the construction of a residual digraph. Let f be a feasible flow in flow graph $G = (N, E, c)$. We define the **residual digraph** G^f for flow f as follows:

- The set of vertices is N (same as the set of vertices in G).
- For every edge $(i,j) \in E$,
  - FORWARD EDGE: if $f(i,j) < c(i,j)$, then (i,j) is an edge in G^f with capacity $c^f(i,j) = c(i,j) - f(i,j)$.
  - REVERSE EDGE: if $f(i,j) > 0$, then (j,i) is an edge in G^f with capacity $c^f(j,i) = f(i,j)$.

Note that this may create two edges from some i to some j in G^f. In that case, we keep the edge which has minimum capacity. The set of edges of G^f is denoted as E^f. So, the residual digraph $G^f = (N, E^f, c^f)$. We illustrate the residual digraphs for two flows in Figures 1.47 and 1.48.

The next theorem illustrates the importance of a residual digraph.

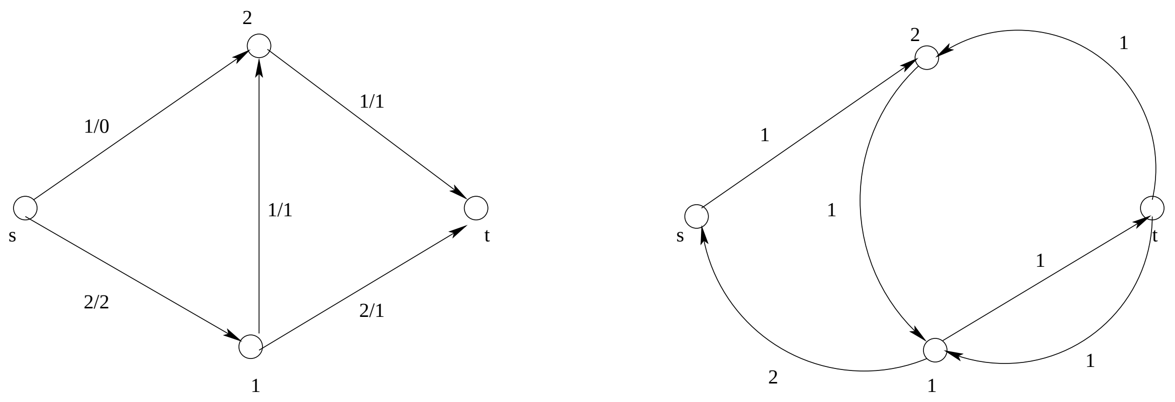


Figure 1.47: Residual digraph of a flow

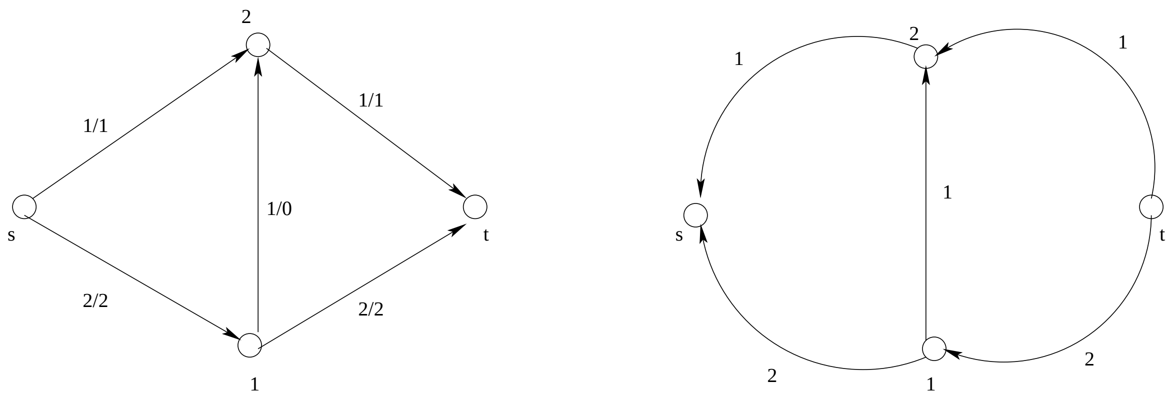


Figure 1.48: Residual digraph of a flow

Theorem **18** *Let f be a feasible flow of a flow graph $G = (N, E, c)$ and G^f be the residual digraph for flow f. The feasible flow f is a maximum flow for G if and only if there is no path from s to t in G^f.*

Proof: Suppose f is a maximum flow. Assume for contradiction that there is path $P = (s, v_1, v_2, \dots, v_k, t)$ from s to t in G^f. Let E^P be the set of edges in P corresponding to original graph G, and let E^{P^+} be the set of forward edges in P and E^{P^-} be the set of reverse edges in P (again, corresponding to original graph G). Define $\delta = \min_{(i,j)\in E^P} c^f(i, j)$ and let

- $f'(i, j) = f(i, j) + \delta$ if $(i, j) \in E^{P^+}$,
- $f'(i, j) = f(i, j) - \delta$ if $(i, j) \in E^{P^-}$, and
- $f'(i, j) = f(i, j)$ if $(i, j) \in E \setminus E^P$.

Call such a path P an **augmenting path**. By definition $\delta > 0$.

First, we show that f' is a feasible flow of G. We show that in two steps:

Capacity and Non-Negativity Constraints. For any edge not corresponding to path P in G^f, the capacity constraints are met since f is feasible. For a forward edge $(i,j) \in E^P$, we increase the flow by $\delta \leq c^f(i,j) = c(i,j) - f(i,j)$. Hence, $f(i,j) + \delta \leq c(i,j)$, and capacity constraints are satisfied. For a reverse edge $(i,j) \in E^P$, we decrease flow by $\delta \leq c^f(i,j) = f(i,j)$, and hence, $f(i,j) - \delta \geq 0$. So, non-negativity constraints in these edges is satisfied.

Flow Balancing Constraints. For any vertex i not part of path P, the flow balancing constraints hold. For any vertex $i \notin \{s,t\}$ in P, let (i^p, i) be the edge incoming to i in P and (i, i^s) be the edge outgoing from i in P. The following possibilities exist:

1. (i^p, i) and (i, i^s) are forward edges, in which case δ is added to incoming flow to i and δ is added from the outgoing flow of i in G. So, flow balancing holds.

2. (i^p, i) is a forward edge and (i, i^s) is a reverse edge. Then, δ is added to incoming flow (by (i^p, i)), but subtracted from the incoming flow (by (i^s, i)). So, flow balancing holds.

3. (i^p, i) is a reverse edge and (i, i^s) is a forward edge. Then, δ is subtracted from the outgoing flow (by (i, i^p)) but added to the outgoing flow (by (i, i^s)). So, flow balancing holds.

4. (i^p, i) and (i, i^s) are reverse edges. Then, δ is subtracted from outgoing flow and also subtracted from the incoming flow. So, flow balancing holds.

So, f' is a feasible flow. Let (v_k, t) be the unique edge in P which is incoming to t. Note that there is no outgoing edge of t which is part of P. If (v_k, t) is a forward edge, then the inflow to t is increased by δ from f to f'. If (v_k, t) is a reverse edge, then the outflow from t is decreased by δ from f to f'. In either case, the excess flow of t is increased from f to f' by δ. So, $\nu(f') = \nu(f) + \delta > \nu(f)$. Hence, f is not a maximum flow. This is a contradiction.

It is worth going to Figure 1.47, and understand the augmenting path a bit more. Here, the augmenting path in the residual digraph is $(s, 2, 1, t)$. Note that $\delta = 1$. So, we push 1 unit of flow more on $(s, 2)$, then push back 1 unit of flow on $(1, 2)$, and finally push 1 unit of flow more on $(1, t)$.

Suppose there is no path from s to t in G^f. Let S be the set of all vertices i in G^f such that there is a path from s to i. Now, $(S, N \setminus S)$ defines an (s,t)-cut in G. Since there is no path from s to t in G^f, there is no edge $(i,j) \in E^f$ such that $i \in S$ and $j \in (N \setminus S)$ in G^f. This implies that in the original flow graph G,

1. for every edge $(i,j) \in E$ such that $i \in S$ and $j \in (N \setminus S)$, we have $f(i,j) = c(i,j)$, and
2. for every edge $(j,i) \in E$ such that $i \in S$ and $j \in (N \setminus S)$, we have $f(j,i) = 0$.

This implies that the cut $(S, N \setminus S)$ is a saturated cut for flow f in flow graph G. By Lemma 14, f is a maximum flow. ■

This theorem leads to one of the most well known results in graph theory. Denote by F^G, the set of all feasible flows of a flow graph $G = (N,E,c)$. Denote by S^G, the set $\{S \subseteq N : (S, N \setminus S) \text{ is an } (s,t)\text{-cut of } G\}$.

THEOREM **19** *For every flow graph $G = (N,E,c)$ with a source vertex s and a terminal vertex t*

$$\max_{f \in F^G} \nu(f) = \min_{S \in S^G} \kappa(S).$$

Proof: Suppose f is a maximum flow. It is immediate that $\nu(f) \leq \kappa(S)$ for any $S \in S^G$ (by Lemma 14). By Theorem 18, there is no path from s to t in the residual digraph G^f. Let S be the set of nodes for which there is some path from s in G^f. So, $(S, N \setminus S)$ is an (s,t)-cut in G. Since there is no path from s to t in G^f, $(S, N \setminus S)$ is a saturated cut for flow f in G. Hence, $\nu(f) = \kappa(S)$ (by Lemma 14). This implies that $\nu(f) = \min_{S \in S^G} \kappa(S)$. ■

1.11.4 Ford-Fulkerson Algorithm

The following algorithm, known as the Ford-Fulkerson algorithm, finds the maximum flow of a flow graph if the capacities are rational.

We are given a flow graph $G = (N,E,c)$ with a source vertex s and a terminal vertex t. Assume that there is at least one path from s to t. Then, the algorithm goes as follows.

S0 Start with zero flow, i.e. $f(i,j) = 0$ for all $(i,j) \in E$.

S1 Construct the residual digraph G^f.

S2 Check if the residual digraph G^f has a path from s to t.

S2.1 If not, STOP - f is the maximum flow.

S2.2 If yes, increase flow along an **augmenting path** (i.e., a path in G^f from s to t) by the minimum residual capacity on that path as shown in Theorem 18 (feasibility is maintained). Iterate from Step S1.

If the algorithm terminates, then from Theorem 18 it must find a maximum flow. If the flows are integral, then the algorithm must terminate since capacities are finite integers and in every iteration the flow increases by at least one. Note that if all capacities are integral, then the algorithm outputs an integral flow. So, if all capacities are integral, then the maximum flow is also an integral flow. As an exercise, find the maximum flow of the digraph in Figure 1.49. You can verify that the maximum flow amount is 7.

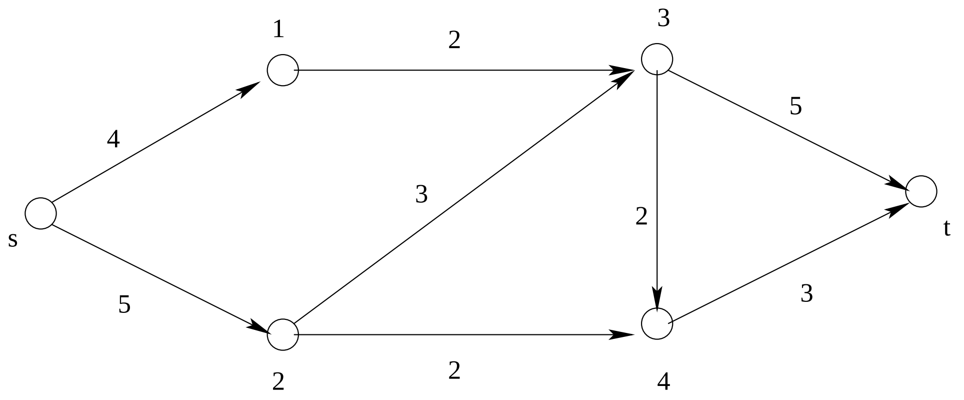


Figure 1.49: Maximum flow

We show some of the steps. Let us start from a feasible flow as shown in Figure 1.50. Then, Figure 1.51 shows the residual digraph for this flow. We see that there is a path from s to t in this graph: $(s, 2, 3, t)$. We can augment flow along this path by 3 units. The new flow is shown in Figure 1.52. This is a maximum flow since if the cut $(\{s, 1, 2\}, \{3, 4, t\})$ has a capacity of 7, and this flow value is also 7.

We now formally prove how the Ford-Fulkerson algorithm finds a maximum flow if the capacities are rational numbers.

THEOREM **20** *If all capacities are rational, then the Ford-Fulkerson algorithm terminates finitely with a maximum flow.*

Proof: If all capacities are rational then there exists a natural number K such that $Kc(i, j)$ is an integer for all $(i, j) \in E$. Then, in every iteration, the flow is augmented by at least $\frac{1}{K}$. Since the flow value is bounded (by the minimum cut capacity), the algorithm must terminate finitely. ■

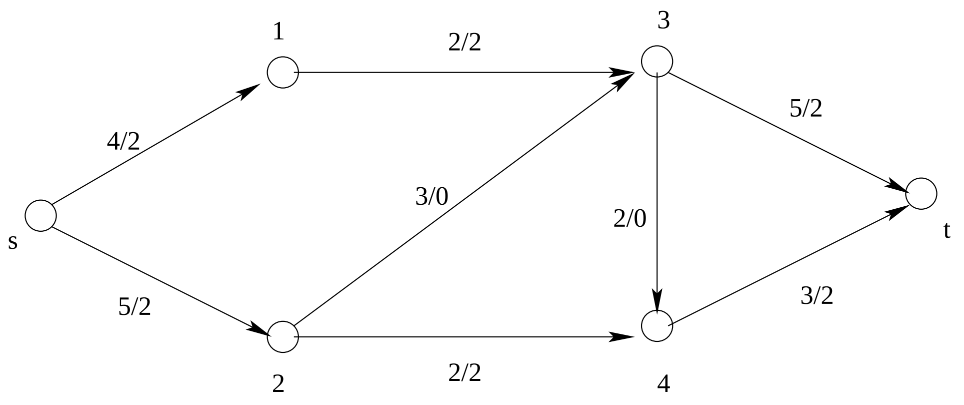


Figure 1.50: A feasible flow

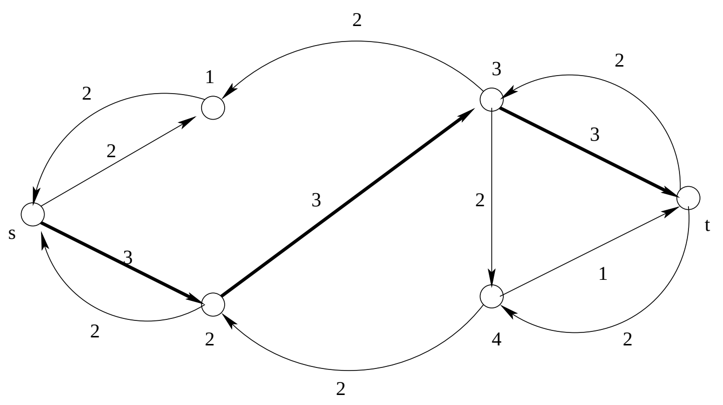


Figure 1.51: Residual digraph for flow in Figure 1.50

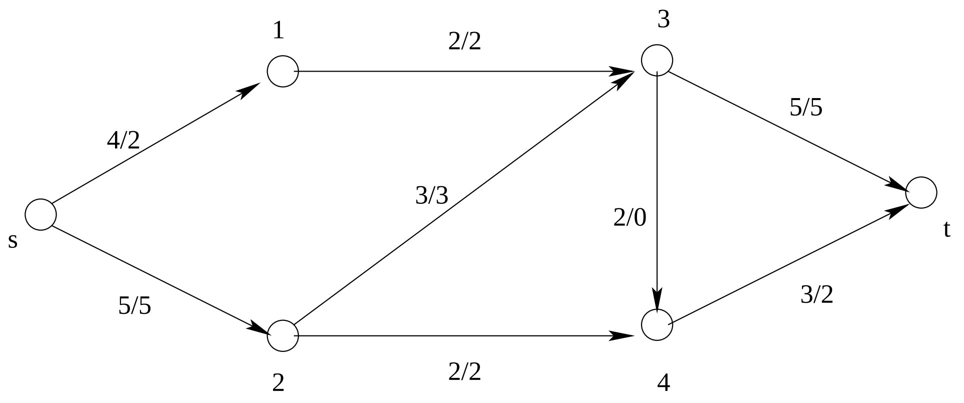


Figure 1.52: Maximum flow for the flow graph in Figure 1.49

However, the algorithm may not terminate in general for irrational capacities. You are encouraged to think of an example with irrational capacities. If all capacities are integers, then the Ford-Fulkerson algorithm generates integral flows in every iteration and hence, terminates with an integral maximum flow. This leads to the following lemma.

Lemma **15** *If all capacities are integers, then the maximum flow is an integral flow.*

1.12 Disjoint Paths

We now study another graph problem, and a max-min relation on this. In this problem, we are given a digraph and two vertices in it, and we are asked to find the maximum number of disjoint paths in such a digraph. This has applications in communication networks. For instance, if we know there are two disjoint paths, we know that if one of the edges on one path fails, there is another "back-up" path.

The premise of this problem is a directed graph $G = (N, E)$ (not weighted) and two special vertices s and t. We are interested in finding the number of edge-disjoint paths from s to t, where two paths are edge disjoint if they do not share an edge. Two disjoint paths (in dark black and blue) for a digraph are shown in Figure 1.53.

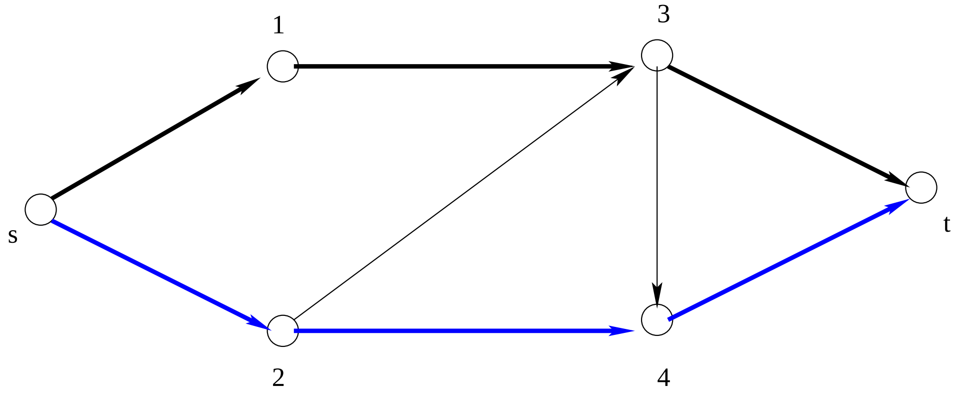


Figure 1.53: Two disjoint paths

The dual problem to this is the following network connectivity problem. Suppose there is digraph G with source vertex s and terminal vertex t. We want to find out what is the minimum number of edges from G that must be removed to *disconnect* t from s, i.e., no path from s to t. The following theorem, and the ensuing corollary, show that the two problems are related.

Theorem **21 (Menger's Theorem)** *A digraph $G = (N, E)$ with source vertex s and terminal vertex t has at least k disjoint paths from s to t if and only if there is a path from s to t after deleting any $(k - 1)$ edges from G.*

Proof: Suppose there are at least k disjoint paths from s to t. Then deleting $(k-1)$ edges from G will not delete one edge from at least one of the paths from s to t. Hence, there will remain at least one path from s to t.

Suppose there is a path from s to t after deleting any $(k-1)$ edges from G. We convert G into a flow graph, where the capacity function is defined as follows: $c(i,j) = 1$ for all $(i,j) \in E$. Note that since capacities of every edge is an integer, the capacity of every cut is an integer, and by the max-flow min-cut Theorem (Theorem 19), the maximum flow of this digraph is an integer. Further, since there is at least one path from s to t after deleting any $(k-1)$ edges from G, the capacity of any (s,t)-cut in G must be at least k. Hence, the maximum flow in G must be at least k and an integer. Further, if we apply the Ford-Fulkerson algorithm, it must terminate with integral flows (Lemma 15).

Now, by the flow decomposition theorem (Theorem 17) any feasible flow in G can be decomposed into flows on (a) a set of paths from s to t and flows along (b) cycles. Consider an integral maximum flow. If the flow decomposition of such a flow assigns flows along any cycle, we can reduce the flows along the cycles to zero without reducing the maximum flow. Hence, without loss of generality, the flow decomposition of the maximum flow assigns flows to $s-t$ paths only.

Since the capacity of every edge is just 1 and flows are integral, one edge can carry flow of only one path from s to t. Hence, each unit of flow from s to t corresponds to a unique path from s to t. So, there are at least k disjoint paths from s to t to carry k units of flow. ■

An immediate corollary to this result is the following.

COROLLARY **2** *Suppose $G = (N,E)$ is a digraph with source vertex s and terminal vertex t. The number of disjoint paths from s to t in G equals the minimum number of edges that need to be removed from G such that there are no paths from s to t.*

Proof: Suppose there are k disjoint paths from s to t. Let the minimum number of edges that need to be removed from G such that there are no paths from s to t be ℓ. This means by deleting any $\ell - 1$ edges from G, there is still a path from s to t. By Theorem 21, $k \geq \ell$. Suppose $k > \ell$. Then, again by Theorem 21, by removing any $k-1$ edges there is still a path from s to t. This contradicts the fact that by removing ℓ edges there is no path from s to t. ■

A small comment about disjoint paths. Suppose we have k disjoint paths. Let the first edges of these paths be: $(s,i^1),(s,i^2),\ldots,(s,i^k)$. This obviously means there are at least

k edges from s to the rest of the graph. However, **it does not mean** that by deleting $(s, i^1), (s, i^2), \dots, (s, i^k)$, there is no path from s to t. The example in Figure 1.54 illustrates that. There are three disjoint paths from s to t in the graph in Figure 1.54. Take for example the disjoint paths $(s, 1, 2, t)$, $(s, 4, t)$, and $(s, 3, 4, t)$. If we remove two edges $(s, 1), (s, 4), (s, 3)$, there are still paths from s to t - the path $(s, 2, t)$ still remains. The question of finding the correct minimal set of edges that need to be removed boils down to identifying the min capacity cut of the underlying flow network.

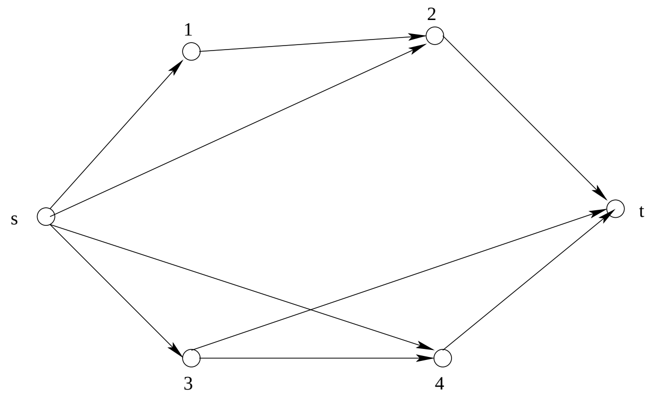


Figure 1.54: Two disjoint paths

Chapter 2

Introduction to Convex Sets

2.1 Convex Sets

A set $C \subseteq \mathbb{R}^n$ is called **convex** if for all $x, y \in C$, we have $\lambda x + (1-\lambda)y \in C$ for all $\lambda \in [0,1]$. The definition says that for any two points in set C, all points on the segment joining these two points must lie in C for C to be convex. Figure 2.1 shows two sets which are convex and two sets which are not convex.

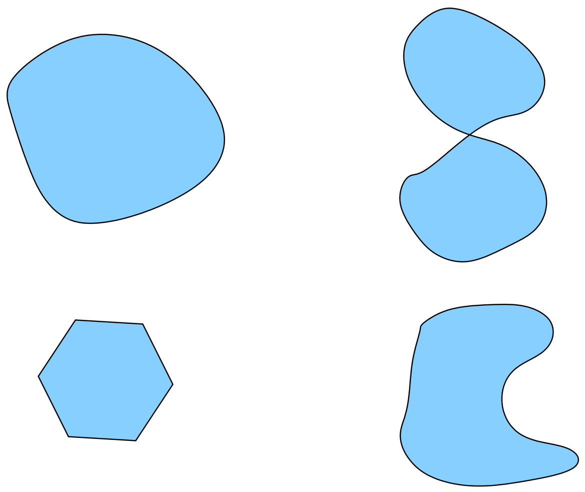


Figure 2.1: Sets on left are convex, but sets on right are not

Examples of convex sets:

- $C = \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 + 2x_2 - x_3 = 4\}$. This is the equation of a plane in $\mathbb{R}^3$. In general, a **hyperplane** is defined as $C = \{x \in \mathbb{R}^n : p \cdot x = \alpha\}$, where $\alpha \in \mathbb{R}$ and $p \in \mathbb{R}^n$, called the normal to the hyperplane. Notation: As before $p \cdot x$ means dot product of p and x, i.e., $\sum_{i=1}^{n} p_i x_i$. For simplicity, we will sometimes write this as px. We will often denote a hyperplane as (p, α).

A hyperplane is a convex set. Take any $x, y \in C$. Then for any $\lambda \in [0,1]$ define $z = \lambda x + (1-\lambda)y$. Now, $pz = \lambda px + (1-\lambda)py = \lambda\alpha + (1-\lambda)\alpha = \alpha$. Hence $z \in C$.

- $C = \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 + 2x_2 - x_3 \leq 4\}$. These are points on one side of the hyperplane. In general, a **half-space** is defined as $C = \{x \in \mathbb{R}^n : p \cdot x \leq \alpha\}$, where $\alpha \in \mathbb{R}$ and $p \in \mathbb{R}^n$. As in the case of a hyperplane, every half-space is a convex set.

- $C = \{(x_1, x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 4\}$. The set C is a circle of radius two with center $(0, 0)$.

- $C = \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 + x_2 - x_3 \leq 2, x_1 + 2x_2 - x_3 \leq 4\}$. Set C is the intersection of two half-spaces. In general, intersection of a finite number of half-spaces is called a **polyhedral set**, and is written as $C = \{x : Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Here, C is the intersection of m half-spaces. A polyhedral set is convex, because intersection of convex sets is a convex set, which we prove next.

LEMMA **16** *If C_1 and C_2 are two convex sets, then so is $C_1 \cap C_2$.*

Proof: Suppose $x, y \in C_1 \cap C_2$. Define $z = \lambda x + (1-\lambda)y$ for some $\lambda \in [0,1]$. Since $x, y \in C_1$ and C_1 is convex, $z \in C_1$. Similarly, $z \in C_2$. Hence, $z \in C_1 \cap C_2$. ■

Weighted averages of the form $\sum_{i=1}^{k} \lambda_i x^i$ with $\sum_{i=1}^{k} \lambda_i = 1$ and $\lambda_i \geq 0$ for all i is called **convex combination** of points $(x^1, \ldots, x^k)$.

DEFINITION **14** *The* **convex hull** *of a set $C \subseteq \mathbb{R}^n$, denoted as $H(C)$, is collection of all convex combinations of C, i.e., $x \in H(C)$ if and only if x can be represented as $x = \sum_{i=1}^{k} \lambda_i x^i$ with $\sum_{i=1}^{k} \lambda_i = 1$ and $\lambda_i \geq 0$ for all i and $x^1, \ldots, x^k \in C$ for some integer k.*

Figure 2.2 shows some sets and their convex hulls.

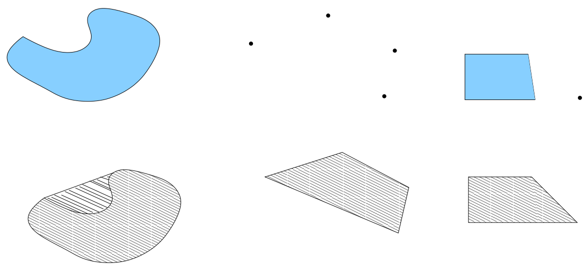


Figure 2.2: Convex Hulls

The convex hull of a set is a convex set. Take $x, y \in H(C)$. Define $z = \lambda x + (1-\lambda)y$ for any $\lambda \in [0,1]$. This is a convex combination of x and y, which in turn is a convex combination of points in C. Hence, z can be written as convex combination of points in C. In fact, z is a special convex set.

Lemma **17** *Suppose $C \subseteq \mathbb{R}^n$. Then, $H(C)$ is the smallest convex set containing C, i.e., suppose S is a convex set such that $C \subseteq S$, then $H(C) \subseteq S$.*

We skip the proof. A consequence of this lemma is that the convex hull of a convex set is the same set.

2.2 Hyperplanes and Separations

In this section, we prove an important result in convexity. It deals with separating a point outside a convex set from the convex set itself by using a hyperplane.

Definition **15** *Let S_1 and S_2 be two non-empty sets in $\mathbb{R}^n$. A hyperplane $H = \{x : px = \alpha\}$ is said to* **separate** *S_1 and S_2 if $px \geq \alpha$ for each $x \in S_1$ and $px \leq \alpha$ for each $x \in S_2$.*

The hyperplane H is said to **strictly separate** *S_1 and S_2 if $px > \alpha$ for all $x \in S_1$ and $px < \alpha$ for all $x \in S_2$.*

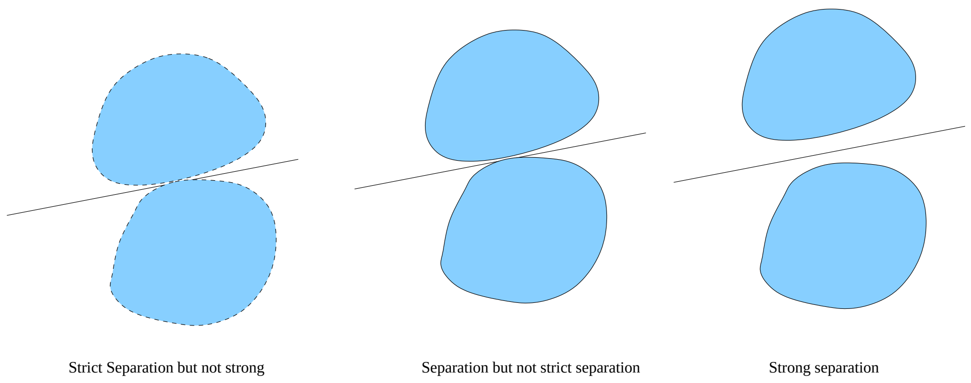

Strict Separation but not strong
Separation but not strict separation
Strong separation

Figure 2.3: Different types of separation

The idea of separation is illustrated in Figure 2.3. Not every pair of sets can be separated. For example, if the interior of two sets intersect, then they cannot be separated. Figure 2.4 shows two pairs of sets, one of which can be (strictly) separated but the other pair cannot. So, what kind of sets can be separated. There are various results, all some form of separation theorems, regarding this. We will study a particular result.

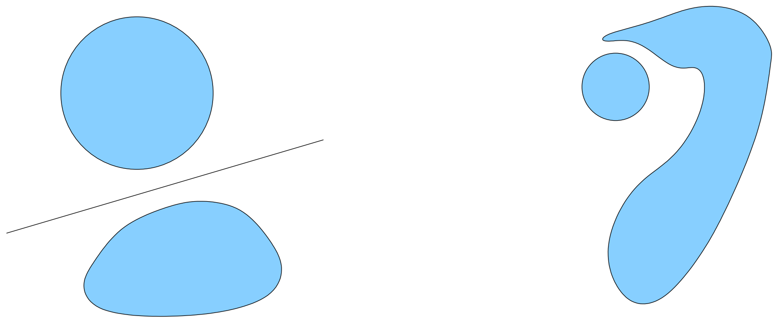


Figure 2.4: Possibility of separation

THEOREM **22** *Let C be a closed convex set in $\mathbb{R}^n$ and $z \in \mathbb{R}^n$ be a point such that $z \notin C$. Then, there exists a hyperplane which strictly separates z and C.*

Before we prove the theorem, it is instructive to look at Figure 2.5, which gives a geometric interpretation of the proof. We take a point $z \notin C$. Take the closest point to z in C, say y. Then take the hyperplane passing through the mid-point between z and y and perpendicular to the vector $z - y$. The proof involves showing that this hyperplane separates z and C.

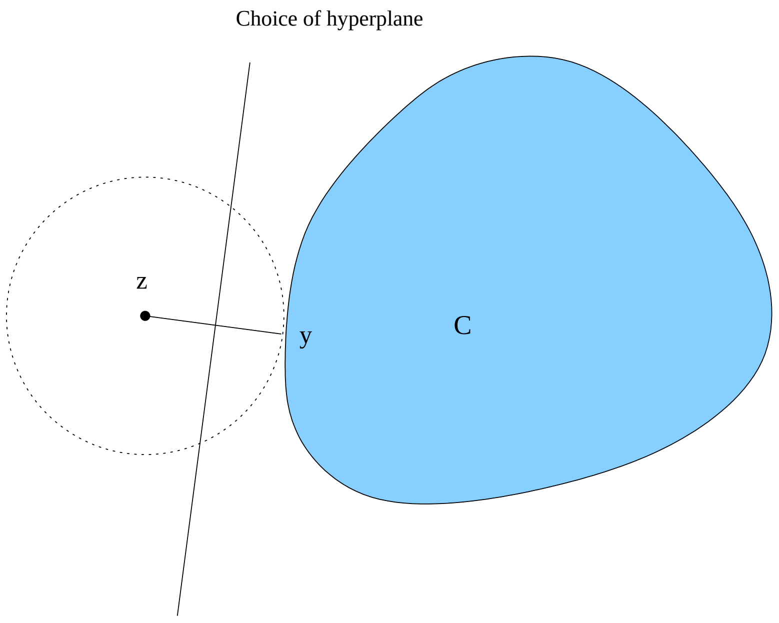


Figure 2.5: Geometric illustration of separating hyperplane theorem

Proof: The proof is trivial if $C = \emptyset$. Suppose $C \neq \emptyset$. Then, consider $z \notin C$. Since $z \notin C$ and C is closed and convex, there exists a point $y \in C$ such that $\|z - y\|$ (i.e., the Euclidean distance between z and y) is minimized. (This claim is trivial geometrically - take any ball $B(z, r)$, which is centered at z and has a radius r, and grow it till it intersects C. Since $C \neq \emptyset$, it will intersect $B(z, r)$ for some r. Also, since C is closed and convex, $B(z, r) \cap C$ is closed and convex. Also, since $B(z, r)$ is bounded, $B(z, r) \cap C$ is compact. So, we need to minimize the continuous function $\|z - y\|$ over a compact set $B(z, r) \cap C$. By Weierstrass' Theorem, a minimum exists of such a function.)

Now set $p = z - y$ and $\alpha = \frac{1}{2}\big[\|z\|^2 - \|y\|^2\big]$. We show that $pz > \alpha$ and $px < \alpha$ for all $x \in C$.

Now,

$$\begin{aligned} pz - \alpha &= (z - y)z - \frac{1}{2}\big[\|z\|^2 - \|y\|^2\big] \\ &= \frac{1}{2}\big[\|(z - y)\|^2\big] \\ &> 0. \end{aligned}$$

Note that $py < py + \frac{1}{2}\|p\|^2 = (z - y)y + \frac{1}{2}\big[\|z - y\|^2\big] = \alpha$. Assume for contradiction, there exists $x \in C$ such that $px \geq \alpha > py$. Hence, $p(x - y) > 0$. Define

$$\delta = \frac{2p(x - y)}{\|(x - y)\|^2} > 0. \tag{2.1}$$

Now, choose $1 \geq \lambda > 0$ and $\lambda < \delta$. Such a λ exists because of inequality 2.1. Define $w = \lambda x + (1 - \lambda)y$. Since C is convex, w belongs to C. Now,

$$\begin{aligned} \|(z - w)\|^2 &= \|(z - y) + \lambda(y - x)\|^2 \\ &= \|p - \lambda(x - y)\|^2 \\ &= \|p\|^2 - 2\lambda(x - y)p + \lambda^2\|(x - y)\|^2 \\ &< \|p\|^2 \\ &= \|(z - y)\|^2. \end{aligned}$$

Hence, w nearer to z than y. This is a contradiction. Hence, for all $x \in C$ we should have $px < \alpha$. ■

There are other generalizations of this theorem, which we will not cover here. For example, if you drop the requirement that the convex set be closed, then we will get weak separation. The separation theorems have a wide variety of applications.

2.3 Farkas Lemma

We will now state the most important result of this section. The result is called **Farkas Lemma** or **Theorem of Alternatives**.

Suppose $A \in \mathbb{R}^{m \times n}$. Let $(a_1, \ldots, a_n)$ be the columns of A. The set of all non-negative linear combinations of columns of A is called the **cone** of A. Formally,

$$cone(A) = \{b \in \mathbb{R}^m : Ax = b \text{ for some } x \in \mathbb{R}^n_+\}$$

As an example, consider the following 2×3 matrix,

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -2 & -2 \end{bmatrix}$$

If we take $x = (2, 3, 1)$, then we get the following $b \in cone(A)$.

$$b = \begin{bmatrix} 4 + 0 + 1 = 5 \\ 2 + (-6) + (-2) = -6 \end{bmatrix}$$

The $cone(A)$ is depicted in Figure 2.6. As can be seen from the figure, $cone(A)$ is a convex and closed set.

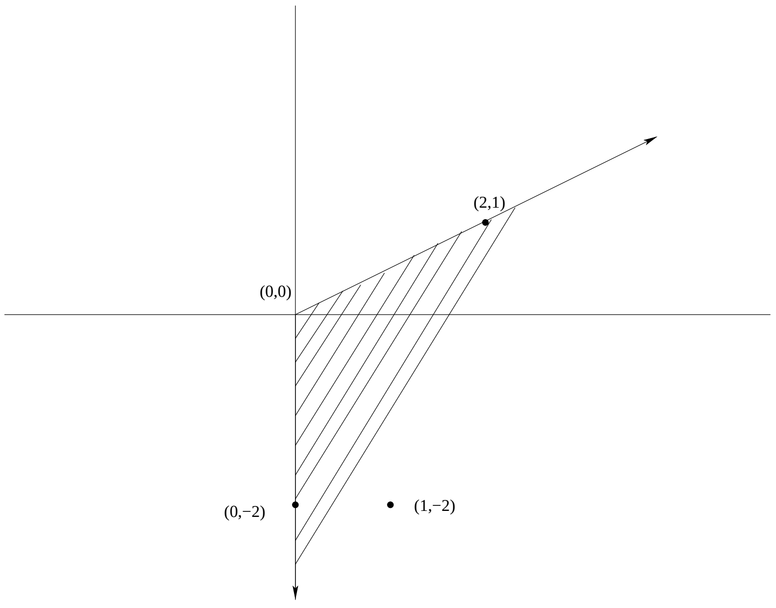


Figure 2.6: Illustration of cone in $\mathbb{R}^2$

Lemma **18** *Suppose $A \in \mathbb{R}^{m \times n}$. Then $cone(A)$ is a convex set.*

Proof: Let $b^1, b^2 \in cone(A)$. Define $b = \lambda b^1 + (1-\lambda)b^2$ for some $\lambda \in (0,1)$. Let $b^1 = Ax^1$ and $b^2 = Ax^2$ for $x^1, x^2 \in \mathbb{R}^n_+$. Then $b = \lambda Ax^1 + (1-\lambda)Ax^2 = A\big[\lambda x^1 + (1-\lambda)x^2\big]$. Since $\mathbb{R}^n_+$ is convex $x = \lambda x^1 + (1-\lambda)x^2 \in \mathbb{R}^n_+$. So, $b \in cone(A)$, and hence, $cone(A)$ is convex. ■

We state the following result without a proof.

LEMMA **19** *Suppose $A \in \mathbb{R}^{m\times n}$. Then $cone(A)$ is a closed set.*

THEOREM **23** *Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Suppose $F = \{x \in \mathbb{R}^n_+ : Ax = b\}$ and $G = \{y \in \mathbb{R}^m : yA \geq 0, yb < 0\}$. Then, $F \neq \emptyset$ if and only if $G = \emptyset$.*

Before we describe the proof, a word about the notation used in Theorem 23. The inequality $yA \geq 0$ is a system of n inequalities. It means that $ya^j \geq 0$ for every column vector a^j of A. The multiplication yA is a matrix multiplication, where appropriate transpose needs to be taken. We have abused notation to write yA to mean this matrix multiplication.

The system of inequalities in G is called the **Farkas alternative**. Let us apply Farkas Lemma to some examples. Does the following system have a non-negative solution?

$$\begin{bmatrix} 4 & 1 & -5 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The Farkas alternative for this system is:

$$\begin{aligned} y_1 + y_2 &< 0 \\ 4y_1 + y_2 &\geq 0 \\ y_1 &\geq 0 \\ -5y_1 + 2y_2 &\geq 0. \end{aligned}$$

The last three inequalities can be written in matrix form as follows.

$$\begin{bmatrix} 4 & 1 \\ 1 & 0 \\ -5 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Since $y_1 \geq 0$, from the first inequality, we get $y_2 < 0$. This contradicts the last inequality. Hence, Farkas alternative have no solution, implying that the original system of equations have a solution.

Proof: Suppose $F \neq \emptyset$. Let $x \in F$. Choose $y \in \mathbb{R}^m$ such that $yA \geq 0$. Then, $yb = y(Ax) = (yA)x \geq 0$. Hence, $G = \emptyset$.

Suppose $G = \emptyset$, and assume for contradiction $F = \emptyset$. This means $\{x \in \mathbb{R}^n_+ : Ax = b\} = \emptyset$, i.e., $b \notin cone(A)$. Since $cone(A)$ is closed and convex, we can separate it from b by a hyperplane (y, α) such that $yb < \alpha$ and $yz > \alpha$ for all $z \in cone(A)$. Notice that $0 \in cone(A)$. Hence, $\alpha < 0$. So, we get $yb < 0$.

Let a^j be a column vector of A. We will show that $ya^j \geq 0$. Consider any $\lambda > 0$. By definition $\lambda a^j \in cone(A)$. Hence, $y(\lambda a^j) > \alpha$. Assume for contradiction $ya^j < 0$. Since $\lambda > 0$ can be chosen arbitrarily large, we can choose it such that $y(\lambda a^j) < \alpha$. This is a contradiction. Hence, $ya^j \geq 0$. Thus, we get $yA \geq 0$ and $yb < \alpha < 0$. Hence $G \neq \emptyset$. ■

Does the following system have a non-negative solution?

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

The farkas alternative is:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$2y_1 + 2y_2 + 2y_3 + y_4 < 0.$$

One possible solution to the Farkas alternative is $y_1 = y_2 = y_3 = \frac{-1}{2}$ and $y_4 = 1$. Hence, the original system has no non-negative solution. Intuitively, it says if we multiply $\frac{-1}{2}$ to the first three equations in the original system and multiply 1 to the last equation, and add all of them up, we will get a contradiction: $0 = -2$.

Often we come across systems of equations and inequalities, with variables that are **free** (no non-negative constraints) and variables that are constrained to be non-negative. Farkas Lemma can be easily generalized to such systems.

Theorem **24** *Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{m \times t}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{k \times n}$, $D \in \mathbb{R}^{k \times t}$, and $d \in \mathbb{R}^k$. Suppose $F = \{x \in \mathbb{R}^n_+, x' \in \mathbb{R}^t : Ax + Bx' = b, Cx + Dx' \leq d\}$ and $G = \{y \in \mathbb{R}^m, y' \in \mathbb{R}^k_+ : yA + y'C \geq 0, yB + y'D = 0, yb + y'd < 0\}$. Then, $F \neq \emptyset$ if and only if $G = \emptyset$.*

Proof: Consider the system of inequalities $Cx + Dx' \leq d$. This can be converted to a system of equations by introducing **slack** variables for every inequality. In particular, consider variables $s \in \mathbb{R}^k_+$ such that $Cx + Dx' + s = d$.

Now, the vectors $x' \in \mathbb{R}^t$ can be written as $x' = x^+ - x^-$, where $x^+, x^- \in \mathbb{R}^t_+$. This is because every real number can be written as difference of two non-negative real numbers. So, the set F can be rewritten as $F = \{x \in \mathbb{R}^n_+, x^+, x^- \in \mathbb{R}^t_+, s \in \mathbb{R}^k_+ : Ax + Bx^+ - Bx^- + 0.s = b, Cx + Dx^+ - Dx^- + Is = d\}$, where I is the identity matrix. In matrix form, this looks as follows:

$$\begin{bmatrix} A & B & -B & 0 \\ C & D & -D & I \end{bmatrix} \begin{bmatrix} x \\ x^+ \\ x^- \\ s \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

The Farkas alternative for this system of equations has two sets of variables $y \in \mathbb{R}^m$ and $y' \in \mathbb{R}^k$. One inequality is $by + dy' < 0$. The other inequalities are $Ay + Cy \geq 0$, $By + Dy' \geq 0$, $-By - Dy' \geq 0$, and $y' \geq 0$. This simplifies to $Ay + Cy' \geq 0$ and $By + Dy' = 0$ with $y' \in \mathbb{R}^k_+$. From Farkas Lemma (Theorem 23), the result now follows. ■

Here is an example of how to write Farkas alternative for general system of constraints. Consider the following set of constraints.

$$\begin{aligned} x_1 - 3x_2 + x_3 &\leq -3 \\ x_1 + x_2 - x_3 &\geq 2 \\ x_1 + 2x_2 + 3x_3 &= 5 \\ x_1 - x_2 &= 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Here, x_3 is a *free* variable (without the non-negativity constraint). The first step is to conver the set of constraints into the form in Theorem 24. For this, we need to convert the second constraint into $\leq$ form.

$$\begin{aligned} x_1 - 3x_2 + x_3 &\leq -3 \\ -x_1 - x_2 + x_3 &\leq -2 \\ x_1 + 2x_2 + 3x_3 &= 5 \\ x_1 - x_2 &= 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

For writing the Farkas alternative, we first associate a variable with every constraint: $y = (y_1, y_2, y_3, y_4)$ for four constraints. Out of this, first and second constraints are inequalities, so corresponding variables (y_1, y_2) are non-negative, while variables (y_3, y_4) corresponding to equations are free.

Now, the strict inequality in the Farkas alternative is easy to write: $-3y_1 - 2y_2 + 5y_3 + 2y_4 < 0$. There will be four constraints in the Farkas alternative, each corresponding to the variables in the original system. The constraints corresponding to non-negative variables are weak inequalities, while the constraints corresponding to free variables are equations. For example, the inequality corresponding to variable x_1 is $y_1 - y_2 + y_3 + y_4 \geq 0$. Similarly, the inequality corresponding to x_2 is $-3y_1 - y_2 + 2y_3 - y_4 \geq 0$. Since the variable x_3 is free, the constraint corresponding to it is an equation: $y_1 + y_2 + 3y_3 = 0$. Hence, the Farkas alternative is:

$$\begin{aligned}
-3y_1 - 2y_2 + 5y_3 + 2y_4 &< 0 \\
y_1 - y_2 + y_3 + y_4 &\geq 0 \\
-3y_1 - y_2 + 2y_3 - y_4 &\geq 0 \\
y_1 + y_2 + 3y_3 &= 0 \\
y_1, y_2 &\geq 0.
\end{aligned}$$

2.4 Application: Core of Cooperative Games

A cooperative game (with transferrable utility) is defined by a set of n players N and a value function $v : 2^N \rightarrow \mathbb{R}$ which represents the value or worth of a coalition (subset) of players. For every coalition $S \subseteq N$, a value $v(S)$ is attached. The exact method of finding the value function depends on the problems. The tuple (N, v) defines a cooperative game. We had already defined cooperative games using cost function. Here is an example with value function.

Sale of an item. Consider the sharing of an item between two buyers and a seller (who owns the item). The set of players can be denoted as $N = \{1, 2, s\}$, where s is the seller. The valuation of the item to buyers (i.e., the utility a buyer gets by getting the item) are: 5 and 10. The seller has no value for the item. The cooperative game can be defined as follows: $v(\emptyset) = v(\{s\}) = 0$; $v(\{1\}) = v(\{2\}) = v(\{1, 2\}) = 0$; $v(\{1, s\}) = 5, v(\{2, s\}) = v(\{1, 2, s\}) = 10$ (by assigning the item to the highest valued buyer).

The definition of a cooperative game says nothing about how the value of a game should be divided between players. The cooperative game literature deals with such issues in details.

A vector $x \in \mathbb{R}^n$ is called an **imputation** if $\sum_{i \in N} x_i = v(N)$ and $x_i \geq v(\{i\})$. One can think of an imputation as a division of $v(N)$ that gives every player at least as much as he will get himself. When we generalize this to every coalition of agents, we get the notion of *core*.

Definition **16** *The* **core** *of a game* (N, v) *is the set* $C(N, v) = \Big\{x \in \mathbb{R}^n : \sum_{i \in N} x_i = v(N),\ \ \sum_{i \in S} x_i \geq v(S)\ \forall\ S \subsetneq N\Big\}$.

The core constraints are stability condition. It stipulates that every coalition of agents must get at least as they will get if they form their own coalition. Otherwise, such a coalition may break from the grand coalition.

In the example above, the set of inequalities for core are

$$\begin{aligned} x_1 + x_2 + x_s &= 10 \\ x_1 + x_2 &\geq 0 \\ x_1 + x_s &\geq 5 \\ x_2 + x_s &\geq 10 \\ x_i &\geq 0\ \forall\ i \in \{1, 2, s\} \end{aligned}$$

Simple substitutions give, $x_2 \leq 5$ and $x_1 \leq 0$. $x_1 \geq 0$ gives $x_1 = 0$, and thus $x_s \geq 5$. So, $x_1 = 0, 0 \leq x_2 \leq 5$, and $5 \leq x_s \leq 10$ with $x_2 + x_s = 10$ constitutes the core of this game.

But not all games have a core. For example, consider a game with two agents $\{1, 2\}$ with $v(\{1\}) = 1 = v(\{2\})$ but $v(\{1, 2\}) = 3$. This game has an empty core since no (x_1, x_2) can satisfy $x_1 \leq 1, x_2 \leq 1$, and $x_1 + x_2 = 3$.

A necessary and sufficient condition can be found by using Farkas Lemma.

Let $B(N)$ be the set of feasible solutions to the following system:

$$\begin{aligned} \sum_{S \subsetneq N : i \in S} y_S &= 1, \qquad \forall\ i \in N, \\ y_S &\geq 0, \qquad \forall\ S \subsetneq N. \end{aligned}$$

y_S can be thought as the weight given to coalition S. It is easy to verify that $B(N) \neq \emptyset$. For example, by setting $y_S = 1$ for all S with $|S| = 1$ and setting $y_S = 0$ otherwise gives a feasible $y \in B(N)$.

Theorem **25 (Bondareva-Shapley)** $C(N, v) \neq \emptyset$ *if and only if*

$$v(N) \geq \sum_{S \subsetneq N} v(S) y_S, \qquad \forall\ y \in B(N).$$

(If a game satisfies this condition then it is called a **balanced** *game, i.e., the core of a game is non-empty if and only if it is balanced).*

Proof: Consider the following system of constraints corresponding to core of a game:

$$\begin{aligned}
&\sum_{i\in N} x_i = v(N) && && \textbf{(CORE)}\\
&\sum_{i\in S} x_i \geq v(S) && \forall\, S \subsetneq N\\
&x_i \text{ free} && \forall\, i \in N.
\end{aligned}$$

The Farkas alternative for (**CORE**) is

$$\begin{aligned}
v(N)y_N - \sum_{S\subsetneq N} v(S)y_S &< 0 && && \textbf{(BAL)}\\
y_N - \sum_{S\subsetneq N: i\in S} y_S &= 0 && \forall\, i \in N\\
y_S &\geq 0 && \forall\, S \subsetneq N\\
y_N &\text{ free.}
\end{aligned}$$

Now, suppose $C(N, v) \neq \emptyset$. Then (**CORE**) has a solution, and Farkas alternative (**BAL**) has no solution. Now, consider any $y \in B(N)$. For any $y \in B(N)$, we let $y_N = 1$ and the final two constraints of (**BAL**) are satisfied. Since (**BAL**) has no solution, we must have $v(N) - \sum_{S\subsetneq N} v(S)y_S \geq 0$. This implies that the game is balanced.

Now, suppose the game is balanced. Then for all $y \in B(N)$, we have $v(N) \geq \sum_{S\subsetneq N} v(S)y_S$. We will show that (**BAL**) has no solution, and hence, (**CORE**) has a solution and $C(N, v) \neq \emptyset$. Assume for contradiction, (**BAL**) has a solution y. Note that since $y_S \geq 0$ for all $S \subsetneq N$, $y_N = \sum_{S\subsetneq N: i\in S} y_S \geq 0$ for all $i \in N$. Hence, $y_N \geq 0$. Further $y_N > 0$ since if $y_N = 0$, then $y_S = 0$ for all $S \subsetneq N$, and this will contradict the first inequality in (**BAL**).

Now, define, $y'_S = \frac{y_S}{y_N}$ for all $S \subseteq N$. Since y is a solution to (**BAL**), we get

$$\begin{aligned}
v(N) &< \sum_{S\subsetneq N} v(S)y'_S\\
\sum_{S\subsetneq N: i\in S} y'_S &= 1 && \forall\, i \in N\\
y'_S &\geq 0 && \forall\, S \subsetneq N.
\end{aligned}$$

Hence, $y' \in B(N)$. But the first inequality contradicts the fact that the game is balanced. ■

Let us verify that the game in our earlier example (of sale of an item) is balanced. Notice that in that game the only coalitions, besides the grand coalition, having positive values are

$\{1, s\}$ and $\{2, s\}$. So, we need to show for every $y \in B(N)$, we have

$$v(\{1,2,s\}) = 10 \geq v(\{1,s\})y_{\{1,s\}} + v(\{2,s\})y_{\{2,s\}} = 5y_{\{1,s\}} + 10y_{\{2,s\}}.$$

But $y \in B(N)$ implies that $y_{\{1,s\}} + y_{\{2,s\}} \leq 1$. Since $y_{\{2,s\}} \leq 1$, we get the desired balanced game.

Indeed, it is easy to state a result for general cooperative "market game". A market game is defined by a seller s and a set of buyers B. So, the set of players is $N = B \cup \{s\}$. The key feature of the market game is the special type of value function. In particular, for every $S \subseteq N$ we define $v(S)$ to be the value of the market with player set (coalition) S. The restriction we put is $v(S) = 0$ if $s \notin S$. Now, call a market game **monotonic** if $v(S) \leq v(T)$ for all $S \subseteq T \subseteq N$.

THEOREM **26** *Every monotonic market game is balanced.*

Proof: Pick any $y \in B(N)$. Now,

$$\begin{aligned}\sum_{S \subsetneq N} v(S)y_S &= \sum_{S \subsetneq N : s \in S} v(S)y_S \\ &\leq v(N) \sum_{S \subsetneq N : s \in S} y_S,\end{aligned}$$

where the inequality uses the fact that market game is monotonic. But since $y \in B(N)$, we get that $\sum_{S \subsetneq N : s \in S} y_S = 1$. This shows that $\sum_{S \subsetneq N} v(S)y_S \leq v(N)$. So, the monotonic market game is balanced. ■

Indeed, a trivial element in the core of a monotonic market game is $x_s = v(N)$ and $x_i = 0$ if $i \neq s$.

Note on cost games. If the cooperative game is defined by a cost function c instead of a value function v, then the core constraints change in the following manner:

$$\begin{aligned}&\sum_{i \in N} x_i = c(N) \\ &\sum_{i \in S} x_i \leq c(S) \qquad \forall\ S \subsetneq N \\ &x_i \text{ free} \qquad \forall\ i \in N.\end{aligned}$$

It is easy to verify that the corresponding Farkas alternative gives the following balancedness condition.

$$\sum_{S \subsetneq N} c(S)y_S \geq c(N) \qquad \forall\ y \in B(N).$$

2.5 Carathéodory Theorem

This section is devoted to an improtant result in the theory of convex sets. The result says that if we choose any point in the convex hull of an arbitrary set $S \subseteq \mathbb{R}^n$, it can be expressed as convex combination of at most $n+1$ points in S. To prove this result, we start with reviewing some basic definitions in linear algebra.

For any set of points $x_1, \ldots, x_k \in \mathbb{R}^n$ and $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$ the point $\sum_{i=1}^k \lambda_i x_i$ is called

- a **linear combination** of $x_1, \ldots, x_k$,
- an **affine combination** of $x_1, \ldots, x_k$ if $\sum_{i=1}^k \lambda_i = 1$,
- a **convex combination** of $x_1, \ldots, x_k$ if $\sum_{i=1}^k \lambda_i = 1$ and $\lambda_1, \ldots, \lambda_k \geq 0$.

Note that if $x \in \mathbb{R}^n$ is a convex combination of points $x_1, \ldots, x_k \in \mathbb{R}^n$ then it is also an affine combination and linear combination of these points. Similarly, if $x \in \mathbb{R}^n$ is an affine combination of $x_1, \ldots, x_k \in \mathbb{R}^n$ then it is also a linear combination of these points.

A set of points $x_1, \ldots, x_k \in \mathbb{R}^n$ are **linearly independent** if none of them can be expressed as a linear combination of others. In other words, if $x_1, \ldots, x_k$ are linearly independent then $\sum_{i=1}^k \lambda_i x_i = 0$ implies that $\lambda_i = 0$ for all $i \in \{1, \ldots, k\}$. If $x_1, \ldots, x_k$ are not linearly independent then they called **linearly dependent**.

Here are some examples of linearly independent vectors.

$$\begin{aligned} &(4, -90) \\ &(1,0,5), (2,5,2) \\ &(1,4), (3,-3). \end{aligned}$$

Here are some examples of linearly dependent vectors.

$$\begin{aligned} &(1,-2), (-2,4) \\ &(0,1,0), (2,-3,5), (2,-2,5). \end{aligned}$$

Similarly, we can define the notion of affine independence. A set of points $x_1, \ldots, x_k \in \mathbb{R}^n$ are **affinely independent** if none of them can be expressed as affine combination of others. A set of points $x_1, \ldots, x_k \in \mathbb{R}^n$ are **affinely dependent** if they are not affinely independent. The alternate definition of affine independence is here.

Lemma **20** *A set of point $x_1, \ldots, x_k \in \mathbb{R}^n$ are affinely independent if and only if $x_2 - x_1, x_3 - x_1, \ldots, x_k - x_1$ are linearly independent.*

Proof: Suppose $x_1, \ldots, x_k$ are affinely independent. Assume for contradiction $x_2 - x_1, x_3 - x_1, \ldots, x_k - x_1$ are linearly dependent. Then, there exists $\lambda_2, \ldots, \lambda_k$, not all of them zero, and $x_j - x_1$ such that such that

$$\sum_{i=2, i\neq j}^{k} \lambda_i (x_i - x_1) = x_j - x_1.$$

This implies that

$$\sum_{i=2, i\neq j}^{k} \lambda_i x_i + [1 - \sum_{i=1, i\neq j}^{k} \lambda_i] x_1 = x_j.$$

This shows that $x_1, \ldots, x_k$ are affinely dependent. This is a contradiction.

Now, suppose $x_2 - x_1, x_3 - x_1, \ldots, x_k - x_1$ are linearly independent. Assume for contradiction $x_1, \ldots, x_k$ are affinely dependent. Then, some point, say x_j, can be expressed as affine combination of others. Hence, for some λs with $\sum_{i=1, i\neq j}^{k} \lambda_i = 1$

$$\begin{aligned} x_j &= \sum_{i=1, i\neq j}^{k} \lambda_i x_i \\ \Rightarrow x_j - x_1 &= \sum_{i=1, i\neq j}^{k} \lambda_i x_i - \sum_{i=1, i\neq j}^{k} \lambda_i x_1 = \sum_{i=2, i\neq j}^{k} \lambda_i [x_i - x_1]. \end{aligned}$$

This is a contradiction to the fact that $x_2 - x_1, \ldots, x_k - x_1$ are linearly independent. ■

This shows that if $x_1, \ldots, x_k$ are affinely dependent then $x_2 - x_1, \ldots, x_k - x_1$ must be linearly dependent. The maximum number of linearly independent points in $\mathbb{R}^n$ is n. We state this as a lemma.

Lemma **21** *The maximum number of linearly independent points in $\mathbb{R}^n$ is n.*

Proof: The proof is left as an exercise. It has to do with solving a system of n equations with n variables. ■

The idea in the previous lemma is extended to any arbitrary set to define the dimension of a set. Consider any set S. The maximum number of linearly independent points in S is called the **dimension** of S. In that sense, the dimension of $\mathbb{R}^n$ is n.

Theorem **27 (Carathéodory Theorem)** *Let $S \subseteq \mathbb{R}^n$ be an arbitrary set. If $x \in H(S)$, then there exists $x_1, \ldots, x_{n+1} \in S$ such that $x \in H(\{x_1, \ldots, x_{n+1}\})$.*

In words, any point in the convex hull of a set in $\mathbb{R}^n$ can be written as the convex combination of at most $n+1$ points in that set.

Proof: Pick $x \in H(S)$. Let $x = \sum_{i=1}^k \lambda_i x_i$ with $\lambda_i > 0$ for all $i \in \{1, \dots, k\}$ and $\sum_{i=1}^k \lambda_i = 1$. If $k \leq n+1$, then we are done. Suppose $k > n+1$. Then, $x_2 - x_1, \dots, x_k - x_1$ are more than n points in $\mathbb{R}^n$. Hence, they are linearly dependent. Hence, there exists $\mu_2, \dots, \mu_k$, not all of them equal to zero, such that $\sum_{i=2}^k \mu_i(x_i - x_1) = 0$. Let $\mu_1 = -\sum_{i=2}^k \mu_i$. Thus, $\sum_{i=1}^k \mu_i x_i = 0$ with $\sum_{i=1}^k \mu_i = 0$ and at least one μ_i positive. Thus for any $\alpha > 0$ we have

$$x = \sum_{i=1}^k \lambda_i x_i = \sum_{i=1}^k \lambda_i x_i - \alpha \sum_{i=1}^k \mu_i x_i = \sum_{i=1}^k (\lambda_i - \alpha\mu_i) x_i.$$

Choose α as follows:

$$\alpha = \min_{1 \leq i \leq k} \{\frac{\lambda_i}{\mu_i} : \mu_i > 0\} = \frac{\lambda_j}{\mu_j}.$$

Note that $\alpha > 0$ since $\lambda_j > 0$. Further, for any $i \in \{1, \dots, k\}$, $\lambda_i - \alpha\mu_i > 0$ if $\mu_i < 0$ and if $\mu_i > 0$, then $\lambda_i - \alpha\mu_i = \lambda_i - \lambda_j \frac{\mu_i}{\mu_j} \geq 0$. Also, note that $\sum_{i=1}^k (\lambda_i - \alpha\mu_i) = 1$. Hence, x is a convex combination of $x_1, \dots, x_k$ but with $\lambda_j - \alpha\mu_j = 0$. Hence, x can be expressed as convex combination of $k-1$ points in S. The process can be repeated till we have $n+1$ points. ■

A consequence of Carathéodory Theorem are the following results.

LEMMA **22** *Suppose $C \subseteq \mathbb{R}^n$. Then, $H(C)$ is the smallest convex set containing C, i.e., suppose S is a convex set such that $C \subseteq S$, then $H(C) \subseteq S$.*

Proof: Consider any convex set S such that $C \subseteq S$. We will show that every point in $H(C)$ which is a convex combination of k points in C belongs to S. We use induction on k. By Caratheéodory Theorem, $k \leq n+1$.

The claim holds for $k = 1$ since $C \subseteq S$. Suppose claim holds for $k = m$. We show that the claim holds for $k = m+1$. Let $x^1, \dots, x^{m+1} \in C$ and $x = \sum_{i=1}^{m+1} \lambda_i x^i$, with usual conditions for convex combination. So, $x \in H(C)$. Now, set $\mu = \sum_{i=1}^m \lambda_m$. By our assumption $\mu > 0$ and $\lambda_{m+1} = 1 - \mu$. So, $x = \mu\big[\sum_{i=1}^m \frac{\lambda_i}{\mu} x^i\big] + (1-\mu)x^{m+1}$. Now, $\sum_{i=1}^m \frac{\lambda_i}{\mu} x^i$ is convex combination of m points in C, and hence must belong to S by our induction hypothesis. Further, $x^{m+1} \in C \subseteq S$. Thus, x is a convex combination of two points in S. Since S is convex, x must belong to S. ■

LEMMA **23** *The convex hull of a compact set is a compact set.*

Proof: Let S be a compact set. Since S is bounded, $H(S)$ is also bounded. Now, take a sequence $\{x^k\}$ in $H(S)$. By Carathéodory theorem, each x^k can be represented as convex combination of $n+1$ points in S. This gives a sequence in αs and xs, where α belongs to a bounded set and $x \in S$ belongs to a bounded set. Hence, both of them must have limit points. Since S is closed, the result follows. ■

Chapter 3

Linear Programming

3.1 Introduction

Optimization of a function f over a set S involves finding the maximum (minimum) value of f (objective function) in the set S (feasible set). Properties of f and S define various types of optimization. Primarily, optimization can be classified into three categories.

1. **Linear Programming**: If f is a linear function (e.g., $f(x_1, x_2) = x_1+2x_2$) and the set S is defined by a finite collection of linear inequalities and equalities, then it is called a **linear program**. As an example, consider the following linear program.

$$
\begin{aligned}
\max_{x_1,x_2} f(x_1, x_2) = \max_{x_1,x_2}&[x_1 + 2x_2] \\
\text{s.t.}& \\
x_1 + x_2 &\leq 6 \\
x_1 &\geq 2 \\
x_2 &\geq 0
\end{aligned}
$$

2. **Integer Programming**: An integer program is a linear program with further restriction that the solution be integers. In the previous example, if we impose that x_1 and x_2 can only admit integer values, then it becomes an integer program.

3. **Nonlinear Programming**: If f is a non-linear function and the feasible set S is defined by a finite collection of non-linear equations, then it is called a **non-linear program**. There are further classifications (and extensions) of non-linear programs

depending on the specific nature of the problem. Typically, f is assumed to be continuous and differentiable. An example is the following non-linear program.

$$\begin{aligned}
\max_{x_1,x_2} f(x_1,x_2) = \max_{x_1,x_2}&[-2x_1^2 - 3x_2^2] \\
\text{s.t.}& \\
x_1 + x_2 &= 1.
\end{aligned}$$

In general, an optimization problem written in mathematical form is referred to as a **mathematical program**. In this course, we will learn about **linear and integer programming problems, their solution methods**.

To understand a little more about linear and integer programs, consider the above example. The feasible set/region can be drawn on a plane. Figure 3.1 shows the feasible regions for the linear program (dashed region), and the integer points inside that feasible region is the feasible region of the integer program. Notice that the optimal solution of this linear program has to lie on the boundary of the feasible region. Moreover, an *extreme* point is an optimal solution ($x_1 = 2,\ x_2 = 4$). This is no accident, as we will show. If we impose the integer constraints on x_1 and x_2, then the feasible region has a finite set of points. Again, the optimal solution is $x_1 = 2,\ x_2 = 4$ (this is obvious since this is an integral solution, and is an optimal solution of the linear program).

3.2 Steps in Solving an Optimization Problem

There are some logical steps to solve an optimization problem.

- **Modeling**: It involves reading the problem carefully to decipher the variables of the problem. Then, one needs to write down the objective and the constraints of the problem in terms of the variables. This defines the objective function and the feasible set, and hence the mathematical program. This process of writing down the mathematical program from a verbal description of the problem is called **modeling**. Modeling, though it does not give the solution, is an important aspect of optimization. A good modeling helps in getting solutions faster.

- **Solving**: Once the problem is modeled, the solution is sought. There are algorithms (techniques) to solve mathematical programs. Commercial software companies have come up with *solvers* that have built packages using these algorithms.

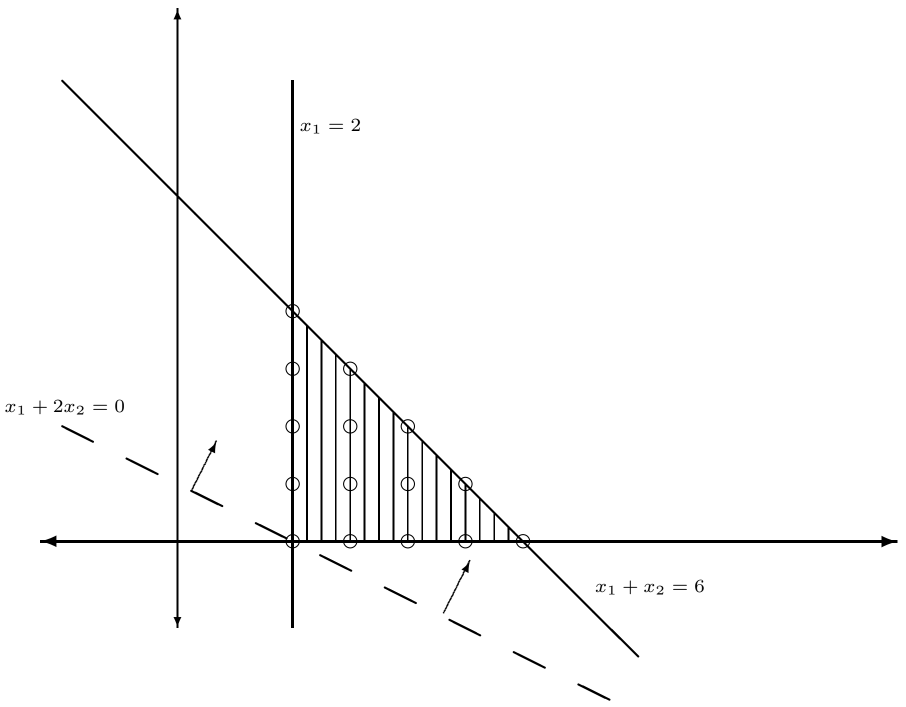


Figure 3.1: Feasible set and objective function of linear and integer programs

3.3 Linear Programming

3.3.1 An Example

Example 1 *There is 100 units of water to be distributed among three villages. The water requirement of the villages are 30, 50, and 40 respectively. The water shortage costs of the three villages are 4, 3, and 5 respectively. Water supply to no two villages should exceed 70. Find a water distribution that minimizes the total water shortage cost.*

Modeling: Let x_i ($i \in \{1, 2, 3\}$) denote the amount of water supplied to village i. Since the total amount of water is 100, we immediately have

$$x_1 + x_2 + x_3 = 100.$$

Further, water supply of no two villages should exceed 70. This gives us,

$$\begin{aligned} x_1 + x_2 &\leq 70 \\ x_2 + x_3 &\leq 70 \\ x_1 + x_3 &\leq 70. \end{aligned}$$

The water requirement of every village puts an upper bound on the supply. So, we can put $x_1 \leq 30,\ x_2 \leq 50,\ x_3 \leq 40$. Of course, the water supply should all be non-negative, i.e., $x_1, x_2, x_3 \geq 0$. Finally, the total water shortage costs of the three villages are $4(30 - x_1) + 3(50 - x_2) + 5(40 - x_3) = 470 - 4x_1 - 3x_2 - 5x_3$. If we want to minimize the total water shortage cost, then it is equivalent to just maximizing $4x_1 + 3x_2 + 5x_3$. So, the problem can be **formulated** as:

$$
\begin{aligned}
Z = \max_{x_1,x_2,x_3}\ & 4x_1 + 3x_2 + 5x_3 \\
\text{s.t.} \qquad & \\
\sum_{i=1}^{3} x_i &= 100 \\
x_i + x_j &\leq 70 \qquad \forall\ i,j \in \{1,2,3\},\ i \neq j \\
x_1 &\leq 30 \\
x_2 &\leq 50 \\
x_3 &\leq 40 \\
x_i &\geq 0 \qquad i \in \{1,2,3\}
\end{aligned}
\tag{\textbf{P1}}
$$

Problems of this type are called linear programming formulations.

3.3.2 Standard Form

In general, if $c_1, \ldots, c_n$ are real numbers, then the function f of real variables $x_1, \ldots, x_n$ ($\mathbf{x} = (x_1, \ldots, x_n)$) defined by

$$f(\mathbf{x}) = c_1x_1 + \ldots + c_nx_n = \sum_{j=1}^{n} c_jx_j$$

is called a **linear function**. If g is a linear function and b is a real number then

$$g(\mathbf{x}) = b$$

is called a **linear equation**, whereas

$$g(\mathbf{x}) \leq (\geq) b$$

is called a **linear inequality**. A **linear constraint** is one that is either a linear equation or a linear inequality. A **linear programming (LP)** problem is one which maximizes (minimizes) a linear function subject to (s.t.) a finite collection of linear constraints. Formally, any LP can be written in the following form:

$$
\begin{aligned}
Z = \max_{\mathbf{x}} \sum_{j=1}^{n} c_j x_j & \\
\text{s.t.} & \qquad\qquad (\mathbf{LP})\\
\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad & \forall\, i \in \{1, \dots, m\} \\
x_j \ge 0 \qquad & \forall\, j \in \{1, \dots, n\}.
\end{aligned}
$$

In matrix notation, this can be written as $\max_x cx$ subject to $Ax \le b$ and $x \ge 0$. Problems in the form (**LP**) will be referred to as the problems in **standard form**. As we have seen any LP problem can be converted to a problem in standard form. The key difference between any LP problem not in standard form and a problem in standard form is that the constraints in standard form are all inequalities (written in a particular way). Also, the last collection of constraints say that variable have to be non-negative. This type of inequalities are special, and referred to as **non-negativity constraints**. The linear function that is to be maximized or minimized is called the **objective function**. In the standard form, the objective function will always be maximized (this is only our notation).

If $(x_1^*, \dots, x_n^*)$ satisfy all the constraints of (**LP**), then it is called a **feasible solution** of (**LP**). For example, in the problem (**P1**), a feasible solution is $(x_1, x_2, x_3) = (30, 35, 35)$. A feasible solution that gives the maximum value to the objective function amongst all feasible solutions is called an **optimal solution**, and the corresponding value of the objective function is called the **optimal value**. The optimal solution of (**LP**) is $(x_1, x_2, x_3) = (30, 30, 40)$, and the optimal value is 410 (hence the minimum total water shortage cost is 60).

Not every LP problem has an optimal solution. As we will show later, every LP problem can be put in one of the following three categories.

1. **Optimal solution exists**: This is the class of LP problems whose optimal solution exists. An example is (**P1**).

2. **Infeasible**: This is the class of LP problems for which no feasible solution exists. An

example is the following:

$$
\begin{aligned}
Z = \max_{x_1,x_2} & \; x_1 + 5x_2 \\
\text{s.t.} & \qquad\qquad (\textbf{INF-LP}) \\
x_1 + x_2 &\leq 3 \\
-3x_1 - 3x_2 &\leq -11 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

3. **Unbounded**: This is the class of LP problems for which feasible solutions exist, but for every number M, there exists a feasible solution that gives the objective function a value more than M. So, none of the feasible solutions is optimal. An example is the following:

$$
\begin{aligned}
Z = \max_{x_1,x_2} & \; x_1 - x_2 \\
\text{s.t.} & \qquad\qquad (\textbf{UNBD-LP}) \\
-2x_1 + x_2 &\leq -1 \\
-x_1 - 2x_2 &\leq -2 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

   To understand why (**UNBD-LP**) is unbounded, it is useful to look at its feasible region and objective function in a figure. Figure 3.2 shows how the objective function can increase indefinitely.

3.4 History of Linear Programming

The second world war brought about many new things to the world. This included the use and rapid growth of the field of linear programming. In 1947, **George B. Dantzig**, regarded by many as the founder of the discipline, designed the **simplex method** to solve linear programming problems for the U.S. Air Force. After that, the field showed rapid growth in terms of research. Production management and economics were the primary areas which applied linear programming in a variety of problems. Tremendous application potential of this field led to increase in theoretical research in the field, and thus, a new branch of applied mathematics was born.

In 1947, T.C. Koopmans pointed out several applications of linear programming in economic theory. Till today, a significant portion of economic theory is still governed by the fundamentals of linear programming (as we will discuss).

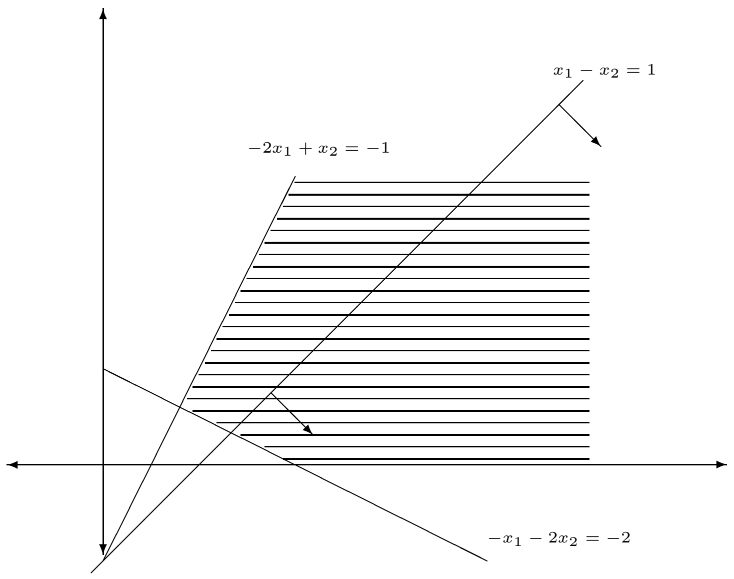


Figure 3.2: Unbounded LP

The fundamentals of linear programming has its roots as early as 1820s, when Fourier investigated techniques to solve systems of linear equations. L.V. Kantorovich pointed out the significance of linear programming and its applications in 1939. Unfortunately, his work was not noticed for a long time (since he carried out most of his research in U.S.S.R.). In 1975, Kantorovich and Koopmans were awarded the Nobel prize in economics "for their contributions to the theory of optimum allocation of resources". Another event in 1970s attracted a lot of media attention. Ever since the invention of simplex method, mathematicians were working for a *theoretically* satisfactory algorithm to solve linear programs. In 1979, L.G. Khachian published the description of such an algorithm - though its performance has been extremely unsatisfactory in practice. On the other hand, simplex method, whose theortical performance is not good, does a very good job in practice.

3.5 Simplex Preview

One of the first discovered, and immensely effective linear programming (LP) algorithms is the **simplex method**. The objective of this section is to give examples to illustrate the method.

3.5.1 First Example

Consider the following example.

$$
\begin{aligned}
& Z = \max 5x_1 + 4x_2 + 3x_3 \\
\text{s.t.} & \qquad\qquad\qquad\qquad\qquad\qquad\qquad (\textbf{EX-1}) \\
& 2x_1 + 3x_2 + x_3 \le 5 \qquad\qquad\qquad\qquad (3.1) \\
& 4x_1 + x_2 + 2x_3 \le 11 \qquad\qquad\qquad\qquad (3.2) \\
& 3x_1 + 4x_2 + 2x_3 \le 8 \qquad\qquad\qquad\qquad (3.3) \\
& x_1, x_2, x_3 \ge 0.
\end{aligned}
$$

The first step in the method consists of introducing **slack variables** for every constraint. For example, in Equation 3.1, the *slack* between 5 and $2x_1 + 3x_2 + x_3$ is assigned a slack variable x_4, i.e., $x_4 = 5 - 2x_1 - 3x_2 - x_3$. Notice that $x_4 \ge 0$. Thus, the equations in formulation (**EX-1**) can be rewritten using slack variables as

$$
\begin{aligned}
x_4 &= 5 - 2x_1 - 3x_2 - x_3 \\
x_5 &= 11 - 4x_1 - x_2 - 2x_3 \\
x_6 &= 8 - 3x_1 - 4x_2 - 2x_3 \\
x_1, x_2, x_3, x_4, x_5, x_6 &\ge 0.
\end{aligned}
$$

The new variables x_4, x_5, x_6 are called **slack variables**, and the old variables x_1, x_2, x_3 are called **decision variables**. Hence our new LP is to

$$\max z \qquad \text{s.t.} \qquad x_1, x_2, x_3, x_4, x_5, x_6 \ge 0, \tag{3.4}$$

where $z = 5x_1 + 4x_2 + 3x_3$ and x_4, x_5, and x_6 are determined by the equations above. This new LP is equivalent (same set of feasible solutions in terms of decision variables) to (**EX-1**), given the equations determining the slack variables. The simplex method is an iterative procedure in which having found a feasible solution $x_1, \ldots, x_6$ of (3.4), we look for another feasible solution $\bar{x}_1, \ldots, \bar{x}_6$ of (3.4) such that

$$5\bar{x}_1 + 4\bar{x}_2 + 3\bar{x}_3 > 5x_1 + 4x_2 + 3x_3.$$

If an optimal solution exists, we can repeat this finite number of iterations till there is no improvement in the objective function value, at which point we stop. The first step is to find a feasible solution, which is easy in our example: $x_1 = x_2 = x_3 = 0$, which gives $x_4 = 5, x_5 = 11, x_6 = 8$. This gives $z = 0$.

We now need to look for a solution that gives a higher value to z. For this, we look to increase values of any one of the variables x_1, x_2, x_3. We choose x_1. Keeping x_2 and x_3 at zero, we notice that we can increase x_1 to $\min(\frac{5}{2}, \frac{11}{4}, \frac{8}{3}) = \frac{5}{2}$ to maintain $x_4, x_5, x_6 \geq 0$. As a result of this, the new solution is

$$x_1 = \frac{5}{2}, x_2 = 0, x_3 = 0, x_4 = 0, x_5 = 1, x_6 = \frac{1}{2}, z = \frac{25}{2}.$$

Notice that by increasing the value of x_1, a variable whose value was positive (x_4) got a value of zero. Now, we have to create system of equation similar to previous iteration. For that we will write the value of z and variables having non-zero values (x_1, x_5, x_6) in terms of variables having zero values (x_4, x_2, x_3).

$$\begin{aligned}
x_1 &= \frac{5}{2} - \frac{3}{2}x_2 - \frac{1}{2}x_3 - \frac{1}{2}x_4. \\
x_5 &= 1 + 5x_2 + 2x_4. \\
x_6 &= \frac{1}{2} + \frac{1}{2}x_2 - \frac{1}{2}x_3 + \frac{3}{2}x_4.
\end{aligned}$$

$$z = \frac{25}{2} - \frac{7}{2}x_2 + \frac{1}{2}x_3 - \frac{5}{2}x_4.$$

Of x_2, x_3, x_4, the value of z decreases by increasing the values of x_2 and x_4. So, the only candidate for increasing value in this iteration is x_3. The amount we can increase the value of x_3 can again be obtained from the feasibility conditions of $x_1, x_5, x_6 \geq 0$, which is equivalent to (given $x_2 = x_4 = 0$) $\frac{5}{2} - \frac{1}{2}x_3 \geq 0$ and $\frac{1}{2} - \frac{1}{2}x_3 \geq 0$. This gives that the maximum possible value of x_3 in this iteration can be $\min(5, 1) = 1$. By setting $x_3 = 1$, we get a new solution as

$$x_1 = 2, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, z = 13. \tag{3.5}$$

Two things should be noticed here: (a) this solution is also a solution of the previous system of equations and (b) the earlier solution is also a solution of this system of equations. This is precisely because we are just rewriting the system of equations using a different set of decision and slack variables in every iteration, and that is the central theme of the simplex method.

So, the new variable that takes zero value is x_6. We now write the system of equations in terms of x_2, x_4, x_6.

$$\begin{aligned} x_3 &= 1 + x_2 + 3x_4 - 2x_6 \\ x_1 &= 2 - 2x_2 - 2x_4 + x_6 \\ x_5 &= 1 + 5x_2 + 2x_4 \end{aligned}$$

$$z = 13 - 3x_2 - x_4 - x_6.$$

Now, the value of z will decrease by increasing the values of any of the variables x_2, x_4, x_6. So, we have reached a dead-end. In fact, we have reached an optimal solution. This is clear from the fact that any solution requires $x_2, x_4, x_6 \geq 0$, and by assigning any value not equal to zero to these variables, we will decrease the value of z. Hence, $z = 13$ is an optimal solution. The corresponding values of x_1, x_3, x_5 are $2, 1, 1$ respectively.

3.5.2 Dictionaries

Consider a general LP in standard form:

$$\begin{aligned} & Z = \max \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} & \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall\, i \in \{1, \dots, m\} \\ & x_j \geq 0 \qquad \forall\, j \in \{1, \dots, n\} \end{aligned} \tag{\textbf{LP}}$$

The first step in the simplex method is to introduce slack variables, $x_{n+1}, \dots, x_{n+m} \geq 0$ corresponding to m constraints, and denote the objective function as z. So,

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j \qquad \forall\, i \in \{1, \dots, m\} \tag{3.6}$$

$$z = \sum_{j=1}^{n} c_j x_j. \tag{3.7}$$

$$x_j \geq 0 \qquad \forall\, j \in \{1, \dots, n, n+1, \dots, n+m\} \tag{3.8}$$

In simplex method, we search for a feasible solution $\bar{x}_1, \ldots, \bar{x}_{m+n}$ given a feasible solution $x_1, \ldots, x_{m+n}$ so that the objective function is better, i.e.,

$$\sum_{j=1}^{n} c_j \bar{x}_j > \sum_{j=1}^{n} c_j x_j.$$

As we have seen in the example, a feasible solution is represented with a system of linear equations consisting of *dependent* variables. These system of equations corresponding to a feasible solution is called a **dictionary**. A dictionary will have the following features:

1. Every solution of the system of equations of the dictionary must be a solution of system of equations (3.6), (3.7), and (3.8), and vice versa.
2. The equations of every dictionary must express m of the variables $x_1, \ldots, x_{m+n}$ and the objective function z (dependent variables) in terms of the remaining n variables (independent variables).

Consider the following starting dictionary.

$$\begin{aligned} x_3 &= 5 - x_2 + x_1 \\ x_4 &= 3 - x_2 - 2x_1 \\ z &= x_1 + 3x_2 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

In this dictionary, we can set $x_1 = x_2 = 0$ to get a feasible solution. Rewriting the first equation in terms of x_2, we get the following dictionary.

$$\begin{aligned} x_2 &= 5 + x_1 - x_3 \\ x_4 &= -2 - 3x_1 + x_3 \\ z &= 15 + 4x_1 - 3x_3 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

Unlike the first dictionary, we cannot put the value of independent variables to zero to get a feasible solution: putting $x_1 = x_3 = 0$ gives us $x_2 = 5$ but $x_4 = -2 < 0$. This is an undesirable feature. To get over this feature, we need the following notion.

In the dictionary, the dependent variables are kept on the left hand side (LHS), and they are expressed in terms of the independent variables on the right hand side (RHS). An additional feature of a dictionary is

- setting the RHS variables at zero and evaluating the LHS variables, we arrive at a feasible solution.

A dictionary with this additional property is called a **feasible dictionary**. Hence, every feasible dictionary describes a feasible solution. The second dictionary above is not feasible but the first one is.

A feasible solution that can be described in terms of a feasible dictionary is called a **basic solution**. The characteristics feature of the simplex method is that it works with basic solutions only.

3.5.3 Second Example

We conclude the discussion by giving another example.

$$Z = \max 5x_1 + 5x_2 + 3x_3$$

$$\begin{aligned} &\text{s.t.} \\ x_1 + 3x_2 + x_3 &\leq 3 \\ -x_1 + 3x_3 &\leq 2 \\ 2x_1 - x_2 + 2x_3 &\leq 4 \\ 2x_1 + 3x_2 - x_3 &\leq 2 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

In this case, the initial feasible dictionary has all the slack variables as dependent variables, and it looks as follows:

$$\begin{aligned} x_4 &= 3 - x_1 - 3x_2 - x_3 \\ x_5 &= 2 + x_1 - 3x_3 \\ x_6 &= 4 - 2x_1 + x_2 - 2x_3 \\ x_7 &= 2 - 2x_1 - 3x_2 + x_3 \end{aligned}$$

$$z = 5x_1 + 5x_2 + 3x_3.$$

The feasible dictionary describes the following solution:

$$x_1 = x_2 = x_3 = 0, x_4 = 3, x_5 = 2, x_6 = 4, x_7 = 2.$$

As before, we try to increase the value of z by increasing the value of one of the independent variables as much as we can. Right now, since x_1, x_2, x_3 have all positive coefficients in the z equation, we randomly choose x_1. From feasibility of $x_4, \ldots, x_7 \geq 0$, we get $x_1 \leq 1$ to be the most stringent constraint. So, we make $x_1 = 1$, which in turn makes $x_7 = 0$. We now write the new dictionary with x_1 leaving the independent variables and x_7 entering the independent variables. First, substitute,

$$x_1 = 1 - \frac{3}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_7.$$

Substituting for x_1 in terms of new set of independent variables in the previous dictionary, we get

$$\begin{aligned}
x_1 &= 1 - \frac{3}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_7 \\
x_4 &= 2 - \frac{3}{2}x_2 - \frac{3}{2}x_3 + \frac{1}{2}x_7 \\
x_5 &= 3 - \frac{3}{2}x_2 - \frac{5}{2}x_3 - \frac{1}{2}x_7 \\
x_6 &= 2 + 4x_2 - 3x_3 + x_7
\end{aligned}$$

$$z = 5 - \frac{5}{2}x_2 + \frac{11}{2}x_3 - \frac{5}{2}x_7.$$

Some comments about terminology are in order:

1. Dependent variables, which appear on the LHS of any dictionary, are called **basic variables**. Independent variables are called **non-basic variables**. In the previous dictionary, x_1, x_4, x_5, x_6 are basic variables, and x_2, x_3, x_7 are non-basic variables.

2. Set of basic and non-basic variables change from iteration to iteration.

3. Choice of **entering** basic variable is motivated by the fact that we want to increase the value of z, and we choose one that does that, and increase its value the maximum possible.

4. Choice of **leaving** basic variable is motivated by the need to maintain feasibility. This is done by identifying the basic variable that poses the most stringent bound on the entering basic variable.

5. The formula for the entering basic variable appears in the **pivot row**, and the process of constructing a new dictionary is called **pivoting**. In the previous dictionary, x_3 is the next entering basic variable, and x_6 is the leaving basic variable. So, the formula for x_3 appears in

$$x_3 = \frac{2}{3} + \frac{4}{3}x_2 - \frac{1}{3}x_6 + \frac{1}{3}x_7,$$

which is the pivot row.

Continuing with our example, the clear choice of entering basic variable is x_3. Calculations give that x_6 imposes the most stringent bound on x_3, and should be the leaving basic variable. So, we arrive at the new dictionary.

$$\begin{aligned}
x_3 &= \frac{2}{3} + \frac{4}{3}x_2 + \frac{1}{3}x_7 - \frac{1}{3}x_6 \\
x_1 &= \frac{4}{3} - \frac{5}{6}x_2 - \frac{1}{3}x_7 - \frac{1}{6}x_6 \\
x_4 &= 1 - \frac{7}{2}x_2 + \frac{1}{2}x_6 \\
x_5 &= \frac{4}{3} - \frac{29}{6}x_2 - \frac{4}{3}x_7 + \frac{5}{6}x_6 \\
\\
z &= \frac{26}{3} + \frac{29}{6}x_2 - \frac{2}{3}x_7 - \frac{11}{6}x_6.
\end{aligned}$$

Now, the entering basic variable is x_2, and the leaving basic variable is x_5. Pivoting yields the following dictionary:

$$x_2 = \frac{8}{29} - \frac{8}{29}x_7 + \frac{5}{29}x_6 + \frac{6}{29}x_5$$

$$\begin{aligned}
x_3 &= \frac{30}{29} + \frac{1}{29}x_7 - \frac{3}{29}x_6 - \frac{8}{29}x_5 \\
x_1 &= \frac{32}{29} - \frac{3}{29}x_7 - \frac{9}{29}x_6 + \frac{5}{29}x_5 \\
x_4 &= \frac{1}{29} + \frac{28}{29}x_7 - \frac{3}{29}x_6 + \frac{21}{29}x_5
\end{aligned}$$

$$z = 10 - 2x_7 - x_6 - x_5.$$

At this point, no more pivoting is possible, and we arrive at the optimal solution described by the last dictionary as:

$$x_1 = \frac{32}{29}, x_2 = \frac{8}{29}, x_3 = \frac{30}{29},$$

and this yields an optimal value of $z = 10$.

3.6 Pitfalls and How to Avoid Them

Three kind of pitfalls can occur in simplex method:

1. **Initialization**: We may not be able to start. We may not have a feasible dictionary to start.
2. **Iteration**: We may get stuck in some iteration. Can we always choose a new entering and leaving variable?
3. **Termination**: We may not be able to finish. Can the simplex method construct an endless sequence of dictionaries without reaching an optimal solution?

We look at each of these three pitfalls. Before proceeding, let us review the general form of a dictionary. There is a set of basic variables B with $\#B = m$, and the linear program is written in the following form in this dictionary.

$$\begin{aligned} x_i &= \bar{b}_i - \sum_{j \notin B} \bar{a}_{ij} x_j \qquad \forall\ i \in B \\ z &= \bar{v} + \sum_{j \notin B} \bar{c}_j x_j \end{aligned}$$

Here, for each $i \in B$, $\bar{b}_i \geq 0$ if the dictionary is a feasible dictionary.

3.6.1 Iteration

3.6.1.1 Choosing an Entering Variable

The entering variable is a *non-basic variable with a positive coefficient $\bar{c}_j$ in the last row of the current dictionary.* This rule is ambiguous in the sense that it may provide more than one candidate for entering or no candidate at all.

The latter alternative implies that the current dictionary has an optimal solution. This is because any solution which is not the current solution will involve some current non-basic variable taking on positive value. Since all the $\bar{c}_j$s are negative, this will imply objective function value decreasing from the current value.

If there are more than one candidate for entering the basis, then any of these candidates may serve.

3.6.1.2 Finding a Leaving Variable

The leaving variable is that *basic variable whose non-negativity imposes the most stringent upper bound on the increase of the entering variable.* Again, there may be more than one candidate or no candidate at all.

If there are more than one candidate, then we may choose any one of them.

If there are no candidate at all, then an interesting conclusion can be drawn. Recall that a linear program is **unbounded** if for every real number M there exists a feasible solution of the linear program such that the objective function value is larger than M.

Here is an example of a dictionary:

$$\begin{aligned} x_2 &= 5 + 2x_3 - x_4 - 3x_1 \\ x_5 &= 7 - 3x_4 - 4x_1 \end{aligned}$$

$$z = 5 + x_3 - x_4 - x_1.$$

The entering variable is x_3. However, neither of the two basic variables x_2 and x_5 put an upper bound on x_3. Hence, we can increase x_3 as much as we want without violating feasibility. Set $x_3 = t$ for any positive number t, and we get the solution $x_1 = 0, x_2 = 5 + 2t, x_3 = t, x_4 = 0, x_5 = 7$, and $z = 5 + t$. Since t can be made arbitrarily large, so can be z, and we conclude that the problem is unbounded. The same conclusion can be reached in general: if there is no candidate for leaving the basis, then we can make the value of the entering variable, and hence the value of the objective function, as large as we wish. In that case, the problem is unbounded.

3.6.1.3 Degeneracy

The presence of more than one candidate for leaving the basis has interesting consequences. For example, consider the dictionary

$$\begin{aligned} x_4 &= 1 - 2x_3 \\ x_5 &= 3 - 2x_1 + 4x_2 - 6x_3 \\ x_6 &= 2 + x_1 - 3x_2 - 4x_3 \end{aligned}$$

$$z = 2x_1 - x_2 + 8x_3.$$

Having chosen x_3 as the entering variable, we see that x_4, x_5, and x_6 are all candidates for leaving variable. Choosing x_4, and pivoting, we get the new dictionary as

$$\begin{aligned} x_3 &= 0.5 - 0.5x_4 \\ x_5 &= -2x_1 + 4x_2 + 3x_4 \\ x_6 &= x_1 - 3x_2 + 2x_4 \end{aligned}$$

$$z = 4 + 2x_1 - x_2 - 4x_4.$$

This dictionary is different from others in one important aspect: along with the non-basic variables, two of the basic variables, x_5 and x_6 have value of zero. Basic solutions with one or more basic variables at zero are called **degenerate**.

Although harmless, degeneracy has annoying side effects. In the next iteration, we have x_1 as the entering variable, and x_5 as the leaving variable. But the value of x_1 can be increased by a maximum of zero. Hence, the objective function value does not change. Pivoting changes the dictionary to:

$$\begin{aligned} x_1 &= 2x_2 + 1.5x_4 - 0.5x_5 \\ x_3 &= 0.5 - 0.5x_4 \\ x_6 &= -x_2 + 3.5x_4 - 0.5x_5 \end{aligned}$$

$$z = 4 + 3x_2 - x_4 - x_5.$$

but the solution remains the same. Simplex iterations that do not change the basic solution are called **degenerate**. One can verify that the next iteration is also degenerate, but the one after that is not - in fact, it is the optimal solution.

Degeneracy is an accident. Many practical problems face degeneracy, and when it happens the simplex goes through few (many a times quite a few) degenerate iterations before coming up with a non-degenerate solution. But there are occasions when this may not happen.

3.6.2 Cycling

Sometimes, a sequence of dictionary can appear again and again. This phenomenon is called **cycling**. To understand cycling let us look at a series of dictionaries.

$$\begin{aligned}
x_5 &= -0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \\
x_6 &= -0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \\
x_7 &= 1 - x_1
\end{aligned}$$

$$z = 10x_1 - 57x_2 - 9x_3 - 24x_4.$$

The following rule for selecting the entering and leaving variable is the following:

- The entering variable will always be the nonbasic variable that the largest coefficient in the z-row of the dictionary.
- If two or more basic variables compete for leaving the basis, then the candidate with the smallest subscript will be made to leave.

Now, the sequence of dictionaries constructed in the first six iterations goes as follows. After the first iteration:

$$\begin{aligned}
x_1 &= 11x_2 + 5x_3 - 18x_4 - 2x_5 \\
x_6 &= -4x_2 - 2x_3 + 8x_4 + x_5 \\
x_7 &= 1 - 11x_2 - 5x_3 + 18x_4 + 2x_5
\end{aligned}$$

$$z = 53x_2 + 41x_3 - 20x_4 - 20x_5.$$

After the second iteration:

$$\begin{aligned} x_2 &= -0.5x_3 + 2x_4 + 0.25x_5 - 0.25x_6 \\ x_1 &= -0.5x_3 + 4x_4 + 0.75x_5 - 2.75x_6 \\ x_7 &= 1 + 0.5x_3 - 4x_4 - 0.75x_5 - 13.25x_6 \end{aligned}$$

$$z = 14.5x_3 - 98x_4 - 6.75x_5 - 13.25x_6.$$

After the third iteration:

$$\begin{aligned} x_3 &= 8x_4 + 1.5x_5 - 5.5x_6 - 2x_1 \\ x_2 &= -2x_4 - 0.5x_5 + 2.5x_6 + x_1 \\ x_7 &= 1 - x_1 \end{aligned}$$

$$z = 18x_4 + 15x_5 - 93x_6 - 29x_1.$$

After the fourth iteration:

$$\begin{aligned} x_4 &= -0.25x_5 + 1.25x_6 + 0.5x_1 - 0.5x_2 \\ x_3 &= -0.5x_5 + 4.5x_6 + 2x_1 - 4x_2 \\ x_7 &= 1 - x_1 \end{aligned}$$

$$z = 10.5x_5 - 70.5x_6 - 20x_1 - 9x_2.$$

After the fifth iteration:

$$\begin{aligned} x_5 &= 9x_6 + 4x_1 - 8x_2 - 2x_3 \\ x_4 &= -x_6 - 0.5x_1 + 1.5x_2 + 0.5x_3 \\ x_7 &= 1 - x_1 \end{aligned}$$

$$z = 24x_6 + 22x_1 - 93x_2 - 21x_3.$$

After the sixth iteration:

$$\begin{aligned}
x_5 &= -0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \\
x_6 &= -0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \\
x_7 &= 1 - x_1
\end{aligned}$$

$$z = 10x_1 - 57x_2 - 9x_3 - 24x_4.$$

Since the dictionary after the sixth iteration is identical with the initial dictionary, the simplex method will go through the same set of dictionaries again and again without ever finding the optimal solution (which is $z = 1$ in this example).

Notice that cycling means, we have a series of degenerate solutions, else we will have increase in objective function, and cannot have the same dictionaries repeating. It is important to note that **cycling implies that we get the same solution in every iteration, even though the set of basic variables change**. It is not possible that we are changing the value of some variable without changing the objective function value (because we always choose an entering variable that changes the objective function value when its value is changed).

THEOREM **28** *If the simplex method fails to terminate, then it must cycle.*

Proof: There are a total of $m+n$ variables. Since in every iteration of the simplex method we choose m basic variables, there are finite number of ways to choose them. Hence, if the simplex method does not terminate, then there will be two dictionaries with the same set of basic variables. Represent the two dictionaries as:

$$x_i = b_i - \sum_{j \notin B} a_{ij} x_j \qquad \forall\ i \in B$$

$$z = v + \sum_{j \notin B} c_j x_j.$$

and

$$x_i = b_i^* - \sum_{j \notin B} a_{ij}^* x_j \qquad \forall\ i \in B$$

$$z = v^* + \sum_{j \notin B} c_j^* x_j.$$

with the same set of basic variables $x_i (i \in B)$.

But there is a unique way of representing a (basic) variable in terms of a set of non-basic variable. Hence the two dictionaries must be exactly equal. ■

Cycling is a rare phenomena, but sometimes they do occur in practice. In fact, constructing an LP problem on which the simplex method may cycle is difficult. It is known that if the simplex method cycles off-optimum on a problem that has an optimal solution, then the dictionaries must involve at least six variables and at least three equations. In practice, cycling occurs very rarely.

Two popular rules for avoiding cycling are: (a) perturbation method and lexicographic ordering (b) smallest subscript rule. We describe the smallest subscript rule here. The former requires extra computation to choose the entering and leaving variables while the latter leaves no choice in the hands of users to choose entering variables, which we can get in the former one.

To avoid cycling, we introduce a rule called (Bland's) **smallest subscript rule**. This refers to breaking ties in the choice of the entering and leaving variables by always choosing the candidate x_k that has the smallest subscript k.

THEOREM **29** *The simplex method terminates as long as the entering and leaving variables are selected by the smallest subscript rule (SSR) in each iteration.*

Proof: By virtue of previous theorem, we need to show that cycling is impossible when the SSR is used. Assume for contradiction that the simplex method with SSR generates a sequence of dictionaries $D_0, D_1, \ldots, D_k$ such that $D_k = D_0$.

Call a variable **fickle** if it is nonbasic in some iteration and basic in some other. Among all fickle variables, let x_t have the largest subscript. Due to cycling, there is a dictionary D in the sequence $D_0, \ldots, D_k$ with x_t leaving (basic in D but nonbasic in the next dictionary), and some other fickle variable x_s entering (nonbasic in D but basic in the next iteration). Further along in the sequence $D_0, D_1, \ldots, D_k, D_1, \ldots, D_k$, there is a dictionary D^* with x_t entering. Let us record D as

$$x_i = b_i - \sum_{j \notin B} a_{ij} x_j \qquad \forall\, i \in B$$

$$z = v + \sum_{j \notin B} c_j x_j.$$

Since all iterations leading from D to D^* are degenerate, the objective function z must have the same value v in both D and D^*. Thus, the last row of D^* may be recorded as

$$z = v + \sum_{j=1}^{m+n} c_j^* x_j,$$

where $c_j^* = 0$ wherever x_j is basic in D^*. Since this equation has been obtained from D by algebraic manipulations, it must satisfy every solution of D. In particular, it must be satisfied by

$$x_s = y, x_j = 0 \ (j \notin B, j \neq s), x_i = b_i - a_{is}y \ (i \in B), z = v + c_s y \qquad \forall\ y.$$

Thus, we have

$$v + c_s y = v + c_s^* y + \sum_{i \in B} c_i^*(b_i - a_{is}y).$$

and, after simplification,

$$\Big(c_s - c_s^* + \sum_{i \in B} c_i^* a_{is}\Big) y = \sum_{i \in B} c_i^* b_i$$

for every choice of y. Since the RHS of the previous equation is a constant independent of y, we have

$$c_s - c_s^* + \sum_{i \in B} c_i^* a_{is} = 0.$$

But x_s is entering in D, implying $c_s > 0$. Since x_s is not entering in D^* and yet $s < t$, we have $c_s^* \leq 0$. Hence for some $r \in B$, $c_r^* a_{rs} < 0$. Note two points now:

- Since $r \in B$, the variable x_r is basic in D; since $c_r^* \neq 0$, the same variable is nonbasic in D^*. Hence, x^r is fickle, and we have $r \leq t$.
- $r \neq t$: since x_t is leaving in D, we have $a_{ts} > 0$ and so $c_t^* a_{ts} > 0$ (since $c_t^* > 0$ with x_t entering in D^*).

This shows that $r < t$ and yet x_r is not entering in D^*. Thus, $c_r^* \leq 0$, and $a_{rs} > 0$. Since all iterations from D to D^* are degenerate, the two dictionaries describe the same solution. Since x_r is non-basic in D^*, its value is zero in both D and D^*, meaning $b_r = 0$. Hence, x_r was a candidate for leaving the basis of D - yet we picked x_t, even though $r < t$. This is a contradiction. ■

3.6.3 Initialization

The only remaining point that needs to be explained is getting hold of the initial feasible dictionary in a problem

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j x_j \\ &\text{s.t.} \\ &\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall\, i \in \{1, \dots, m\} \\ &x_j \geq 0 \qquad \forall\, j \in \{1, \dots, n\}. \end{aligned}$$

with an infeasible origin. The problem with infeasible origin is that we may not know whether a feasible solution exists at all, and even we know what a feasible dictionary will be for that solution. One way of getting around these two problems is by the so called **auxiliary problem**:

$$\begin{aligned} &\min x_0 \\ &\text{s.t.} \\ &\sum_{j=1}^{n} a_{ij} x_j - x_0 \leq b_i \qquad \forall\, i \in \{1, \dots, m\} \\ &x_j \geq 0 \qquad \forall\, j \in \{0, 1, \dots, n\}. \end{aligned}$$

A feasible solution of the auxiliary problem is readily available: set $x_j = 0$ for $j \neq 0$ and make the value of x_0 sufficiently large. Further, the auxiliary problem is not unbounded since $x_0 \geq 0$ and the objective function is $\min x_0$.

Theorem **30** *The original LP problem has a feasible solution if and only if the optimal value of the associated auxiliary problem is zero.*

Proof: If the original problem has a feasible solution than the auxiliary problem has the same feasible solution with $x_0 = 0$. This is clearly the optimal value. Further if the auxiliary problem has a feasible (optimal) solution with $x_0 = 0$, then the original problem has the same feasible solution. ■

Hence, our objective is to solve the auxiliary problem. Consider the following example.

$$\begin{aligned} &\max x_1 - x_2 + x_3 \\ &\text{s.t.} \\ &2x_1 - x_2 + 2x_3 \leq 4 \\ &2x_1 - 3x_2 + x_3 \leq -5 \\ &-x_1 + x_2 - 2x_3 \leq -1 \\ &x_1, x_2, x_3 \geq 0. \end{aligned}$$

To avoid unnecessary confusion, we write the auxiliary problem in its maximization form, and construct the dictionary as

$$\begin{aligned} x_4 &= 4 - 2x_1 + x_2 - 2x_3 + x_0 \\ x_5 &= -5 - 2x_1 + 3x_2 - x_3 + x_0 \\ x_6 &= -1 + x_1 - x_2 + 2x_3 + x_0 \\ \\ w &= -x_0, \end{aligned}$$

which is an infeasible dictionary. But it can be made feasible by pivoting on the most negative b_i row, i.e., x_5 in this case, and choosing x_0 as the entering variable. The new (feasible) dictionary is:

$$\begin{aligned} x_0 &= 5 + 2x_1 - 3x_2 + x_3 + x_5 \\ x_4 &= 9 - 2x_2 - x_3 + x_5 \\ x_6 &= 4 + 3x_1 - 4x_2 + 3x_3 + x_5 \\ \\ z &= -5 - 2x_1 + 3x_2 - x_3 - x_5. \end{aligned}$$

In general, the dictionary corresponding to the auxiliary problem is:

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij}x_j + x_0 \qquad \forall\ i \in \{1, \ldots, m\}$$

$$w = -x_0.$$

which is infeasible. However, this can be transformed into a feasible dictionary. This is done by a single pivot in which x_0 enters and the "most infeasible" x_{n+i} leaves. More precisely, the leaving variable is that x_{n+k} whose b_k has the smallest (negative) value among all negative b_is. After pivoting, x_0 assumes the positive value $-b_k$, and each basic x_{n+i} assumes the non-negative value $b_i - b_k$. Now, we can continue with our simplex method. In our example, two more iterations yield the following dictionary:

$$\begin{aligned}
x_3 &= 1.6 - 0.2x_1 + 0.2x_5 + 0.6x_6 - 0.8x_0 \\
x_2 &= 2.2 + 0.6x_1 + 0.4x_5 + 0.2x_6 - 0.6x_0 \\
x_4 &= 3 - x_1 - x_6 + 2x_0 \\
\\
w &= -x_0.
\end{aligned}$$

This dictionary is an optimal solution of the auxiliary problem with $x_0 = 0$. Further, this points to a feasible dictionary of the original problem.

$$\begin{aligned}
x_3 &= 1.6 - 0.2x_1 + 0.2x_5 + 0.6x_6 \\
x_2 &= 2.2 + 0.6x_1 + 0.4x_5 + 0.2x_6 \\
x_4 &= 3 - x_1 - x_6 \\
\\
z &= -0.6 + 0.2x_1 - 0.2x_5 + 0.4x_6.
\end{aligned}$$

So, we learned how to construct the auxiliary problem, and its first feasible dictionary. In the process of solving the auxiliary problem, **it may be possible that x_0 may be a candidate for the leaving variable in which case we *pick* x_0**. Immediately, after pivoting we get

- x_0 as a non-basic variable, in which case $w = 0$.

Clearly, this is an optimal solution. However, we may also reach an optimal dictionary of auxiliary problem with x_0 basic. If value of w is non-zero in that, then we simply conclude that the original problem is infeasible. Else, x_0 is basic and the optimal value of w is zero. We argue that this is not possible. Since the dictionary preceding the final dictionary was not optimal, the value of $w = -x_0$ must have changed from some negative value to zero in the final iteration. To put it differently, the value of x_0 must have changed from some positive level to zero in this iteration. This means, x_0 was also a candidate for leaving the basis, and we should have picked it according to our policy. This is a contradiction.

Hence, we either construct an optimal solution of the auxiliary problem where x_0 is a non-basic variable, and we proceed to the original problem by constructing a new feasible dictionary, or we conclude that the original problem is infeasible.

This strategy of solving an LP is known as the **two phase simplex method**. In the first phase, we set up and solve the auxiliary problem; if we find an optimal solution of the auxiliary problem, then we proceed to the **second phase**, solving the original problem.

THEOREM **31 (Fundamental Theorem of Linear Programming)** *Every LP problem in the standard form has the following three properties:*

1. *If it has no optimal solution, then it is either unbounded or infeasible.*

2. *If it has a feasible solution, then it has a basic feasible solution.*

3. *If it has an optimal solution, then it has a basic optimal solution.*

Proof: The first phase of the two phase simplex method either discovers that the problem is infeasible or else it delivers a basic feasible solution. The second phase either discovers that the problem is unbounded or gives a basic optimal solution. ■

Note that if the problem is not in standard form then the theorem does not hold, e.g., max x s.t. $x < 0$ has no optimal solution even though it is neither infeasible nor unbounded.

3.6.4 An Example Illustrating Geometry of the Simplex Method

We give an example to illustrate how the simplex method works. Consider the following linear program.

$$\begin{aligned} \max_{x_1,x_2} \; & x_1 + 2x_2 \\ \text{s.t.} & \\ x_1 + x_2 &\leq 4 \\ x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

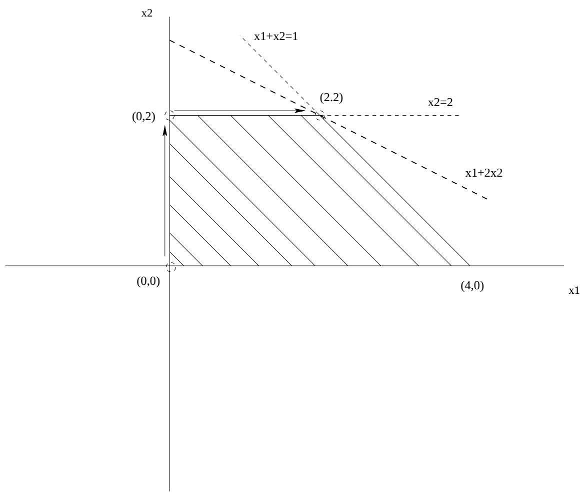


Figure 3.3: Illustrating the Simplex Method

The feasible region for this LP is shown in Figure 3.3. Clearly, no first phase is required here since the origin is a feasible solution. Hence, the first dictionary looks as follows (x_3 and x_4 are the slack variables).

$$x_3 = 4 - x_1 - x_2$$
$$x_4 = 2 - x_2$$

$$z = x_1 + 2x_2.$$

Note that the feasible dictionary gives the solution $x_3 = 4$ and $x_4 = 2$. It shows the amount of slack in the two constraints. The two constraints $x_1 \geq 0$ and $x_2 \geq 0$ are tight. This describes the point $(0, 0)$.

Let us choose x_2 as the entering variable. In that case, the binding constraint is $x_4 = 2 - x_2$. So, x_4 is the leaving variable. Hence, $x_4 = 0$. This means the constraint $x_2 \leq 2$ will now be tight (along with $x_1 \geq 0$). This describes the point $(0, 2)$.

Finally, x_1 is the entering variable, in which case the constraint corresponding to x_3 became tight (along with the constraint corresponding to x_4). This describes the point $(2, 2)$, which is the optimal solution according to the simplex method.

Hence, we go from one *corner* point to the other in the simplex method as shown in Figure 3.3.

3.7 Polyhedra and Polytopes

Polyhedra are special classes of closed convex sets. We have already shown (in assignments) that a closed convex set is characterized by intersection of (possibly infinite) half-spaces. If it is the intersection of a finite number of half-spaces, then it is called a polyhedron.

Definition **17** *A set $P \subseteq \mathbb{R}^n$ is called a* **polyhedron** *if there exists a $m \times n$ matrix A and a vector $b \in \mathbb{R}^m$ such that $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.*

A polytope is the convex hull of a finite set of points.

Definition **18** *A set $P \subset \mathbb{R}^n$ is called a* **polytope** *if there exists finite number of vectors $x^1, \ldots, x^t \in \mathbb{R}^n$ such that $P = H(x^1, \ldots, x^t)$.*

Definition **19** *Let P be a convex set. A point $z \in P$ is called an* **extreme point** *of P if P cannot be expressed as a convex combination of two other points in P, i.e., there do not exist $x, y \in P \setminus \{z\}$ and $0 < \lambda < 1$ such that $z = \lambda x + (1 - \lambda)y$.*

Figure 3.4 shows two polyhedra, out of which the one on the right is a polytope. It also shows some extreme points of these polyhedra.

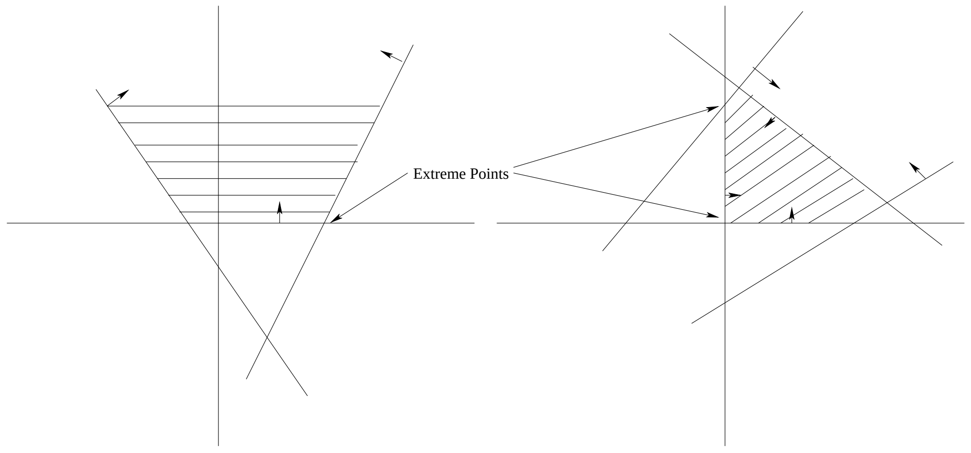


Figure 3.4: A polyhedron, a polytope, and extreme points

We prove a fundamental result characterizing the extreme points of a polyhedron. For this, we use the following notation. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a polyhedron and $z \in P$. Then, A_z denotes the submatrix of A for which $a_i z = b_i$ for every row a_i of A_z.

As an example consider $P = \{(x_1, x_2) : x_1 + 2x_2 \leq 2, -x_1 \leq 0, -x_2 \leq 0\}$. If we let $z = (0, 1)$, then A_z corresponds to a matrix with rows $(1, 2)$ and $(-1, 0)$.

Now, we remind ourselves of some basic definitions of linear algebra.

Definition **20** *The* **rank** *of a finite set S of vectors, denoted as $r(S)$, is the cardinality of the largest subset of linearly independent vectors in S.*

If $S = \{(1,2),(-2,4)\}$, then $r(S) = 1$. If $S = \{(0,1,0),(-2,2,0),(-2,3,0)\}$, then $r(S) = 2$.

Let A be a $m \times n$ matrix. Then the rank of row vectors of A and the rank of column vectors of A are same. So, for a matrix, we can talk about rank of that matrix. We denote rank of matrix A as $r(A)$.

Theorem **32** *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a polyhedron and $z \in P$. Then z is an extreme point of P if and only if $r(A_z) = n$.*

Proof: Suppose z is an extreme point of P. Assume for contradiction $r(A_z) < n$. This means there exists a vector $x \in \mathbb{R}^n$ and $x \neq 0$ such that $A_z x = 0$. By definition, for all rows a_i not in A_z we have $a_i z < b_i$. This means there exists a $\delta > 0$ such that for every a_i not in A_z we have

$$a_i(z + \delta x) \leq b_i \text{ and } a_i(z - \delta x) \leq b_i.$$

To see why this is true, consider the a row a_i not in A_z. Suppose $a_i x \leq 0$. Then $\delta a_i x \leq 0$. This means, $a_i z + \delta a_i x < b_i$ since $a_i z < b_i$. Also, since δ can be chosen arbitrarily small, $a_i z - \delta a_i x \leq b_i$. Analogously, if $a_i x \geq 0$, we will have $a_i z + \delta a_i x \leq b_i$ and $a_i z - \delta a_i x < b_i$. For all rows in A_z, $a_i(z + \delta x) = b_i$ and $a_i(z - \delta x) = b_i$.

So, we get $A(z + \delta x) \leq b$ and $A(z - \delta x) \leq b$. Hence, $z + \delta x$ and $z - \delta x$ belong to P. Since z is a convex combination of these two points, z cannot be an extreme point. This is a contradiction.

Suppose $r(A_z) = n$. Assume for contradiction z is not an extreme point of P. Then there exists $x, y \in P$ with $z \neq x \neq y$ and $0 < \lambda < 1$ such that $z = \lambda x + (1-\lambda)y$. Then for every row a_i in A_z we can write $a_i x \leq b_i = a_i z = a_i(\lambda x + (1-\lambda)y)$. Rearranging, we get $a_i(x - y) \leq 0$. Similarly, $a_i y \leq b_i = a_i z = a_i(\lambda x + (1-\lambda)y)$. This gives us $a_i(x-y) \geq 0$. Hence, $a_i(x-y) = 0$. This implies that $A_z(x-y) = 0$. But $x \neq y$ implies that $r(A_z) \neq n$, which is a contradiction. ■

Remark: This theorem implies that a polyhedron has only a finite number of extreme points. This follows from the fact that there can be only finite number of subrows of A.

Remark: Also, if the number of linearly independent rows (constraints) of A is less than n, then rank of every submatrix of A will be less than n. In that case, the polyhedron has no extreme points - in two dimension, if the constraints are all parallel lines then the rank of any submatrix is one, and clearly we cannot have any extreme point. Hence, if z is an extreme point of P, then we should have more constraints than variables. This is ensured in the standard formulation of the linear program by having n non-negativity constraints, whose row vectors are linearly independent.

REMARK: Suppose z and z' are two distinct extreme points of a polyhedron. Then A_z and $A_{z'}$ are distinct. Else, we will have $A_z(z - z') = 0$, and $z - z' \neq 0$ will imply that $r(A_z) < n$.

The result can be used to prove the following theorem, which we state without proving.

THEOREM **33** *Let P be a bounded polyhedron with extreme points $(x^1, \ldots, x^t)$. Then $P = H(x^1, \ldots, x^t)$, i.e., every bounded polyhedron is a polytope. Moreover, every polytope is a bounded polyhedron.*

3.8 Extreme Points and Simplex Method

We will discuss a fundamental property of a simplex dictionary. Below we describe the constraints of a linear program as $Ax \leq b$, where we fold the non-negativity constraints into them.

THEOREM **34** *A feasible solution described by a feasible dictionary (i.e., a basic feasible solution) of a linear program* $\max cx$ *subject to* $x \in \{x \in \mathbb{R}_n : Ax \leq b\}$ *is an extreme point of* $\{x \in \mathbb{R}_n : Ax \leq b\}$.

Proof: Let z be a feasible solution of max cx s.t. $Ax \leq b$ ($x \geq 0$ is folded into the constraints) described by a feasible dictionary. An implication of this is n non-basic variables have value zero in the dictionary. This implies that the constraints (either non-negativity or the original constraints) corresponding to these non-basic variables are binding, and hence, belong to A_z.

Suppose $r(A_z) < n$. Then, we know that (from earlier proof) that there exists $\delta > 0$ and $z' \neq 0$ with $A_z z' = 0$ such that $z + \delta z'$ is a solution to the linear program. The set of constraints in A_z are also binding for $(z + \delta z')$. Hence, the values of non-basic variables in the dictionary corresponding to z are also zero in $(z + \delta z')$.

But setting, a set of non-basic variables to zero, describes a unique feasible solution to the linear program. This implies that $z' = 0$, a contradiction. Hence, $r(A_z) = n$. From our characterization of extreme points of polyhedron, z is an extreme point of the polyhedron $Ax \leq b$. ■

3.9 A Matrix Description of Dictionary

The objective of this section is to write the dictionary in matrix form. Consider the following dictionary.

$$\begin{aligned}
x_1 &= 54 - 0.5x_2 - 0.5x_4 - 0.5x_5 + 0.5x_6 \\
x_3 &= 63 - 0.5x_2 - 0.5x_4 + 0.5x_5 - 1.5x_6 \\
x_7 &= 15 + 0.5x_2 - 0.5x_4 + 0.5x_5 + 2.5x_6 \\
\\
z &= 1782 - 2.5x_2 + 1.5x_4 - 3.5x_5 - 8.5x_6.
\end{aligned}$$

The dictionary arises after two iterations of simplex method to the following linear program:

$$\begin{aligned}
\max 19x_1 + 13x_2 + 12x_3 + 17x_4 \\
\text{s.t.} \\
3x_1 + 2x_2 + x_3 + 2x_4 &\leq 225 \\
x_1 + x_2 + x_3 + x_4 &\leq 117 \\
4x_1 + 3x_2 + 3x_3 + 4x_4 &\leq 420 \\
x_1, x_2, x_3, x_4 &\geq 0.
\end{aligned} \tag{\textbf{EX}}$$

The slack variables are x_5, x_6, x_7, and they convert the inequalities to equations in the dictionary.

$$\begin{aligned}
3x_1 + 2x_2 + x_3 + 2x_4 + x_5 &= 225 \\
x_1 + x_2 + x_3 + x_4 + x_6 &= 117 \\
4x_1 + 3x_2 + 3x_3 + 4x_4 + x_7 &= 420
\end{aligned} \tag{3.9}$$

The given dictionary is obtained by solving for x_1, x_3, and x_7 from these equations. In matrix terms the solution may be described very compactly. First, we record the system as $Ax = b$, where

$$A = \begin{bmatrix} 3 & 2 & 1 & 2 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 4 & 3 & 3 & 4 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 225 \\ 117 \\ 420 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

To write the system in terms of basic variables x_1, x_3, x_7, we rewrite $Ax = b$ as $A_B x_B + A_N x_N$, where

$$A_B = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix}$$

$$A_N = \begin{bmatrix} 2 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 3 & 4 & 0 & 0 \end{bmatrix}$$

$$x_B = \begin{bmatrix} x_1 \\ x_3 \\ x_7 \end{bmatrix}$$

$$x_N = \begin{bmatrix} x_2 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

So, now we can write the system of equations as

$$A_B x_B = b - A_N x_N$$

If the square matrix A_B is non-singular, we can multiply both sides of the last equation by A_B^{-1} on left to get

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

This is a compact record of the equations in the given dictionary. To write the objective function in matrix terms, we write it as cx with $c = [19, 13, 12, 17, 0, 0, 0]$, or more precisely as $c_B x_B + c_N x_N$, where $c_B = [19, 12, 0]$ and $c_N = [13, 17, 0, 0]$. Substituting for x_B we get

$$z = c_B(A_B^{-1} b - A_B^{-1} A_N x_N) + c_N x_N = c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N.$$

The given dictionary can now be recorded quite compactly as

$$x_B = A_B^{-1}b - A_B^{-1}A_N x_N$$

$$z = c_B(A_B^{-1}b - A_B^{-1}A_N x_N) + c_N x_N = c_B A_B^{-1}b + (c_N - c_B A_B^{-1}A_N)x_N.$$

The only thing that we have not proved so far is that A_B is non-singular. The simplex method chooses B in a way so that A_B is non-singular. This implies that the above matrix description is valid.

3.10 Duality

Duality is probably the most used concept of linear programming in both theory and practice. The central motivation to look for a dual is the following: **How do we find bounds on the objective function of a linear program without solving it completely**? To understand further, let us look at the following example.

$$Z = \max 4x_1 + x_2 + 5x_3 + 3x_4$$

$$\begin{aligned} &\text{s.t.} \\ x_1 - x_2 - x_3 + 3x_4 &\leq 1 \\ 5x_1 + x_2 + 3x_3 + 8x_4 &\leq 55 \\ -x_1 + 2x_2 + 3x_3 - 5x_4 &\leq 3 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

Rather than solving this LP, let us try to find bounds on the optimal value z^* of this LP. For example, $(0,0,0,0)$ is a feasible solution. Hence, $z^* \geq 0$. Another feasible solution is $(0,0,1,0)$ which gives $z^* \geq 5$. Another feasible solution is $(3,0,2,0)$ which gives $z^* \geq 22$. But there is no systematic way in which we were looking for the estimate - it was purely guess work. Duality provides one systematic way of getting this estimate.

Let us multiply the second constraint by $\frac{5}{3}$, which gives us

$$\frac{25}{3}x_1 + \frac{5}{3}x_2 + 5x_3 + 40x_4 \leq \frac{275}{3}.$$

But notice that

$$4x_1 + x_2 + 5x_3 + 3x_4 \leq \frac{25}{3}x_1 + \frac{5}{3}x_2 + 5x_3 + 40x_4 \leq \frac{275}{3}.$$

Hence $z^* \leq \frac{275}{3}$. With a little thinking, we can improve this bound further. In particular, add the second and third constraints to get

$$4x_1 + 3x_2 + 6x_3 + 3x_4 \leq 58.$$

Using the same logic as before, we get $z^* \leq 58$. Here, we are constructing a series of upper bounds for the objective function value, while earlier we were constructing a series of lower bounds.

Formally, we construct linear combinations of the inequalities. We multiply j^{th} constraint by y_j, and add them all up. The resulting inequality reads

$$(y_1 + 5y_2 - y_3)x_1 + (-y_1 + y_2 + 2y_3)x_2 + (-y_1 + 3y_2 + 3y_3)x_3 + (3y_1 + 8y_2 - 5y_3)x_4 \\ \leq y_1 + 55y_2 + 3y_3. \quad (3.10)$$

Of course, each of these multipliers must be non-negative. Next, we want to use the LHS of Equation (3.10) as an upper bound on $4x_1 + x_2 + 5x_3 + 3x_4$. This can be justified only if in (3.10), the coefficient of each x_i is at least as big as the corresponding coefficient in the objective function. More explicitly, we want

$$\begin{aligned} y_1 + 5y_2 - y_3 &\geq 4 \\ -y_1 + y_2 + 2y_3 &\geq 1 \\ -y_1 + 3y_2 + 3y_3 &\geq 5 \\ 3y_1 + 8y_2 - 5y_3 &\geq 3. \end{aligned}$$

If the multipliers are non-negative (note here that if the constraints are equalities, then we do not need the multipliers to be non-negative - they can be free) and if they satisfy these inequalities, then we can get an upper bound on the objective function, i.e., for every feasible solution (x_1, x_2, x_3, x_4) of the original problem and every feasible solution (y_1, y_2, y_3) of the previous set of inequalities, we have

$$4x_1 + x_2 + 5x_3 + 3x_4 \leq y_1 + 55y_2 + 3y_3.$$

Further optimal solution z^* of the original LP satisfies

$$z^* \leq y_1 + 55y_2 + 3y_3.$$

Of course we want this bound to be as close to optimal as possible. This can be done by minimizing $y_1 + 55y_2 + 3y_3$. So, we are led to another LP problem that gives us an upper bound of the original problem.

$$\begin{aligned} \min y_1 + 55y_2 + 3y_3 & \\ \text{s.t.} & \\ y_1 + 5y_2 - y_3 &\geq 4 \\ -y_1 + y_2 + 2y_3 &\geq 1 \\ -y_1 + 3y_2 + 3y_3 &\geq 5 \\ 3y_1 + 8y_2 - 5y_3 &\geq 3 \\ y_1, y_2, y_3 &\geq 0. \end{aligned}$$

3.10.1 Writing Down the Dual

From our discussion of the example, it is clear how to write the dual of an original problem. In general, the **dual problem** of

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j x_j \\ &\text{s.t.} \\ &\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall\, i \in \{1, \ldots, m\} \\ &x_j \geq 0 \qquad \forall\, j \in \{1, \ldots, n\}. \end{aligned} \tag{\textbf{P}}$$

is defined as the problem

$$\begin{aligned} &\min \sum_{i=1}^{m} b_i y_i \\ &\text{s.t.} \\ &\sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad \forall\, j \in \{1, \ldots, n\} \\ &y_i \geq 0 \qquad \forall\, i \in \{1, \ldots, m\}. \end{aligned} \tag{\textbf{D}}$$

Notice the following things in the dual (**D**):

1. For every constraint of (**P**), we have a variable in (**D**).

2. Further, for every variable of (**P**), we have a constraint in (**D**).

3. The coefficient in the objective function of (**P**) appears on the RHS of constraints in (**D**) and the RHS of constraints in (**P**) appear as coefficients of objective function in (**D**).

As an exercise, verify that dual of (**D**) is (**P**).

LEMMA **24 (Weak Duality)** *Let $(x_1, \ldots, x_n)$ be a feasible solution of (**P**) and $(y_1, \ldots, y_m)$ be a feasible solution of (**D**). Then*

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i y_i.$$

Proof:

$$\begin{aligned}\sum_{j=1}^{n} c_j x_j &\leq \sum_{j=1}^{n} (\sum_{i=1}^{m} a_{ij} y_i) x_j \\ &= \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j) y_i \\ &\leq \sum_{i=1}^{m} b_i y_i.\end{aligned}$$

■

Lemma 24 is useful since if we find feasible solutions of (**P**) and (**D**) at which their objective functions are equal, then we can conclude that they are optimal solutions. Indeed, Lemma 24 implies that if $(x_1^*, \ldots, x_n^*)$ is an optimal solution of (**P**) and $(y_1^*, \ldots, y_m^*)$ is an optimal solution of (**D**) such that $\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$, then for every feasible solution $(x_1, \ldots, x_n)$ of (**P**) and for every feasible solution $(y_1, \ldots, y_n)$, we can write

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i y_i^* = \sum_{j=1}^{n} c_j x_j^* \leq \sum_{i=1}^{m} b_i y_i.$$

3.11 THE DUALITY THEOREM

The explicit version of the theorem is due to Gale, but it is supposed to have originated from conversations between Dantzig and von Neumann in the fall of 1947.

Theorem **35 (The Duality Theorem - Strong Duality)** *Let* $(x_1^*, \ldots, x_n^*)$ *be a feasible solution of (**P**) and* $(y_1^*, \ldots, y_m^*)$ *be a feasible solution of (**D**).* $(x_1^*, \ldots, x_n^*)$ *is an optimal solution of (**P**) and* $(y_1^*, \ldots, y_m^*)$ *is an optimal solution of (**D**) if and only if*

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*. \tag{SD}$$

Before presenting the proof, let us illustrate the crucial point of the theorem: the optimal solution of (**D**) can be read off the z-row of the final dictionary for (**P**). For the example, the final dictionary of (**P**) is

$$\begin{aligned} x_2 &= 14 - 2x_1 - 4x_3 - 5x_5 - 3x_7 \\ x_4 &= 5 - x_1 - x_3 - 2x_5 - x_7 \\ x_6 &= 1 + 5x_1 + 9x_3 + 21x_5 + 11x_7 \\ \\ z &= 29 - x_1 - 2x_3 - 11x_5 - 6x_7. \end{aligned}$$

Note that the slack variables x_5, x_6, x_7 can be matched with the dual variables y_1, y_2, y_3 in a natural way. In the z-row of the dictionary, the coefficients of these slack variables are $(-11, 0, -6)$. As it turns out the optimal dual solution is obtained by reversing the signs of these coefficients, i.e., $(11, 0, 6)$. The proof of the duality theorem works on this logic.

Proof: Suppose Equation (**SD**) holds. Assume for contradiction that $(x_1', \ldots, x_n') \neq (x_1^*, \ldots, x_n^*)$ is a feasible solution of (**P**) such that $\sum_{j=1}^{n} c_j x_j' > \sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$. By Lemma 24, this is a contradiction. Hence, $(x_1^*, \ldots, x_n^*)$ is an optimal solution of (**P**). A similar argument shows that $(y_1^*, \ldots, y_m^*)$ is an optimal solution of (**D**).

For the other side of the proof, we assume that $(x_1^*, \ldots, x_n^*)$ is an optimal solution of (**P**), and find a feasible solution $(y_1^*, \ldots, y_m^*)$ that satisfies the claim in the theorem, and such a solution will be optimal by the first part of the proof. In order to find that feasible solution, we solve (**P**) using the simplex method using slack variables

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j \qquad \forall\, i \in \{1, \ldots, m\}.$$

Since an optimal solution exists, the simplex method finds it, and the final row of the final dictionary reads

$$z = z^* + \sum_{k=1}^{m+n} \bar{c}_k x_k,$$

where $\bar{c}_k = 0$ whenever x_k is a basic variable and $\bar{c}_k \le 0$ otherwise. In addition, z^* is the optimal value of (**P**), hence

$$z^* = \sum_{j=1}^{n} c_j x_j^*.$$

We claim that

$$y_i^* = -\bar{c}_{n+i} \qquad \forall\ i \in \{1, \dots, m\}$$

is a feasible solution of (**D**) satisfying the claim of our theorem. Substituting $z = \sum_{j=1}^{n} c_j x_j$ and substituting for slack variables, we get

$$\sum_{j=1}^{n} c_j x_j = z^* + \sum_{j=1}^{n} \bar{c}_j x_j - \sum_{i=1}^{m} y_i^* \Big(b_i - \sum_{j=1}^{n} a_{ij} x_j \Big),$$

which may be rewritten as

$$\sum_{j=1}^{n} c_j x_j = \Big(z^* - \sum_{i=1}^{m} b_i y_i^* \Big) + \sum_{j=1}^{n} \Big(\bar{c}_j + \sum_{i=1}^{m} a_{ij} y_i^* \Big) x_j.$$

This equation is obtained from algebraic manipulations of the definitions of slack variables and objective function, and must hold for all values of $x_1, \dots, x_n$. Hence, we have

$$z^* = \sum_{i=1}^{m} b_i y_i^*; \quad c_j = \bar{c}_j + \sum_{i=1}^{m} a_{ij} y_i^* \qquad \forall\ j \in \{1, \dots, n\}.$$

Since $\bar{c}_k \le 0$ for every $k \in \{1, \dots, m+n\}$ we get

$$\sum_{i=1}^{m} a_{ij} y_i^* \ge c_j \qquad \forall\ j \in \{1, \dots, n\}$$
$$y_i^* \ge 0 \qquad \forall\ i \in \{1, \dots, m\}.$$

This shows that $(y_1^*, \dots, y_m^*)$ is a feasible solution of (**D**). Finally, $z^* = \sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$. ■

3.11.1 Relating the Primal and Dual Problems

First, notice that dual of a dual problem is the original primal problem, i.e., dual of (**D**) is (**P**). A nice corollary to this observation is that **a linear program has an optimal solution if and only if its dual has an optimal solution**.

		Dual		
		Optimal	Infeasible	Unbounded
Primal	Optimal	$\surd$	$\times$	$\times$
	Infeasible	$\times$	$\surd$	$\surd$
	Unbounded	$\times$	$\surd$	$\times$

Table 3.1: Primal-dual combinations possibilities

By Lemma 24, if the primal problem is unbounded, then the dual problem is infeasible. To see this, assume for contradiction that the dual is feasible when the primal is unbounded. This means, the feasible dual solution provides an upper bound on the optimal value of the primal problem. This is a contradiction since the primal problem is unbounded. By the same argument, if the dual is unbounded, then the primal is infeasible.

This also shows that if the primal and dual are both feasible, then they both have optimal solutions, i.e., none of them is unbounded. However, both the primal and the dual can be infeasible. For example,

$$\begin{aligned} \max 2x_1 - x_2 \\ \text{s.t.} \\ x_1 - x_2 \leq 1 \\ -x_1 + x_2 \leq -2 \\ x_1, x_2 \geq 0 \end{aligned}$$

and its dual are infeasible. We summarize these observations in the Table 3.1.

Duality has important practical applications. In certain cases, it may be better to solve the dual problem than the primal problem, and then read the primal solution from the last row of the final dictionary. For example, a primal problem with 100 constraints and two variables will have two constraints in the dual. Typically, the number of simplex method iterations are insensitive to the number of variables and proportional to the number of rows/constraints. Hence, we may be better off solving the dual in this case.

3.11.2 Farkas Lemma and Duality Theory

Here, we prove the Farkas Lemma using duality theory.

THEOREM **36** *Let A be a $m \times n$ matrix and b be a $m \times 1$ matrix. Suppose $F = \{x \in \mathbb{R}^n_+ : Ax = b\}$ and $G = \{y \in \mathbb{R}^m : yb < 0, yA \geq 0\}$. The set F is non-empty if and only if the set*

G is empty.

Proof: Consider the linear program $\max_{x\in\mathbb{R}^n} 0\cdot x$ subject to $x \in F$. Denote this linear program as (**P**). The dual of this linear program is $\min_{y\in\mathbb{R}^m} yb$ subject to $yA \geq 0$. Denote this linear program as (**D**).

Now, suppose F is non-empty. Then, (**P**) has an optimal value, equal to zero. By strong duality, the optimal value of (**D**) is zero. Hence, for any feasible solution y of (**D**), we have $yb \geq 0$. This implies that G is empty.

Suppose G is empty. Hence, for every feasible solution y of (**D**), $yb \geq 0$. This implies that (**D**) is not unbounded. Since, $y = 0$ is a feasible solution of (**D**), it is an optimal solution. This implies that (**P**) has an optimal solution. Hence, F is non-empty. ■

3.11.3 Complementary Slackness

The question we ask in this section is given a feasible solution of the primal problem (**P**) and a feasible solution of the dual problem (**D**), are there conditions under which these solutions are optimal. The following theorem answers this question.

THEOREM **37 (Complementary Slackness)** *Let $(x_1^*,\ldots,x_n^*)$ and $(y_1^*,\ldots,y_m^*)$ be feasible solutions of (**P**) and (**D**) respectively. $(x_1^*,\ldots,x_n^*)$ is an optimal solution of (**P**) and $(y_1^*,\ldots,y_m^*)$ is an optimal solution of (**D**) if and only if*

$$\Big[\sum_{i=1}^{m} a_{ij}y_i^* - c_j\Big]x_j^* = 0 \qquad \forall\, j \in \{1,\ldots,n\} \tag{CS-1}$$

$$\Big[b_i - \sum_{j=1}^{n} a_{ij}x_j^*\Big]y_i^* = 0 \qquad \forall\, i \in \{1,\ldots,m\}. \tag{CS-2}$$

Proof: Denote $y^* := (y_1^*,\ldots,y_m^*)$ and $x^* := (x_1^*,\ldots,x_n^*)$. Since x^* and y^* are feasible, we immediately get

$$\Big[\sum_{i=1}^{m} a_{ij}y_i^* - c_j\Big]x_j^* \geq 0 \qquad \forall\, j \in \{1,\ldots,n\}$$

$$\Big[b_i - \sum_{j=1}^{n} a_{ij}x_j^*\Big]y_i^* \geq 0 \qquad \forall\, i \in \{1,\ldots,m\}.$$

Now suppose that x^* and y^* are optimal. Assume for contradiction that one of the inequalities in the first set of constraints is not tight. In that case, adding up all the constraints in the first set will give us

$$\begin{aligned}
0 &< \sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij}y_i^* x_j^* - \sum_{j=1}^{n} c_j x_j^* \\
&\leq \sum_{i=1}^{m} b_i y_i^* - \sum_{j=1}^{n} c_j x_j^* \qquad \text{Because } x^* \text{ is feasible to } (\mathbf{P}) \\
&= 0 \qquad \text{Because } x^* \text{ and } y^* \text{ are optimal solutions and Theorem 35}
\end{aligned}$$

This gives us a contradiction. A similar proof shows (4.12) holds.

Now suppose (4.11) and (4.12) holds. Then, add all the equations in (4.11) to get

$$\sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij}x_j^* y_i^* = \sum_{j=1}^{n} c_j x_j^*.$$

Similarly, add all the equations in (4.12) to get

$$\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}x_j^* y_i^* = \sum_{i=1}^{m} b_i y_i^*.$$

This gives us $\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$. By Theorem 35, x^* is an optimal solution of (**P**) and y^* is an optimal solution of (**D**). ■

Theorem 37 gives us a certificate of proving optimality. The idea is clear from our earlier interpretation of optimal dual variable values as negative of coefficients of slack variables in the objective function row of the final simplex dictionary. If a dual variable has positive optimal value, this implies that coefficient of slack variable is negative. This further implies that the slack variable is non-basic in the final simplex dictionary. Hence, its value is zero in the primal optimal solution. This implies that the corresponding constraint is binding in the optimal solution. Similarly, if some constraint is non-binding, then the corresponding slack variable has positive value in the optimal solution. This implies that the slack variable is basic in the final simplex dictionary, which further implies that its coefficient is zero in the objective function row. Hence, the corresponding dual solution has zero value.

Consider the following example.

$$\begin{aligned}\max x_1 + x_2 \\ \text{s.t.} \\ x_1 \le 1 \\ 2x_1 + 3x_2 \le 6 \\ x_1, x_2 \ge 0.\end{aligned}$$

Consider an optimal solution (x_1^*, x_2^*) of this LP, and assume that $x_1^* < 1$. Now, let (y_1^*, y_2^*) be a dual optimal solution. This should provide a bound to $x_1^* + x_2^* \le (y_1^* + 2y_2^*)x_1^* + (3y_2^*)x_2^* < y_1^* + 6y_2^*$ (since $x_1^* < 1$). By strong duality, this is not possible unless we set $y_1^* = 0$. This is exactly the idea behind complementary slackness.

3.11.4 Interpreting the Dual

In optimization, the dual variables are often called *Lagrange multipliers*. In economics, the dual variables are interpreted as prices of *resources*, where resources are constraints. Consider the following example.

Suppose there are n products that a firm can manufacture. Each product requires the use of m resources. To manufacture product j, the firm needs a_{ij} amount of resource i (naturally, it makes sense to assume $a_{ij} \ge 0$ here, but one need not). The amount of resource i available is b_i. The market price of product j is c_j (again, both b_i and c_j can be assumed to be non-negative in this story). The firm needs to decide how much to manufacture of each product to maximize his revenue subject to resource constraints. The problem can be formulated as a linear program - formulation (**PE**).

$$\begin{aligned}\max \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \\ \sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad \forall\, i \in \{1, \ldots, m\} \\ x_j \ge 0 \qquad \forall\, j \in \{1, \ldots, n\}.\end{aligned} \tag{\textbf{PE}}$$

Now, suppose an investor wants to buy this firm's resources. He proposes a price for every resource. In particular, he proposes a price of y_i for resource i. Moreover, the investor promises that he will set his prices high enough such that the firm gets at least as much selling the resources as he would turning the resources into products and then selling them at price vector c. Hence, the following constraints must hold

$$\sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad \forall\, j \in \{1, \ldots, n\}.$$

Another way to think of these constraints is that if the constraint for product j does not hold, then the firm will not sell resources required to produce product j since by selling them in the market he gets a per unit price of c_j which will be higher than $\sum_{i=1}^{m} a_{ij} y_i$ - the per unit profit from selling.

Of course, all prices offered by the investor must be non-negative. The investor must now try to minimize the price he needs to pay to buy the resources. Hence, he should

$$\min \sum_{i=1}^{m} b_i y_i.$$

In particular, the investor will solve the following linear programming problem (**DE**).

$$\begin{aligned} &\min \sum_{i=1}^{m} b_i y_i \\ &\text{s.t.} \\ &\sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad \forall\, j \in \{1, \ldots, n\} \\ &\qquad y_i \geq 0 \qquad \forall\, i \in \{1, \ldots, m\}. \end{aligned} \tag{DE}$$

Strong duality theorem says that the optimal value of investor's cost equals the optimal value of firm's profit. The dual variables are thus prices for the resources in the primal problem.

Now, let us interpret the complementarity slackness conditions here. Suppose the firm has an optimal production plan in place. In this plan, he does not use resource, say, i completely, i.e., the constraint corresponding to i is not binding. In that case, can sell the resources at unit price of zero. But if the price offered is strictly positive for resource i, then the production plan must be using all the resources. Intuitively, the demand for the input i is high, leading to positive prices.

Chapter 4

Integer Programming and Submodular Optimization

4.1 Integer Programming

Suppose that we have a linear program

$$\max\{cx : Ax \leq b,\ x \geq 0\} \tag{\textbf{P}}$$

where A is an $m \times n$ matrix, c an n-dimensional row vector, b an m-dimensional column vector, and x an n-dimensional column vector of variables. Now, we add in the restriction that some of the variables in (**P**) must take integer values.

If some but not all variables are integer, we have a **Mixed Integer Program (MIP)**, written as

$$\begin{aligned} \max\, & cx + hy \\ & \text{s.t.} \\ & Ax + Gy \leq b \\ & \qquad x \geq 0 \\ & \qquad y \geq 0 \text{ and integer.} \end{aligned} \tag{\textbf{MIP}}$$

where A is again $m \times n$ matrix, G is $m \times p$, h is a p row vector, and y is a p column vector of integer variables.

If all variables are integer, then we have an **Integer Program (IP)**, written as

$$\begin{aligned} & \max cx \\ \text{s.t.} \quad & \\ & Ax \leq b \\ & x \geq 0 \text{ and integer.} \end{aligned} \tag{\textbf{IP}}$$

If the variables in an IP is restricted to take values in $\{0, 1\}$ then the IP is called a **Binary Integer Program (BIP)**.

4.1.1 Common Integer Programming Problems

4.1.1.1 The Assignment Problem

There is a set of indivisible goods $G = \{1, \ldots, n\}$. The goods need to be assigned to a set of buyers $B = \{1, \ldots, m\}$. Each buyer can be assigned at most one good from G. If good $j \in G$ is assigned to buyer $i \in B$, then it generates a *value* of v_{ij}. The objective is to find an *assignment* that maximizes the total value generated.

Define variables x_{ij} to denote if buyer i is assigned good j, i.e., $x_{ij} = 1$ if i is assigned j, and zero otherwise. So, it is a *binary variable*. The constraints should ensure that no good is assigned to more than one buyer and no buyer is assigned more than one good. The objective function is to maximize $\sum_{i \in B} \sum_{j \in G} v_{ij} x_{ij}$. Hence, the formulation is as follows:

$$\begin{aligned} & \max \sum_{i \in B} \sum_{j \in G} v_{ij} x_{ij} \\ \text{s.t.} \quad & \\ & \sum_{i \in B} x_{ij} \leq 1 \qquad \forall\, j \in G \\ & \sum_{j \in G} x_{ij} \leq 1 \qquad \forall\, i \in B \\ & x_{ij} \in \{0, 1\} \qquad \forall\, i \in B,\ \forall\, j \in G. \end{aligned} \tag{\textbf{AP}}$$

4.1.1.2 The $0-1$ Knapsack Problem

There is a budget b available for investment in n projects. Let a_j be the required investment of project j and π_j is the expected profit from project j. The goal is to choose a set of projects to maximize expected profit, given that the total investment should not exceed the budget b.

Define variables x_j to denote if investment is done in project j or not. The problem can be formulated as an IP as follows:

$$\begin{aligned} &\max \sum_{j=1}^{n} \pi_j x_j \\ &\text{s.t.} \\ &\sum_{j=1}^{n} a_j x_j \le b \\ &\qquad x_j \in \{0,1\} \qquad \forall\, j \in \{1,\dots,n\}. \end{aligned} \tag{\textbf{KNS}}$$

Tha Knapsack problem belongs to one of the hardest classes of optimization problems - a fast algorithm to solve such classes of problems is not known. Consider a knapsack problem with three projects having investment requirement of $1, 3$, and 2 respectively and expected profit of $5, 4$, and 3 respectively. Suppose the budget is 5. A *greedy* way of trying to solve for this problem is to find for every project the profit per unit of investment, which are 5, $\frac{4}{3}$, and $\frac{3}{2}$. Then choose the set of projects with the highest profit per unit of investment given the budget constraint. In this case, it is projects 1 and 3. However, the optimal project choice is projects 1 and 2. Thus, the greedy algorithm does not work in the knapsack problem.

4.1.1.3 The Set Covering Problem

There is a set of M streets and a set of N potential centers. We are interested in setting up public facilities (e.g., fire stations, police stations etc.) at the centers to cover the streets. Every center $j \in N$ can cover a subset $S_j \subseteq M$ of streets. To open a center j, there is a cost of c_j. The objective is to cover the streets with minimum possible cost.

First, we *process* the input data to build an *incidence matrix* a, where for every $i \in M$ and $j \in N$, $a_{ij} = 1$ if $i \in S_j$ and zero otherwise. Then let the variable $x_j = 1$ if center $j \in N$ is opened and zero otherwise. This leads to the following formulation:

$$\begin{aligned} &\min \sum_{j \in N} c_j x_j \\ &\text{s.t.} \\ &\sum_{j \in N} a_{ij} x_j \ge 1 \qquad \forall\, i \in M \\ &\qquad x_j \in \{0,1\} \qquad \forall\, j \in N. \end{aligned} \tag{\textbf{SC}}$$

4.2 Relaxation of Integer Programs

Unlike linear programs, integer programs are hard to solve. We look at some ways to get bounds on the objective function of an integer program without solving it explicitly.

Consider an IP in the form of (**IP**). Denote the feasible region of (**IP**) as $X \subseteq \mathbb{Z}^n$, where $\mathbb{Z}$ is the set of integers. We can write (**IP**) simply as $\max cx$ subject to $x \in X$.

Definition **21** *A* **relaxed problem (RP)** *defined as* $\max f(x)$ *subject to* $x \in T \subseteq \mathbb{R}^n$ *is a* **relaxation** *of (**IP**) if*

- $X \subseteq T$,
- $f(x) \geq cx$ *for all* $x \in X$.

Notice that the relaxation does not impose integrality constraint on x. Also, the feasible space of the relaxation contains the feasible space of (**IP**). Lastly, the objective function value of the relaxation is not less than the objective function value of (**IP**) over the feasible set X (but not necessarily the entire T).

An example:

$$\begin{aligned} \max 4x_1 - x_2 \\ \text{s.t.} \\ 7x_1 - 2x_2 &\leq 14 \\ x_2 &\leq 3 \\ 2x_1 - 2x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \text{ and integers.} \end{aligned}$$

A relaxation of this IP can be the following linear program.

$$\begin{aligned} \max 4x_1 \\ \text{s.t.} \\ 7x_1 - 2x_2 &\leq 14 \\ x_2 &\leq 3 \\ 2x_1 - 2x_2 &\leq 3 \end{aligned}$$

To see why this is a relaxation, notice that the feasible region of this LP contains all the integer points constituting the feasible region of the IP. Also $4x_1 \geq 4x_1 - x_2$ for all $x_1, x_2 \geq 0$.

LEMMA **25** *Let the objective function value at the optimal solution of (**IP**) and its relaxation (**RP**) be z and z^r respectively (if they exist). Then $z^r \geq z$.*

Proof: Let x^* be an optimal solution of (**IP**). $z = cx^*$. By definition of the relaxation $f(x^*) \geq cx^* = z$. Also, since $x^* \in X \subseteq T$, $z^r \geq f(x^*)$. Hence $z^r \geq z$. ■

Not all relaxations are interesting - in the sense that they may give arbitrarily bad bounds. But an interesting relaxation is the following.

DEFINITION **22** *A* **linear programming relaxation** *of (**IP**) is the following linear program:*

$$\begin{aligned} &\max cx \\ &s.t. \\ &\quad Ax \leq b \\ &\quad x \geq 0. \end{aligned}$$

So, a linear programming relaxation of an IP is defined exactly the same as the IP itself except that the integrality constraints are not there.

The linear programming relaxation for the example above has an optimal solution with $x_1 = \frac{20}{7}$ and $x_2 = 3$, with the objective function value $z^{lp} = \frac{59}{7}$. Hence the objective function value of IP at the optimal solution cannot be more than $\frac{59}{7}$. We can say more. Since the optimal solution is integral this bound can be set to 8.

LEMMA **26** 1. *If a relaxation RP is infeasible, then the original IP is infeasible.*

2. *Let x^* be an optimal solution of RP. If $x^* \in X$ and $f(x^*) = cx^*$, then x^* is an optimal solution of IP.*

Proof: (1) As RP is infeasible, $T = \emptyset$, and thus $X = \emptyset$. (2) As $x^* \in X$, optimal objective function value of IP $z \geq cx^* = f(x^*) = z^r$, where z^r is the optimal objective function value of RP. But we know that $z \leq z^r$. Hence $z = z^r$. ■

Notice that an LP relaxation of an IP has the same objective function. Hence, if LP relaxation gives integer solution, then we can immediately conclude that it is indeed the optimal solution of the IP. As an example, consider the following integer program:

$$\begin{aligned} \max 7x_1 + 4x_2 + 5x_3 + 2x_4 \\ \text{s.t.} \\ 3x_1 + 3x_2 + 4x_3 + 2x_4 \leq 6 \\ x_1, x_2, x_3, x_4 \geq 0 \text{ and integers.} \end{aligned}$$

The LP relaxation of this IP gives an optimal solution of $(2, 0, 0, 0)$. This is an integer solution. Hence it is an optimal solution of the integer program.

If a relaxation of an IP is unbounded, then we cannot draw any conclusion. First, the IP is a relaxation of itself (by setting $T = X$ and $f(x) = cx$). Hence, if a relaxation of an IP is unbounded, then the IP can be unbounded. Second, consider the following IP.

$$\begin{aligned} \max -x \\ \text{s.t.} \\ x \geq 2 \\ x \ \text{ integer.} \end{aligned}$$

This IP has an optimal solution of $x = 2$. But if we remove the integrality constraint and $x \geq 2$ constraint, then the feasible set is $\mathbb{R}$, and the relaxed problem is unbounded. Finally, if a relaxation of an IP is unbounded, then the IP can be infeasible. Consider the following IP.

$$\begin{aligned} \max x_1 \\ \text{s.t.} \\ x_2 \leq 1.2 \\ x_2 \geq 1.1 \\ x_1, x_2 \geq 0 \text{ and integer.} \end{aligned}$$

This IP has no feasible solution. However, the LP relaxation is unbounded.

For binary programs, the LP relaxation should add ≤ 1 constraints for all the binary variables. For example, if we impose that $x_1, \ldots, x_4$ are binary variables, then the LP relaxation will impose the constraints $x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1$. This LP relaxation gives an optimal solution of $(1, 1, 0, 0)$, which is also an optimal solution of the (binary) integer program.

Notice that a feasible (integral) solution to an integer program provides a lower bound (of a maximization problem). This is called a **primal bound**. But a relaxation gives an upper bound for the problem. This is called a **dual bound**.

4.3 Integer Programs with Totally Unimodular Matrices

Definition **23** *A matrix A is* **totally unimodular** *(TU) if every square submatrix of A has determinant $+1$, -1, or 0.*

$$\mathbf{A_1} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix A_1 has a determinant of 2. Hence, it is not TU.

$$\mathbf{A_2} = \begin{bmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

However, matrix A_2 is TU. Notice some important properties of TU matrices:

1. If A is a TU matrix, then every element of A is either 1, -1, or 0.
2. By the definition of determinant, the transpose of a TU matrix is also TU. Since transpose of the transpose of a matrix is the same matrix, we can conclude that if the transpose of a matrix is TU, then so is the original matrix.
3. Also note that if a $m \times n$ matrix A is TU and I is a $m \times m$ identity matrix, then the matrix $[A|I]$ is also TU.
4. If A is a TU matrix, then multiplying a particular row or column of A by -1 gives a TU matrix.

Theorem **38** *Consider an IP of the following form:* $\max \mathbf{cx}$ *subject to* $\mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ *and* $\mathbf{x}$ *integer. Suppose* $\mathbf{b}$ *is an integer matrix. If* $\mathbf{A}$ *is totally unimodular, then the optimal solution of the LP relaxation, if it exists, is also the optimal solution of IP.*

Proof: First if A is TU, by adding slack variables, we get a new matrix which is of the form $[A|I]$, where I is the identity matrix. So, this matrix is also a TU matrix. If the optimal solution of LP relaxation exists, then let B be the set of basic variables in the final dictionary of the simplex method. We can then write the optimal solution in matrix form as $\mathbf{x_B} = \mathbf{A_B^{-1}b}$ and the remaining (non-basic) variables take value zero. Since $[\mathbf{A}|\mathbf{I}]$ is TU, determinant of $\mathbf{A_B}$, which is a square submatrix of $[\mathbf{A}|\mathbf{I}]$ is 1 or -1 (it cannot be zero since

$\mathbf{A_B}$ is non-singular). So, $\mathbf{A_B^{-1}b}$ is integral. By Lemma (26), $(\mathbf{x_B}, \mathbf{0})$ is an optimal solution of IP. ■

Note that in many problems, we have the constraints in the form $Ax = b$ instead of $Ax \leq b$. But $Ax = b$ can be written as $Ax \leq b$ and $-Ax \leq -b$. Now, note that the matrix $(A^T, -A^T)$ is totally unimodular if A is totally unimodular (to prove this, add to A a row of $-A$ successively, and perform the elementary matrix operation of substituting the new row with the sum of itself and its negative row to get a row of zeros). This will show that the new constraint matrix is totally unimodular.

Theorem 38 inspires us to look for sufficient conditions under which a matrix can be TU. Here is a simple sufficient condition.

Theorem **39** *A matrix A is TU if*

1. *every element of A is 1,-1, or 0,*
2. *every column contains at most two non-zero elements,*
3. *there exists a partition (M_1, M_2) of set of rows M of A such that each column j containing two non-zero elements satisfies $\sum_{i \in M_1} a_{ij} = \sum_{i \in M_2} a_{ij}$ (Note: Here, M_1 or M_2 can be empty also. If $M_1 = \emptyset$ then $\sum_{i \in M_1} a_{ij} = 0$.).*

Proof: Suppose A is not TU. Consider the smallest square submatrix of A whose determinant is $\notin \{1, -1, 0\}$. Let this submatrix be B. Let B contain the set of rows L. By the first condition $|L| > 1$. B cannot contain a column with a single non-zero entry, as othewise the minor corresponding to that entry will also have a determinant $\notin \{1, -1, 0\}$, and B will not be minimal. So, B contains two non-zero entries in each column.

Now, note that $L \cap M_1$ and $L \cap M_2$ is a partition of rows in L. Using the last condition and the fact that all non-zero entries of a column are either in $L \cap M_1$ or in $L \cap M_2$, we get that

$$\sum_{i \in L \cap M_1} a_{ij} = \sum_{i \in L \cap M_2} a_{ij}.$$

Adding the rows of $L \cap M_1$ and subtracting from the rows in $L \cap M_2$ gives the zero vector, and so determinant of B is zero (this follows from the fact that determinant of a matrix remains the same after an *elementary* matrix operation - the elementary matrix operation here is to replace the first row entries by sum of entries of rows in $L \cap M_1$ minus the sum of entries of rows of $L \cap M_2$, and this generates a zero row vector), which is a contradiction. ■

A simple consequence of Theorem (39) is that the following matrix is TU (using $M_1 = M$ and $M_2 = \emptyset$).

1. Each entry is $1, -1$, or 0.
2. Each row (column) contains at most two non-zero entries.
3. If a row (column) contains two non-zero entries, then the entries are of opposite sign.

Using this, we can verify that the following matrices are TU.

$$\mathbf{A_3} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A_4} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{A_3}$ is TU since every entry is ± 1 or zero, every column containts at most two non-zero entries, and if a column contains two non-zero entries, then they are of opposite sign. To see why $\mathbf{A_4}$ is TU, multiply first and third columns by -1. Then we get a matrix desired in Proposition (39), and this shows that $\mathbf{A_4}$ is TU.

4.3.1 Assignment Problem

Let us revisit the constraints of the assignment problem.

$$\sum_{i \in B} x_{ij} \leq 1 \qquad \forall\, j \in G$$
$$\sum_{j \in G} x_{ij} \leq 1 \qquad \forall\, i \in B.$$

Note that the entries in the coefficient matrix are 0 or ± 1. Further, for every $i \in B$ and $j \in G$, the variable x_{ij} appears in exactly two constraints: once for $i \in B$ and once for $j \in G$. Now multiply the entries corresponding to entries of $i \in B$ by -1 (note that the original constraint matrix is TU if and only if this matrix is TU). Now, every column of the constraint matrix has exactly two non-zero entries and both have opposite signs. This implies that the constraint matrix is TU. Since the b matrix is a matrix of 1s, we get that the LP relaxation of the assignment problem IP gives integral solution.

Here is an example which illustrates why the constraint matrix is TU. Suppose there are two goods and three buyers. Then, there are five constraints.

$$\begin{aligned} x_{11} + x_{12} &\leq 1 \\ x_{21} + x_{22} &\leq 1 \\ x_{31} + x_{32} &\leq 1 \\ x_{11} + x_{21} + x_{31} &\leq 1 \\ x_{12} + x_{22} + x_{32} &\leq 1. \end{aligned}$$

The constraint matrix can be written as follows.

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

Multiplying the last two rows of $\mathbf{A}$ by -1, we get $\mathbf{A}'$ as below, and it is clear that it satisfies the sufficient conditions for being TU.

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

4.3.2 Potential Constraints are TU

Consider the potential constraints of a weighted digraph $G = (N, E, w)$. It says $p : N \to \mathbb{R}$ is a potential if

$$p_j - p_i \leq w_{ij} \qquad \forall \, (i, j) \in E.$$

Now, for every constraint corresponding to $(i, j) \in E$, we have exactly two variables: p_i and p_j. The coefficients of these two variables are of opposite sign and are 1 or -1. Hence, the sufficient conditions for the constraint matrix to be TU are met. This implies that if weights of this digraph are integers, then any linear optimization over these potentials must give an integer potential as an optimal solution.

4.3.3 Network Flow Problem

The network flow problem is a classical problem in combinatorial optimization. In this problem, we are given a digraph $G = (N, E)$. Every edge (i, j) has a capacity $h_{ij} \geq 0$. Every node $i \in N$ has a supply/demand of b_i units of a commodity. If $b_i > 0$, then it is a supply node, if $b_i < 0$, then it is a demand node, else it is a neutral node. The assumption is total supply equals total demand: $\sum_{i \in N} b_i = 0$. There is a cost function associated with the edges of the graph: $c : E \to \mathbb{R}_+$, where c_{ij} denotes the cost of flowing one unit from node i to node j. The objective is to take the units of commodity from supply node and place it at the demand nodes with minimum cost.

The decision variable of the problem is x_{ij} for all $(i, j) \in E$. The amount x_{ij} is called the **flow** in edge (i, j). Hence, the objective function is clearly the following.

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

For constraints, note that the flow must respect the capacity constraints of edges. Hence, the following constraint must hold.

$$0 \leq x_{ij} \leq h_{ij} \qquad \forall\ (i, j) \in E.$$

Define the nodes on outgoing edges from i as $N^+(i) = \{j \in N : (i, j) \in E\}$ and the nodes on incoming edges from i as $N^-(i) = \{j \in N : (j, i) \in E\}$. Now, consider the following set of constraints.

$$\sum_{j \in N^+(i)} x_{ij} - \sum_{j \in N^-(i)} x_{ji} = b_i \qquad \forall\ i \in N.$$

The above constraints are called **flow balancing** constraints, i.e., the amount of flow out of a node must equal the flow into a node plus the supply at that node. Note that if we add all the flow balancing constraints, the left hand side is zero and the right hand side is $\sum_{i \in N} b_i$. Hence, for the problem to be feasible, it is necessary that $\sum_{i \in N} b_i = 0$. Hence, the **minimum cost network flow** problem can be formulated as follows.

$$\begin{aligned} \min \sum_{(i,j) \in E} c_{ij} x_{ij} & \\ \text{s.t.} & \qquad\qquad \textbf{(NF)} \\ \sum_{j \in N^+(i)} x_{ij} - \sum_{j \in N^-(i)} x_{ji} = b_i & \qquad \forall\ i \in N. \\ 0 \leq x_{ij} \leq h_{ij} & \qquad \forall\ (i, j) \in E. \end{aligned}$$

Note that (**NF**) is a linear program. Nevertheless, we show that the constraint matrix of this problem is TU. To see this, note that the capacity constraint matrix is an identity matrix. Hence, it suffices to show that the flow balancing constraint matrix is TU. To see this, note that every edge is an incoming edge of a unique node and outgoing edge of a unique node. Hence, for every $(i, j) \in E$, x_{ij} appears exactly in two flow balancing constraints, once with coefficient 1 and again with coefficient -1. By our earlier result, this matrix is TU. A consequence of this result is that if the supply (b_is) and capacities (h_{ij}s) are integral, then there exists an integral minimum cost network flows.

The following example illustrates the idea of TU in network flow problem. Consider the digraph in Figure 4.1. We do not show the costs or capacities of the edges. The supply at each node is indicated near the respective node. The flow balancing constraints are:

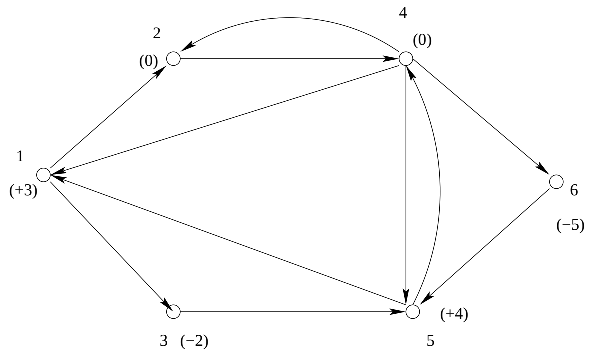


Figure 4.1: Network Flow Example

$$\begin{aligned} x_{12} + x_{14} - x_{31} - x_{51} &= 3 \\ x_{23} - x_{32} - x_{12} &= 0 \\ x_{31} + x_{32} + x_{35} + x_{36} - x_{53} - x_{23} &= 0 \\ x_{45} - x_{14} &= -2 \\ x_{51} + x_{53} - x_{35} - x_{45} - x_{65} &= 4 \\ x_{65} - x_{36} &= -5. \end{aligned}$$

One can easily see how every variable appears in two constraints with opposite sign coefficients.

4.3.4 The Shortest Path Problem

The shortest path problem is a particular type of network flow problem. Here, a digraph $G = (N, E)$ and a cost function $c : E \to \mathbb{R}_+$ is given. Also, given are two nodes: a **source** node s and a **terminal** node t. It is assumed that there exists at least one path from s to every other node (indeed, if some node does not have a path from s, it can be removed from consideration). Further, there are no incoming edges to the source and no outgoing edges from the terminal. The objective is to find the **shortest** path from s to t, i.e., the path with the minimum length over all paths from s to t.

To model this as a minimum cost network flow problem, let $b_i = 1$ if $i = s$, $b_i = -1$ if $i = t$, and $b_i = 0$ otherwise. Let $h_{ij} = 1$ for all $(i, j) \in E$. A flow of $x_{ij} = 1$ indicates that edge (i, j) is chosen. Hence, the formulation for the shortest path problem is as follows.

$$\begin{aligned}
\min \sum_{(i,j)\in E} c_{ij}x_{ij} & \\
\text{s.t.} & \qquad\qquad \textbf{(SP)}\\
\sum_{j\in N^+(i)} x_{ij} = 1 & \qquad i = s\\
\sum_{j\in N^+(i)} x_{ij} - \sum_{j\in N^-(i)} x_{ji} = 0 & \qquad i \notin \{s,t\}\\
- \sum_{j\in N^-(i)} x_{ji} = -1 & \qquad i = t\\
x_{ij} \in \{0,1\} & \qquad \forall\ (i,j) \in E
\end{aligned}$$

To convince that there is an optimal solution which gives a path from s to t, note that by the first constraint, there is only one flow from s. For every node which is not s or t, there is a flow from that node if and only if there is a flow into that node. Finally, by the last constraint, there is one unit flow into node t. Hence, it describes a path. The other possibility is that we get a cycle. But deleting the cycle cannot increase costs since costs are non-negative.

Since the constraint matrix is TU, we can write the relaxation as

$$0 \le x_{ij} \le 1 \qquad \forall\ (i,j) \in E.$$

Since the objective is to minimize the total cost of a path from s to t, the $x_{ij} \le 1$ constraints are redundant. Another way to see this is to consider the dual of the LP relaxation of (**SP**) without the $x_{ij} \le 1$ constraints. For this, we write the LP relaxation of (**SP**) without the $x_{ij} \leq 1$ constraints first.

$$
\begin{aligned}
\min \sum_{(i,j)\in E} c_{ij}x_{ij} & \\
\text{s.t.} & \qquad\qquad\qquad \textbf{(SP-R)} \\
-\sum_{j\in N^+(i)} x_{ij} &= -1 \qquad i = s \\
-\sum_{j\in N^+(i)} x_{ij} + \sum_{j\in N^-(i)} x_{ji} &= 0 \qquad i \notin \{s,t\} \\
\sum_{j\in N^-(i)} x_{ji} &= 1 \qquad i = t \\
x_{ij} &\geq 0 \qquad \forall\, (i,j) \in E
\end{aligned}
$$

The dual of (**SP-R**) is the following formulation.

$$
\begin{aligned}
\max p_t - p_s & \qquad (4.1) \\
\text{s.t.} & \qquad \textbf{(DSP)} \\
p_j - p_i \leq c_{ij} & \qquad \forall\, (i,j) \in E.
\end{aligned}
$$

Note that the constraints in the dual are the potential constraints. We already know a feasible solution of this exists since there can be no cycles of negative length (we assume costs are non-negative here). We know that one feasible potential is $\pi_s = 0$ and $\pi_i =$ shortest path from s to i for all $i \neq s$. Hence, $\pi_t - \pi_s =$ shortest path from s to t. But for any potential p, we know that $p_t - p_s \leq$ shortest path from s to t (to prove it, take the shortest path from s to t and add the potential constaints of the edges in that path). Hence, π describes an optimal solution. By strong duality, (**SP-R**) gives the shortest path from s to t.

4.4 Application: Efficient Assignment with Unit Demand

Consider an economy where the set of agents is denoted by $M = \{1, \ldots, m\}$ and consisting of set of indivisible (not necessarily identical) goods denoted by $N = \{1, \ldots, n\}$. Let $v_{ij} \geq 0$ be the value of agent $i \in M$ to good $j \in N$. Each agent is interested in buying at most one good.

For clarity of expressions that follow, we add a dummy good 0 to the set of goods. Let $N^+ = N \cup \{0\}$. If an agent is not assigned any good, then it is assumed that he assigned the dummy good 0. Further, a dummy good can be assigned to more than one agent and the value of each agent for the dummy good is zero.

A feasible assignment of goods to agents is one where every agent is assigned exactly one good from N^+ and every good in N is assigned to no more than one agent from M. An **efficient assignment** is one that maximizes the total valuations of the agents.

To formulate the problem of finding an efficient assignment as an integer program let $x_{ij} = 1$ if agent i is assigned good j and zero otherwise.

$$
\begin{aligned}
V = \max \sum_{i \in M} \sum_{j \in N} v_{ij} x_{ij} & \\
\text{s.t.} & \qquad\qquad (\mathbf{IP}) \\
\sum_{i \in M} x_{ij} \leq 1 \qquad & \forall\, j \in N \\
x_{i0} + \sum_{j \in N} x_{ij} = 1 \qquad & \forall\, i \in M \\
x_{ij} \in \{0,1\} \qquad & \forall\, i \in M,\ \forall\, j \in N. \\
x_{i0} \geq 0 \text{ and integer} \qquad & \forall\, i \in M.
\end{aligned}
$$

Notice that the first two sets of constraints ensure that $x_{ij} \leq 1$ for all $i \in M$ and for all $j \in N$. The only difference from the constraint matrix of the assignment problem is the introduction of x_{i0} variables, whose coefficients form an identity matrix. Hence, the constraints matrix of this formulation is also TU. Hence, the LP relaxation of **IP** gives integral solution. Note here that the LP relaxation of $x_{ij} \in \{0,1\}$ is $0 \leq x_{ij} \leq 1$, but the constraints $x_{ij} \leq 1$ are redundant here. So, we can write formulation (**IP**) as the following linear program:

$$
\begin{aligned}
V = \max \sum_{i \in M} \sum_{j \in N} v_{ij} x_{ij} & \\
\text{s.t.} & \qquad\qquad (\mathbf{LP}) \\
\sum_{i \in M} x_{ij} \leq 1 \qquad & \forall\, j \in N \\
x_{i0} + \sum_{j \in N} x_{ij} = 1 \qquad & \forall\, i \in M \\
x_{i0}, x_{ij} \geq 0 \qquad & \forall\, i \in M,\ \forall\, j \in N.
\end{aligned}
$$

Now, consider the dual of (**LP**). For that, we associate with constraint corresponding to agent $i \in M$ a dual variable π_i and with constraint corresponding to good $j \in N$ a dual variable p_j. The p_j variables are non-negative since the corresponding constraints are inequality but

π_i variables are free.

$$\begin{aligned} \min \sum_{i \in M} \pi_i + \sum_{j \in N} p_j & \\ \text{s.t.} & \qquad\qquad\qquad\qquad \textbf{(DP)} \\ \pi_i + p_j \geq v_{ij} \qquad & \forall\, i \in M,\ \forall\, j \in N \\ \pi_i \geq 0 \qquad & \forall\, i \in M \\ p_j \geq 0 \qquad & \forall\, i \in M,\ \forall\, j \in N. \end{aligned}$$

Note here that even though the π_i variables are free, when we write the constraint corresponding to variable x_{i0}, it turns out that we recover the non-negativity constraints. The dual has interesting economic interpretation. $\{p_j\}_{j \in M}$ can be thought as a price vector on goods. Given the price vector p, $\pi_i \geq \max_{j \in N}[v_{ij} - p_j]$ for every $i \in M$. If we set $p_0 = 0$, then $v_{i0} - p_0 = 0$ implies that $\pi_i \geq 0$ can be folded into $\pi_i \geq \max_{j \in N^+}[v_{ij} - p_j]$.

Given any price vector $p \in \mathbb{R}_+^{|N^+|}$ (i.e., on set of goods, including the dummy good), with $p_0 = 0$, define **demand set of buyer** i at this price vector p as

$$D_i(p) = \{j \in N^+ : v_{ij} - p_j \geq v_{ik} - p_k\ \forall\, k \in N^+\}.$$

Definition **24** *A tuple (p, x) is a Walrasian equilibrium, where p is a price vector and x is a feasible allocation (i.e., a feasible solution to **LP**), if*

1. *$x_{ij} = 1$ implies that $j \in D_i(p)$ (every buyer is assigned a good from his demand set) and*
2. *$\sum_{i \in M} x_{ij} = 0$ implies that $p_j = 0$ (unassigned good has zero price).*

Given a price vector p (with $p_0 = 0$), we can construct a dual feasible solution from this p. This is done by setting $\pi_i = \max_{j \in N^+}[v_{ij} - p_j]$. Clearly, this (p, π) is a feasible solution of (**DP**) - to be exact, we consider price vector p without the dummy good component here since there is no dual variable corresponding to dummy good. This is because $\pi_i \geq v_{ij} - p_j$ for all $i \in M$ and for all $j \in N$. Further $\pi_i \geq v_{i0} - p_0 = 0$ for all $i \in M$. From now on, whenever we say that p is a dual feasible solution, we imply that the corresponding π variables are defined as above.

Theorem **40** *Let p be a feasible solution of (**DP**) and x be a feasible solution of (**LP**). (p, x) is a Walrasian equilibrium if and only if p and x are optimal solutions of (**DP**) and (**LP**) respectively.*

Proof: Suppose (p,x) is a Walrasian equilibrium. Define $\pi_i = \max_{j \in N^+}[v_{ij} - p_j]$ for all $i \in M$. As argued earlier, (p,π) is a feasible solution of (**DP**). Now, Walrasian equilibrium conditions can be written as

$$\begin{aligned} [\pi_i - (v_{ij} - p_j)]x_{ij} &= 0 \qquad \forall\, i \in M,\ \forall\, j \in N^+ \\ [1 - \sum_{i \in M} x_{ij}]p_j &= 0 \qquad \forall\, j \in N. \end{aligned}$$

The first condition comes from the fact that if $x_{ij} = 1$ then $j \in D_i(p)$, which implies that $\pi_i = v_{ij} - p_j \geq v_{ik} - p_k$ for all $k \in N^+$. The second condition comes from the fact that unassigned goods must have zero price. Now, the CS conditions of the primal and dual problems are

$$\begin{aligned} [\pi_i - (v_{ij} - p_j)]x_{ij} &= 0 \qquad \forall\, i \in M,\ \forall\, j \in N \\ \pi_i x_{i0} &= 0 \qquad \forall\, i \in M \\ [1 - \sum_{i \in M} x_{ij}]p_j &= 0 \qquad \forall\, j \in N. \end{aligned}$$

We can fold the second CS condition into first since $p_0 = 0$ and $v_{i0} = 0$ for all $i \in M$. This gives the CS conditions as

$$\begin{aligned} [\pi_i - (v_{ij} - p_j)]x_{ij} &= 0 \qquad \forall\, i \in M,\ \forall\, j \in N^+ \\ [1 - \sum_{i \in M} x_{ij}]p_j &= 0 \qquad \forall\, j \in N. \end{aligned}$$

Hence, (p,x) satisfies the CS conditions. So, p is an optimal solution of (**DP**) and x is an optimal solution of (**LP**).

The other direction of the proof also follows similarly from the equivalence between the CS conditions and the Walrasian equilibrium conditions. ■

4.5 Application: Efficient Combinatorial Auctions

In this section, we study the combinatorial auctions problem. Let $N = \{1,\ldots,n\}$ be the set of goods and $M = \{1,\ldots,m\}$ be the set of buyers. Let $\Omega = \{S : S \subseteq N\}$ be the set of all bundles of goods. The valuation function of buyer $i \in M$ is $v_i : \Omega \to \mathbb{R}_+$. For buyer $i \in M$, $v_i(S)$ denotes the value on bundle $S \in \Omega$. We assume $v_i(\emptyset) = 0$ for all buyers $i \in M$.

An **allocation** $X = (X_1,\ldots,X_m)$ is a partition of N such that $X_i \in \Omega$ for all $i \in M$, $X_i \cap X_j = \emptyset$ for all $i \neq j$, and $\cup_{i \in M} X_i \subseteq N$. In an allocation X, X_i denotes the bundle of

	$\{1\}$	$\{2\}$	$\{1,2\}$
$v_1(\cdot)$	8	11	12
$v_2(\cdot)$	5	9	15
$v_3(\cdot)$	5	7	16

Table 4.1: An example of combinatorial auction

goods assigned to buyer $i \in M$. So, the two differences from our earlier notation are (a) an allocation X indicates a partition and an assignment of goods and (b) not all goods need to be assigned in an allocation. Let $\mathbb{X}$ be the set of all allocation. An allocation $X \in \mathbb{X}$ is **efficient** if

$$\sum_{i \in M} v_i(X_i) \geq \sum_{i \in M} v_i(Y_i) \qquad \forall\, Y \in \mathbb{X}.$$

Consider the example in Table 4.1. The efficient allocation for this example is $(\{1\}, \{2\}, \emptyset)$, meaning buyer 1 gets good 1, buyer 2 gets good 2, and buyer 3 gets nothing. This gives a total value of 17, which is higher than the total value obtained in any other allocation.

4.5.1 Formulation as an Integer Program

Our objective is to formulate the problem of finding an efficient allocation. The decision variable is: $x_i(S) \in \{0,1\}$ for all buyers $i \in M$ and for all $S \in \Omega$. $x_i(S)$ should be 1 if buyer $i \in M$ is assigned bundle $S \in \Omega$, and zero otherwise. We should have two sets of constraints: (1) to ensure that every buyer gets some bundle of goods (may be the empty set) and (2) to ensure that every good is assigned to at most one buyer. The objective function maximizes the total value of buyers.

$$V(M,N;v) = \max \sum_{i \in M} \sum_{S \in \Omega} v_i(S) x_i(S)$$

s.t. **(CA-IP)**

$$\sum_{S \in \Omega} x_i(S) = 1 \qquad \forall\, i \in M \tag{4.2}$$

$$\sum_{i \in M} \sum_{S \in \Omega : j \in S} x_i(S) \leq 1 \qquad \forall\, j \in N \tag{4.3}$$

$$x_i(S) \in \{0,1\} \qquad \forall\, i \in M,\ \forall\, S \in \Omega. \tag{4.4}$$

The LP relaxation of formulation (**CA-IP**) does not always give integral solutions. Consider the example in Table 4.2. A feasible solution of (**CA-LP**), which is not integral and gives

	$\{1\}$	$\{2\}$	$\{1,2\}$
$v_1(\cdot)$	8	11	12
$v_2(\cdot)$	5	9	18

Table 4.2: An example where Walrasian equilibrium does not exist

an objective function value higher than the optimal solution of (**CA-IP**) is: $x_1(\{1\}) = x_1(\{2\}) = 0.5$ and $x_2(\{1,2\}) = x_2(\emptyset) = 0.5$. The value of objective function of (**CA-LP**) from this feasible solution is $(8+11)0.5 + 18(0.5) = 18.5 > 18 =$ objective function value of optimal solution of (**CA-IP**). Hence linear relaxation of (**CA-IP**) does not give an integral solution in this example.

However, if we restrict Ω to be only singleton bundles. i.e., buyers can be assigned at most one good (this is the assignment problem model we studied earlier), then the resulting constraint matrix becomes totally unimodular, and the LP relaxation always gives integral solution. Besides the assignment problem setting, there are other general settings where the LP relaxation of (**CA-IP**) gives integral solutions. The exact nature of these settings will not be covered in this course. These settings arise in specific types of valuation functions, and do not necessarily result in a TU constraint matrix.

We assume that the valuation functions are such that LP relaxation of (**CA-IP**) gives an optimal solution of formulation (**CA-IP**). Then, the efficient allocation problem can be reformulated as:

$$V(M,N;v) = \max \sum_{i\in M}\sum_{S\in\Omega} v_i(S)x_i(S)$$

$$\text{s.t.} \qquad \textbf{(CA-LP)}$$

$$\sum_{S\in\Omega} x_i(S) = 1 \qquad \forall\, i \in M \tag{4.5}$$

$$\sum_{i\in M}\sum_{S\in\Omega: j\in S} x_i(S) \leq 1 \qquad \forall\, j \in N \tag{4.6}$$

$$x_i(S) \geq 0 \qquad \forall\, i \in M,\ \forall\, S \in \Omega. \tag{4.7}$$

	$\{1\}$	$\{2\}$	$\{1,2\}$
$v_1(\cdot)$	6	8	13
$v_2(\cdot)$	5	8	12

Table 4.3: An example where Walrasian equilibrium exists

The dual of this formulation is:

$$V(M,N;v) = \min \sum_{i \in M} \pi_i + \sum_{j \in N} p_j$$

$$\text{s.t.} \qquad \textbf{(CA-DLP)}$$

$$\pi_i + \sum_{j \in S} p_j \geq v_i(S) \qquad \forall\, i \in M,\ \forall\, S \in (\Omega \setminus \emptyset) \tag{4.8}$$

$$\pi_i \geq 0 \qquad \forall\, i \in M \tag{4.9}$$

$$p_j \geq 0 \qquad \forall\, j \in N. \tag{4.10}$$

The dual variables have nice economic interpretations. We can think p_j to be the price of good j. Assume that if a buyer $i \in M$ gets a bundle of goods S, then he pays $\sum_{j \in S} p_j$, and the payoff he gets is $\pi_i(S) := v_i(S) - \sum_{j \in S} p_j$. Define $\pi_i(\emptyset) = 0$ for all $i \in M$. Hence, constraint (4.8) can be written as $\pi_i \geq \pi_i(S)$ for all $i \in M$ and for all $S \in \Omega$. Now, let us write the complementary slackness conditions. Let x be a feasible solution of (**CA-LP**) and (p, π) be a feasible solution of (**CA-DLP**). They are optimal if and only if

$$x_i(S)\Big[\pi_i - \pi_i(S)\Big] = 0 \qquad \forall\, i \in M,\ \forall\, S \in (\Omega \setminus \emptyset) \tag{4.11}$$

$$x_i(\emptyset)\pi_i = 0 \qquad \forall\, i \in M \tag{4.12}$$

$$p_j\Big[1 - \sum_{i \in M} \sum_{S \in \Omega : j \in S} x_i(S)\Big] = 0 \qquad \forall\, j \in N. \tag{4.13}$$

Equation (4.12) says that if $x_i(\emptyset) = 1$, then $\pi_i = 0 = \pi_i(\emptyset)$. Also, Equation (4.11) says that if $x_i(S) = 1$, then $\pi_i = \pi_i(S)$. Due to dual feasibility, we know that $\pi_i \geq \pi_i(S)$. Hence, at optimality $\pi_i = \max_{S \in \Omega} \pi_i(S)$ for every buyer $i \in M$ - this denotes the maximum payoff of buyer i at a given price vector p. Hence, an optimal solution of (**CA-DLP**) can be described by just $p \in \mathbb{R}^n_+$.

We will introduce some more notations. **Demand set** of a buyer $i \in M$ at a price vector $p \in \mathbb{R}^n_+$ is defined as $D_i(p) = \{S \in \Omega : \pi_i(S) \geq \pi_i(T)\ \forall\, T \in \Omega\}$.

In the example in Table 4.3 above, consider a price vector $p = (4, 4)$ (i.e., price of good 1 is 4, good 2 is 4, and bundle 1,2 is 4+4=8). At this price vector, $D_1(p) = \{\{1, 2\}\}$

and $D_2(p) = \{\{2\}, \{1, 2\}\}$. Consider another price vector $p' = (5, 6)$. At this price vector, $D_1(p') = \{\{1\}\}$, $D_2(p') = \{\{2\}\}$.

Definition **25** *A price vector $p \in \mathbb{R}^n_+$ and an allocation X is called a* **Walrasian equilibrium** *if*

1. *$X_i \in D_i(p)$ for all $i \in M$ (every buyer gets a bundle with maximum payoff),*

2. *$p_j = 0$ for all $j \in N$ such that $j \notin \cup_{i \in M} X_i$ (unassigned goods have zero price).*

The price vector $p' = (5, 6)$ along with allocation $(\{1\}, \{2\})$ is a Walrasian equilibrium of the example in Table 4.3 since $\{1\} \in D_1(p')$, $\{2\} \in D_2(p')$.

Theorem **41** *(p, X) is a Walrasian equilibrium if and only if X corresponds to an integral optimal solution of (**CA-LP**) and p corresponds to an optimal solution of (**CA-DLP**).*

Proof: Suppose (p, X) is a Walrasian equilibrium. Then p generates a feasible solution of (**CA-DLP**) - this feasible solution is generated by setting $\pi_i = \max_{S \in \Omega}[v_i(S) - \sum_{j \in S} p_j]$ for all $i \in M$, and X corresponds to a feasible **integral** solution of (**CA-LP**) - this feasible solution is generated by setting $x_i(X_i) = 1$ for all $i \in M$ and setting zero all other x variables. Now, $X_i \in D_i(p)$ for all $i \in M$ implies that $\pi_i = \pi_i(X_i)$ for all $i \in M$, and this further implies that Equations (4.11) and (4.12) is satisfied. Similarly, $p_j = 0$ for all $j \in N$ that are not assigned in X. This means Equation (4.13) is satisfied. Since complementary slackness conditions are satisfied, these are also optimal integral solutions.

Now, suppose p is an optimal solution of (**CA-DLP**) and X corresponds to an integral optimal solution of (**CA-LP**). Then, the complementary slackness conditions imply that the conditions for Walrasian equilibrium is satisfied. Hence, (p, X) is a Walrasian equilibrium. ■

Another way to state Theorem 41 is that a Walrasian equilibrium exists if and only if an optimal solution of (**CA-LP**) gives an efficient allocation (an optimal solution of (**CA-IP**). This is because if a Walrasian equilibrium (p, X) exists, then X is an optimal solution of (**CA-LP**) that is integral. Hence it is an optimal solution of (**CA-IP**) or an efficient allocation. So, every allocation corresponding to a Walrasian equilibrium is an efficient allocation.

There are combinatorial auction problems where a Walrasian equilibrium may not exist. Consider the example in Table 4.2. It can be verified that this example does not have a Walrasian equilibrium. Suppose there is a Walrasian equilibrium (p, X). By Theorem 41 and the earlier discussion, X is efficient, i.e, $X = (\emptyset, \{1, 2\})$. Since $X_1 = \emptyset$, by definition of

Walrasian equilibrium $\pi_1(\emptyset) = 0 \geq \pi_1(\{1\}) = 8 - p_1$, i.e., $p_1 \geq 8$. Similarly, $p_2 \geq 11$. This means $p_1 + p_2 \geq 19$. But $X_2 = \{1, 2\}$, and $\pi_2(\{1, 2\}) = 18 - (p_1 + p_2) \leq -1 < 0 = \pi_2(\emptyset)$. Hence $\{1, 2\} \notin D_2(p)$. This is a contradiction since (p, X) is a Walrasian equilibrium. This also follows from the fact the LP relaxation of (**CA-IP**) does not have integral optimal solution.

4.6 Submodular Optimization

In continuous optimization problems, concave maximization plays an important role. Besides standard tools like differentiation, various methods to find the maximum of a concave function exist. The main objective of this section is to give a counterpart of concave maximization in the framework of discrete optimization.

We study a class of optimization problems for which a *greedy* solution exists. Further, whenever the primitives of the problems are integral, then the greedy solution is also an integral solution. This illustrates an important class of problems where the TU property need not hold, but still we achieve the integrality of extreme points.

We will be talking about set functions. Let N be a finite set and $\mathcal{P}(N)$ be the set of all subsets of N. We will be interested in functions $f : \mathcal{P}(N) \to \mathbb{R}$, where we will normalize $f(\emptyset) = 0$ throughout.

Definition **26** *A function $f : \mathcal{P}(N) \to \mathbb{R}$ is* **submodular** *if for all $A, B \subseteq N$, we have*

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B).$$

A function $f : \mathcal{P}(N) \to \mathbb{R}$ is **non-decreasing** *if for all $A, B \subseteq N$ with $A \subseteq B$, we have $f(A) \leq f(B)$.*

Consider an example with $N = \{a, b, c\}$ and $f(a) = f(b) = f(c) = 1$, $f(a, b) = 3$, $f(b, c) = 2$, $f(c, a) = 4$, $f(a, b, c) = 4$. The function f is non-decreasing. However, it is not submodular since $f(a) + f(b) = 2 < 3 = f(a, b)$.

An alternate definition of a submodular function is the following.

Theorem **42** *A set function $f : \mathcal{P}(N) \to \mathbb{R}$ is submodular if and only if for all $A, B \subseteq N$ with $A \subseteq B$ and for all $b \notin B$, we have*

$$f(A \cup \{b\}) - f(A) \geq f(B \cup \{b\}) - f(B).$$

The theorem says that the "marginal" contribution to a smaller subset is larger than to a bigger subset. This is consistent with the idea that the derivative of a concave function is non-increasing. *Proof*: Suppose $f : \mathcal{P}(N) \to \mathbb{R}$ is a submodular function. Pick $A \subseteq B \subseteq N$ and $b \notin B$. Note that $A \cup B = B$ and $A \cap B = A$. Using submodularity we get, $f(A \cup \{b\}) + f(B) \geq f(B \cup \{b\}) + f(A)$, which gives the desired inequality.

For the converse, pick any $A, B \subseteq N$. Note that if $B \subseteq A$, then there is nothing to prove. Hence, suppose $B \setminus A = \{b_1, \ldots, b_k\}$. Now,

$$\begin{aligned}
f(A \cup B) - f(A) &= \big[f(A \cup B) - f((A \cup B) \setminus \{b_1\})\big] + \big[f((A \cup B) \setminus \{b_1\}) - f((A \cup B) \setminus \{b_1, b_2\})\big] \\
&+ \ldots + \big[f((A \cup B) \setminus \{b_1, \ldots, b_{k-1}\}) - f(A)\big] \\
&\leq \big[f(B) - f(B \setminus \{b_1\})\big] + \big[f(B \setminus \{b_1\}) - f(B \setminus \{b_1, b_2\})\big] \\
&+ \ldots + \big[f((A \cap B) \cup \{b_k\}) - f(A \cap B)\big] \\
&= f(B) - f(A \cap B),
\end{aligned}$$

which gives the desired submodular inequality. ■

An interesting consequence of submodularity and non-decreasingness is the following.

Lemma 27 *If a set function $f : \mathcal{P}(N) \to \mathbb{R}$ is submodular, then for all $A, B \subseteq N$, we have*

$$f(A \cup B) - f(B) \leq \sum_{a \in A \setminus B} \big[f(B \cup \{a\}) - f(B)\big].$$

Further, if f is submodular and non-decreasing then

$$f(A) - f(B) \leq \sum_{a \in A \setminus B} \big[f(B \cup \{a\}) - f(B)\big].$$

Proof: Let $A \setminus B = \{a_1, \ldots, a_k\}$. Then,

$$\begin{aligned}
f(A \cup B) - f(B) &= \sum_{j=1}^{k} \Big[f(B \cup \{a_1, \ldots, a_j\}) - f(B \cup \{a_1, \ldots, a_{j-1}\})\Big] \\
&\leq \sum_{j=1}^{k} \Big[f(B \cup \{a_j\}) - f(B)\Big] \\
&= \sum_{a \in A \setminus B} \Big[f(B \cup \{a\}) - f(B)\Big]
\end{aligned}$$

The second part follows from the fact that non-decreasing f implies $f(A) - f(B) \leq f(A \cup B) - f(B)$. ■

4.6.1 Examples

We now give some examples of submodular functions.

1. Let S be the set of columns of a matrix. Let $r(X)$ denote the rank of the matrix formed by a set of columns $X \subseteq S$. Then, r is a submodular function.

2. Let $G = (N, E, w)$ be a weighted undirected graph, where weights are non-negative. For any $S \subseteq N$, let $(S, N \setminus S)$ be a cut of the graph and $W(S)$ denote the sum of weights of edges crossing this cut. Then, W is a submodular function.

3. Let $G = (N, E)$ be an undirected graph. For any $S \subseteq E$, let $r(S)$ denote the highest number of edges in S that do not include a cycle. Then, r becomes a submodular function.

4.6.2 Optimization

We will be interested in the following polyhedron associated with a submodular function f:

$$P^f := \{x \in \mathbb{R}^n_+ : \sum_{j \in S} x_j \leq f(S) \; \forall \; S \subseteq N\}.$$

We will attach a linear program with the feasible set being this polyhedron. In particular, we will consider the following linear program:

$$\max_{x \in P^f} \sum_{j \in N} c_j x_j.$$

We will denote this linear program as $\mathbf{LP}^f$. We show that a greedy algorithm gives an optimal solution of $\mathbf{LP}^f$. The greedy algorithm works as follows.

1. Order the variables $c_1 \geq c_2 \geq \ldots \geq c_r > 0 \geq c_{r+1} \geq \ldots \geq c_n$.

2. Set $x_i := f(S^i) - f(S^{i-1})$ for all $i \in \{1, \ldots, r\}$ and $x_j = 0$ for all $j > r$, where $S^i = \{1, \ldots, i\}$ for all $i \in \{1, \ldots, r\}$ and $S^0 = \emptyset$.

Theorem **43** *The greedy algorithm finds an optimal solution of* $\mathbf{LP}^f$ *for any submodular and non-decreasing set function* $f : \mathcal{P}(N) \to \mathbb{R}$.

Proof: As f is non-decreasing, $x_i = f(S^i) - f(S^{i-1}) \geq 0$ for all $i \in \{1, \ldots, r\}$. Also, for each $T \subseteq N$, we note that

$$\begin{aligned}\sum_{j\in T} x_j &= \sum_{j\in T\cap S^r} \left[f(S^j) - f(S^{j-1})\right] \\ &\leq \sum_{j\in T\cap S^r} \left[f(S^j\cap T) - f(S^{j-1}\cap T)\right] \\ &\leq \sum_{j\in S^r} \left[f(S^j\cap T) - f(S^{j-1}\cap T)\right] \\ &= f(S^r\cap T) - f(\emptyset) \leq f(T),\end{aligned}$$

where the first inequality followed from submodularity and the second and the last one from non-decreasingness of f. This shows that the greedy solution is a feasible solution. The objective function value from the greedy solution is

$$\sum_{i=1}^{r} c_i\left[f(S^i) - f(S^{i-1})\right].$$

Now, to prove optimality, we consider the dual of the linear program $\mathbf{LP}^f$. The dual of this linear program is $\min_y \sum_{S\subseteq N} f(S)y_S$ subject to the constraints that $\sum_{S:j\in S} y_S \geq c_j$ for all $j \in N$ and $y_S \geq 0$ for all $S \subseteq N$.

We first give a solution to the dual. Let $y_{S^i} = c_i - c_{i+1}$ for $i = 1, \ldots, r-1$, $y_{S^r} = c_r$, and $y_S = 0$ for all other S. Note that non-negativity constraint is satisfied. Further, for any $j \leq r$, $\sum_{S:j\in S} y_S \geq \sum_{i=j}^{r} y_{S_i} = c_r + \sum_{i=j}^{r-1}[c_i - c_{i+1}] = c_j$. For any $j < r$, $\sum_{S:j\in S} y_S \geq 0 \geq c_j$. Thus, the $\{y_S\}_S$ is a feasible solution. The objective function value of the dual from this feasible solution is $\sum_{S\subseteq N} f(S)y_S = \sum_{i=1}^{r-1}[c_i - c_{i+1}]f(S^i) + c_r f(S^r) = \sum_{i=1}^{r} c_i\left[f(S^i) - f(S^{i-1})\right]$.

Hence, the objective function value of the primal feasible solution and the dual feasible solution is the same. By strong duality theorem, these are optimal solutions. ■

Two interesting observations from the proof of Theorem 43 emerge. First, if $f(S\cup\{a\}) - f(S)$ is integral for all $S \subseteq N$ and $a \in N \setminus S$, then there is a greedy integral feasible solution that is optimal. Second, if $f(S \cup \{a\}) - f(S) \in \{0, 1\}$ for all $S \subseteq N$ and $a \in N \setminus S$, then the greedy algorithm gives a $\{0, 1\}$ feasible solution that is optimal. In this case, f is called a **submodular rank function**.

This point further highlights an important point that even though the constraint matrix is not TU, we have found an interesting class of problems where LP relaxation solves the integer program.

The proof also highlights another interesting point. It identifies the extreme points of the polyhedron P^f. To see this consider, $N = \{1, \ldots, n\}$. Choose any $r \in N$ and set

$x_i = f(S^i) - f(S^{i-1})$ for all $i \le r$ and $x_i = 0$ for all $i > r$. By following the steps in the proof, it is easy to verify that x is a feasible point in P^f. Each of these feasible solutions is determined by a choice of r and an ordering of elements of N (we ordered it as $1, \ldots, n$, but you can choose another order). Since these are optimal solutions of the linear program by choosing an appropriate set of coefficients $\{c_j\}_j$, they are extreme points. With some effort, it is easy to show that they are the only extreme points.

We also noticed that in the submodular optimization problem, if the coefficients of the objective function is integral, then the dual optimal solution is also integral. This leads to an interesting class of problems.

DEFINITION **27** *A set of linear inequalities $Ax \le b$ is called* **totally dual integral (TDI)** *if for all $\{c_j\}_{j \in N}$, where c_j is integral, such that the linear program $\max cx : Ax \le b$ has an optimal solution, the dual linear program $\min by : yA = c, y \ge 0$ has optimal solution with y integral.*

We saw that the polyhedron P^f associated with a non-decreasing submodular function f is TDI. The following theorem, whose proof we skip, gives another way to verify that certain LP relaxations always give integral optimal solution.

THEOREM **44** *If $Ax \le b$ is TDI, b is an integer vector, and $P = \{x \in \mathbb{R}^n : Ax \le b\}$ has extreme points, then all extreme points of P are integral.*

If A is TU, then this is of course true, since transpose of A is a TU matrix, and that ensures that the dual has integral solution.

4.7 A Short Introduction to Matroids

We now formally introduce a notion that can be thought of as an extension of independence from linear algebra to arbitrary discrete systems. Since we encounter such systems quite often in discrete optimization, it is an extremely useful modeling tool.

Let E be any arbitrary set. In matroid theory, E is called a *ground set*. Let $\mathbb{I}$ be a collection of subsets of E, i.e., every $S \in \mathbb{I}$ is a subset of E. Members of $\mathbb{I}$ are called *independent sets*.

DEFINITION **28** *The pair $(E, \mathbb{I})$ is called a* **matroid** *if*

I1 NON-EMPTY. *$\mathbb{I}$ is non-empty.*

I2 HEREDITARY. *If $S \in \mathbb{I}$ and $T \subset S$, then $T \in \mathbb{I}$.*

I3 Augmentation. *If $S, T \in \mathbb{I}$ with $|T| = |S| + 1$, then there is an element $x \in T \setminus S$ such that $S \cup \{x\} \in \mathbb{I}$.*

If $(E, \mathbb{I})$ satisfies only I1 and I2, then it is called an **independence system**.

An independence system $(E, \mathbb{I})$ satisfies **equi-maximality** property if for all $X \subseteq E$, all maximal members of $\{S : S \in \mathbb{I}, S \subseteq X\}$ have the same number of elements.

Theorem **45** *An independence system $(E, \mathbb{I})$ is a matroid if and only if it satisfies equi-maximility property.*

Proof: Suppose $(E, \mathbb{I})$ is a matroid. Let $X \subseteq E$, and define $\mathbb{I}_X = \{S : S \in \mathbb{I}, S \subseteq X\}$. Let $S, T \in \mathbb{I}_X$ be two maximal members. Assume for contradiction, $|S| < |T|$. By hereditary, there exists $K \subseteq T$ such that $|K| = |S| + 1$. By augmentation, there exists $x \in K \setminus S$ such that $S \cup \{x\} \in \mathbb{I}$. Clearly, $S \cup \{x\} \in \mathbb{I}_X$. This contradicts that S is maximal.

Suppose $(E, \mathbb{I})$ is an independence system which satisfies equi-maximality property. Let $S, T \in \mathbb{I}$ with $|T| = |S|+1$. Let $X = S \cup T$. Note that since $|T| > |S|$, due to equi-maximality property S cannot be a maximal member of $\mathbb{I}_X$. This means there exists a $x \in T \setminus S$ such that $S \cup \{x\} \in \mathbb{I}$. ■

We give some examples (types) of matroids.

1. Uniform Matroid. Let E be a set with n-elements and for a fixed integer r with $0 \leq r \leq n$, let $\mathbb{I} = \{S \subseteq E : |S| \leq r\}$. It is easy to verify that $(E, \mathbb{I})$ is a matroid. Such a matroid is called a *uniform matroid*.

2. Graphic Matroid. Let $G = (N, E)$ be a connected graph, where N is the set of vertices and E is the set of edges. Let $\mathbb{I} = \{S : S \subseteq E$ and S does not contain a cycle of $G\}$. Then, $(E, \mathbb{I})$ is a matroid. Such a matroid is called a *graphic matroid*.

   We show that every graphic matroid is indeed a matroid. The non-empty and hereditary properties are obvious. For the augmentation property, take $S, T \in \mathbb{I}$ with $|T| = |S| + 1$. Since both S and T are sets of edges which do not form a cycle (or trees), we *span* $|S| + 1$ and $|T| + 1$ vertices of G by the edges in S and T respectively. This means, there is a vertex of G that is spanned by T but not by S. Let us say $i \in N$ be that vertex, and $x = \{i, j\}$ be the edge in T. Adding x to S obviously gives a tree since i does not belong to the span of S. Hence, augmentation is satisfied.

3. Linear Matroid. Let E be the set of columns of a matrix. If $\mathbb{I}$ consists of all subsets of E that are independent, then it forms a matroid.

4. Partition Matroid. For partition matroid, we are given a partition $E_1, \ldots, E_k$ of E and positive numbers $m_1, \ldots, m_k$. Now, $\mathbb{I} = \{X \subseteq E : |X \cap E_i| \leq m_i \; \forall \; i \in \{1, \ldots, k\}\}$. This forms a matroid.

4.7.1 Equivalent ways of Defining a Matroid

We introduce some equivalent ways of defining a matroid. Suppose $(E, \mathbb{I})$ is a matroid. Then, any $S \subseteq E$ such that $S \in \mathbb{I}$ is called a **independent set**. A maximal independent set is called a **basis**. By Theorem 45, every basis has the same cardinality, and this is called the **rank** of a matroid.

We can define a matroid using its bases.

Theorem **46** *Let $\mathbb{B}$ be the set of subsets of a finite set E. Then, $\mathbb{B}$ is the collection of bases of a matroid on E if and only if $\mathbb{B}$ satisfies the following properties.*

B1 The set $\mathbb{B}$ is non-empty.

B2 Suppose $B_1, B_2 \in \mathbb{B}$ and $x \in B_1 \setminus B_2$. Then, there exists $y \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathbb{B}$.

Proof: Suppose $(E, \mathbb{I})$ is a matroid, and let $\mathbb{B}$ be the collection of its bases. Since $\emptyset \in \mathbb{I}$, $\mathbb{B}$ is non-empty. Now, choose $B_1, B_2 \in \mathbb{B}$. By Theorem 45, $|B_1| = |B_2|$. Choose $x \in B_1 \setminus B_2$, and let $S := B_1 \setminus \{x\}$. Clearly, S has at least one element, and $S \in \mathbb{I}$. But $|B_2| = |S| + 1$. By augmentation property, there exists $y \in B_2 \setminus S$ such that $S \cup \{y\} \in \mathbb{I}$. Note that $y \in B_2 \setminus B_1$. Also, $|S \cup \{y\}|$ is the rank of the matroid. Hence, by Theorem 45, $S \cup \{y\}$ is a basis of matroid $(E, \mathbb{I})$.

Now, consider a set $\mathbb{B}$ which satisfies properties B1 and B2. Define the set $\mathbb{I}$ as follows:

$$\mathbb{I} := \{K \subseteq E : \text{ there exists } B \in \mathbb{B} \text{ with } K \subseteq B\}.$$

Since $\mathbb{B}$ is non-empty, $\emptyset \in \mathbb{I}$. Further if $S \subsetneq T$ and $T \in \mathbb{I}$, then $S \in \mathbb{I}$. So, I1 and I2 is satisfied.

First, we show that for any $B_1, B_2 \in \mathbb{B}$, we have $|B_1| = |B_2|$. To see this, suppose

$$B_1 = \{x_1, \ldots, x_k\}, \quad B_2 = \{y_1, \ldots, y_k, y_{k+1}, \ldots, y_{k+q}\}.$$

Pick $x \in B_1 \setminus B_2$. By property $B2$, we know that there is $y \in B_2 \setminus B_1$ such that $B_1 \setminus \{x\} \cup \{y\} \in \mathbb{B}$. Let $B' := B_1 \setminus \{x_1\} \cup \{y\}$. Pick $x' \in B' \setminus B_2$. By property $B2$, we know that there is $y' \in B_2 \setminus B'$ such that $B' \setminus \{x'\} \cup \{y'\} \in \mathbb{B}$. We can repeat this procedure till we have found

$\bar{B} \in \mathbb{B}$ and $\bar{B} \cup B_2$. Since $|B_2| > |B_2|$, by definition the procedure must terminate with $\bar{B}$ such that $\bar{B} \subsetneq B_2$. This contradicts property $B2$ since if we pick $x \in B_2 \setminus \bar{B}$, there is no $y \in \bar{B} \setminus B_2$ such that $B_2 \setminus \{x\} \cup \{y\}$ belongs to $\mathbb{B}$.

Now, consider $S, T \in \mathbb{I}$ such that $|S| = |T| + 1$. Let $S \subseteq B_1$ and $T \subseteq B_2$ for some $B_1, B_2 \in \mathbb{B}$. Let

$$\begin{aligned} S &= \{x_1, \ldots, x_k\} \\ B_1 &= \{x_1, \ldots, x_k, b_1, \ldots, b_q\} \\ T &= \{y_1, \ldots, y_k, y_{k+1}\} \\ B_2 &= \{y_1, \ldots, y_k, y_{k+1}, c_1, \ldots, c_{q-1}\}. \end{aligned}$$

Consider $X = B_1 \setminus \{b_q\}$. By property B2, there exists $z \in B_2 \setminus B_1$ such that $X \cup \{z\} \in \mathbb{B}$. Hence, $S \cup \{z\} \in \mathbb{I}$. If $z \in T$, then I3 is satisfied. If $z \notin T$, set $X_1 = (X \cup \{z\}) \setminus \{b_{q-1}\}$. Again, there exists $z_1 \in B_2 \setminus (X \cup \{z\})$ such that $X_1 \cup \{z_1\} \in \mathbb{B}$. If $z_1 \in T$, then again I3 is satisfied. Else, we repeat the procedure again. Since $|\{b_1, \ldots, b_q\}| > |\{c_1, \ldots, c_{q-1}|$, after at most q such step, we will replace by an element in T. This will imply that I3 is satisfied. ■

Another way to define a matroid is using the rank axioms. The **rank function** of a matroid $(E, \mathbb{I})$ is a function $r : 2^E \to \mathbb{Z}$ defined by

$$r(S) = \max\{|X| : X \subseteq S, X \in \mathbb{I}\} \qquad \forall\, S \subseteq E.$$

The **rank** of a matroid is simply $r(E)$, i.e., the size of the largest independent set. Note that r is non-decreasing.

THEOREM **47** *The rank function of a matroid is submodular and non-decreasing.*

Proof: Let $S, T \subseteq E$. Let X be a maximum independent set of $S \cap T$ and Y be a maximum independent set of $S \cup T$. Notice that since $(S \cap T) \subseteq (S \cup T)$, we can ensure that $X \subseteq Y$. So, we have $r(S \cap T) = |X|$ and $r(S \cup T) = |Y|$ with $X \subseteq Y$.

Since $Y \in \mathbb{I}$, by hereditary property, $Y \cap S$ and $Y \cap T$ also belong to $\mathbb{I}$. Hence, we have

$$r(S) \geq |Y \cap S|, \qquad r(T) \geq |Y \cap T|.$$

Also, note that since $X \subseteq Y$ and $X \subseteq (S \cap T)$, we have that

$$X = (X \cap Y) \subseteq (S \cap T) \cap Y.$$

Using these two inequalities, we get that

$$\begin{aligned} r(S) + r(T) &\geq |Y \cap S| + |Y \cap T| \\ &= |Y| + |Y \cap (S \cap T)| \\ &\geq |Y| + |X| \\ &= r(S \cup T) + r(S \cap T), \end{aligned}$$

where the first equality followed from the fact that $Y \subseteq (S \cup T)$. ■

4.7.2 The Matroid Polytope

Let $E = \{1, \ldots, n\}$ and $(E, \mathbb{I})$ be a matroid with r being its rank function. Consider the following system of inequalities with variables $x \equiv (x_1, \ldots, x_n)$

$$\mathcal{P} = \{x \in \mathbb{R}^n_+ : \sum_{j \in S} x_j \leq r(S) \ \forall \ S \subseteq E\}.$$

Notice that $\mathcal{P}$ is a bounded polyhedron, i.e., a polytope.

The main theorem of this section characterizes the extreme points of $\mathcal{P}$.

Theorem **48** *If $(x_1, \ldots, x_n)$ is an extreme point of the polytope $\mathcal{P}$ corresponding to matroid $(E, \mathbb{I})$, then there exists $S \in \mathbb{I}$ such that $x_j = 1$ for all $j \in S$ and $x_j = 0$ for all $j \notin S$*

Proof: Let $x^* = (x^*_1, \ldots, x^*_n)$ be an extreme point of $\mathcal{P}$. Since it is an extreme point, there exists some $c_1, \ldots, c_n$ such that $\max \sum_{j \in E} c_j x_j$ subject to $(x_1, \ldots, x_n) \in \mathcal{P}$ has a solution at x^* [1]. But this is equivalent to solving a submodular optimization problem. We know the optimal solution can be obtained by the greedy algorithm (Theorem 43). But the greedy algorithm choses $S \subseteq E$ and sets $x^*_j = 0$ if $j \notin S$. For each $j \in S$, it sets $x^*_j = r(S^j) - r(S^{j-1})$, where $S^j = \{1, \ldots, j\}$. Notice that since r is the rank function for every $j \in S$, $r(S^j) - r(S^{j-1}) = 1$ if $S^j \in \mathbb{I}$ and $r(S^j) - r(S^{j-1}) = 0$ if S^j is not independent. Hence, for some $T \subseteq S$, where $T \in \mathbb{I}$, we have $x^*_j = 1$ if and only if $j \in T$. This proves that x^* is integral and corresponds to an independent set of the polytope $(E, \mathbb{I})$. This proves that $\mathcal{P}$ is a matroid polytope. ■

As an exercise, you are encouraged to think how you can use the result in Theorem 48 to solve optimization problems corresponding to specific type of matroids we have discussed

[1]Though we have not shown this formally, it is true. Intuitively, for every extreme point, we can always construct a hyperplane which when moved *up* will touch that extreme point.

earlier. For instance, finding maximum weight spanning tree is related to optimizing over the graphic matroid and the greedy algorithm corresponds to a well known algorithm we had discussed earlier.