

BASIC GRAPH THEORY

WITH APPLICATIONS TO ECONOMICS

Debasis Mishra*

February 23, 2011

1 WHAT IS A GRAPH?

Let $N = \{1, \dots, n\}$ be a finite set. Let E be a collection of ordered or unordered pairs of distinct¹ elements from N . A **graph** G is defined by (N, E) . The elements of N are called **vertices** 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 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, $E = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$. In the directed graph, $E = \{(1, 2), (1, 3), (2, 3), (3, 2), (3, 4), (4, 1)\}$.

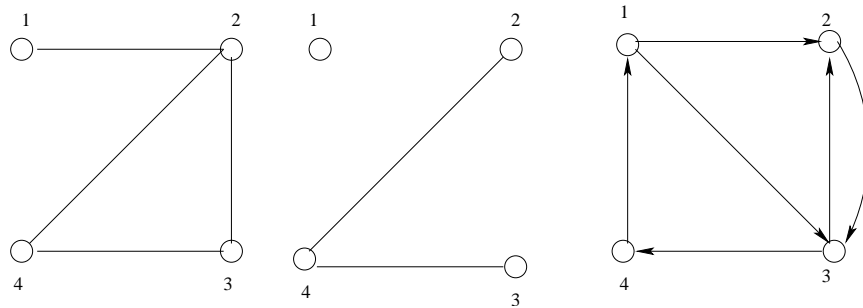


Figure 1: Examples of graph

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

¹In 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, $w : E \rightarrow \mathbb{R}$. 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 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 situation 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 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?

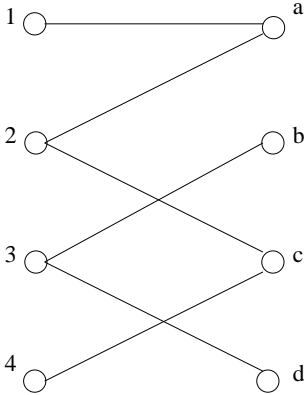


Figure 2: A bipartite graph

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.

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, 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, \dots, i^k) such that $\{i^j, i^{j+1}\} \in E$ for all $1 \leq j < k$. The path (i^1, \dots, 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 is not connected.

A **cycle** is a sequence of vertices $(i^1, \dots, i^k, i^{k+1})$ with $k > 2$ such that $\{i^j, i^{j+1}\} \in E$ for all $1 \leq j \leq k$, (i^1, \dots, i^k) is a path, and $i^1 = i^{k+1}$. In the leftmost graph in Figure 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 are not trees, but the second and third graph in Figure 3 are trees. A graph may contain several trees (i.e. connected acyclic subgraphs). The spanning tree of a connected graph is a subgraph (N, E') such that (N, E') is a tree. Note that $E' \subseteq E$ and a spanning tree does not contain a cycle. By definition, every tree (N', E') is a spanning tree of graph (N', E') .

Figure 3 shows a connected graph (which is not a tree) and two of its spanning trees.

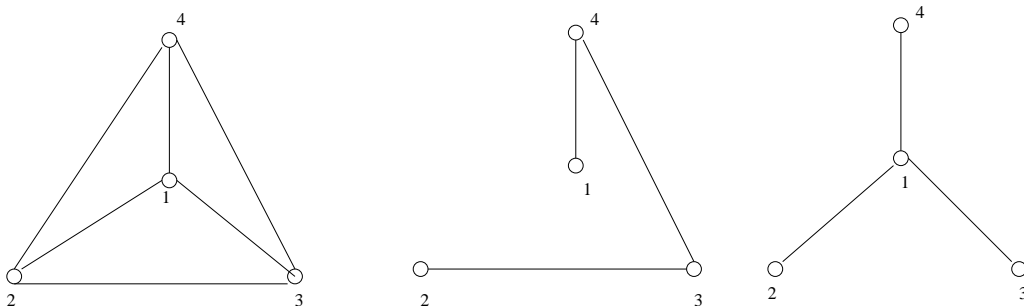


Figure 3: Spanning trees of a graph

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.*
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 $P_1 = (i, i^1, \dots, i^k, j)$ and $P_2 = (i, j^1, \dots, j^q, j)$. Then, consider the following sequence of vertices: $(i, i^1, \dots, i^k, j, j^q, \dots, j^1, i)$. 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.
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 another tree of subgraph $(N' \setminus \{i\}, E' \setminus \{i, j\})$. 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: Call a subgraph of a graph G a **component** of G if it is maximally connected, i.e., $G'' = (N'', E'')$ is a component of $G = (N, E)$ if G'' is connected and there does not exist vertices $i \in N''$ and $j \in N \setminus N''$ such that $\{i, j\} \in E$.

Clearly, any graph can be partitioned into its components with a connected graph having one component, which is the same graph. Now, consider a cycle-free graph $G' = (N, E')$ and let G^1, \dots, 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, \dots, G^q is

$$(n_1 - 1) + \dots + (n_q - 1) = (n_1 + n_2 + \dots + n_q) - q = \#N - q.$$

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 $n - 1$, which equals the number of edges in $E' \setminus \{\{i, j\}\}$. By our induction hypothesis, G'' has no cycles. Hence, G' cannot have any cycle. ■

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, $w \in \mathbb{R}_+^{\#E}$. The **minimum cost spanning tree (MCST) problem** is to find a spanning tree of minimum cost in graph G . Figure 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.

There are many *greedy* algorithms that find an MCST. 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

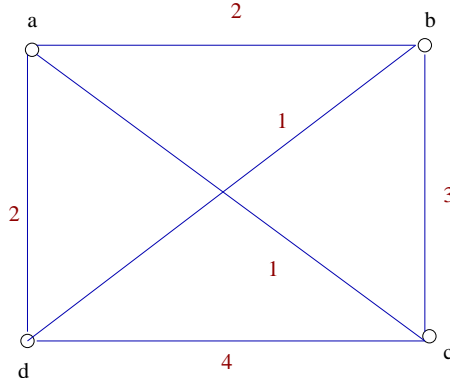


Figure 4: The minimum cost spanning tree problem

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). In Figure 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 (N, A) is not a spanning tree, then find an edge $\{i, j\}$ that is safe for A . Set $A \leftarrow A \cup \{\{i, j\}\}$ and repeat from Step 2.
3. If (N, A) is a spanning tree, then return (N, A) .

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 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. The algorithm we discuss is due to Kruskal, and hence called the Kruskal's algorithm. 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 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.

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

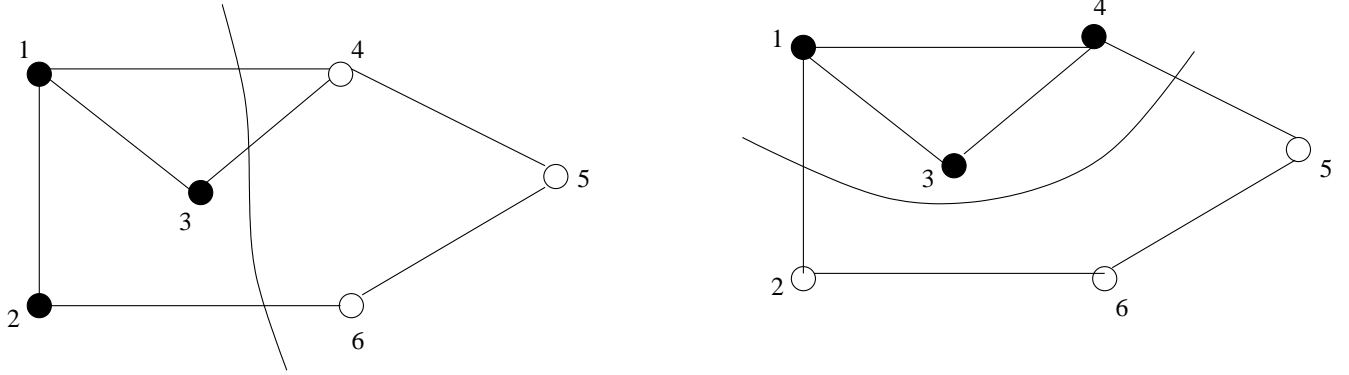


Figure 5: Cuts in a graph

THEOREM 1 *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. Consider the following 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 A 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 1, in every step of the algorithm, we add an edge which is safe. This means the output of the algorithm contains $\#N - 1$ edges and no cycles. By Lemma 2, this is a spanning tree. By Theorem 1, this is an MCST.

We apply this algorithm to the example in Figure 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.

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.

4.1 COOPERATIVE GAMES

Let N be the set of agents. A subset $S \subseteq N$ of agents is called a coalition. Let Ω be the set of all coalitions. A cooperative game is a tuple (N, c) where N is a finite set of agents and c is a characteristic function defined over set of coalitions Ω , i.e., $c : \Omega \rightarrow \mathbb{R}$. The number $c(S)$ can be thought to be the cost incurred by coalition S when they cooperate ².

The problem is to divide the total cost $c(N)$ amongst the agents in N when they cooperate. We give an example of a cooperative game and analyze it below. Before that, we describe a well known solution concept of core.

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

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

²Cooperative games can be defined with value functions also, in which case notations will change, but ideas remain the same.

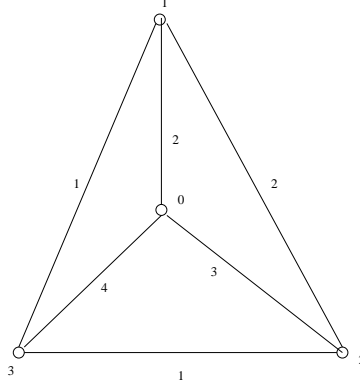


Figure 6: An MCST game

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.

4.2 THE MINIMUM COST SPANNING TREE GAME

The minimum cost spanning tree game (MCST game) is defined by set of agents $N = \{1, \dots, n\}$ and a **source** agent 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 source, they form an MCST using edges between themselves. 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 6. 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 this the Bird allocation - named after the inventor of this allocation.

Figure 7 gives an example with 5 agents (the edges not show have very high cost). The MCST is shown with red edges in Figure 7. 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

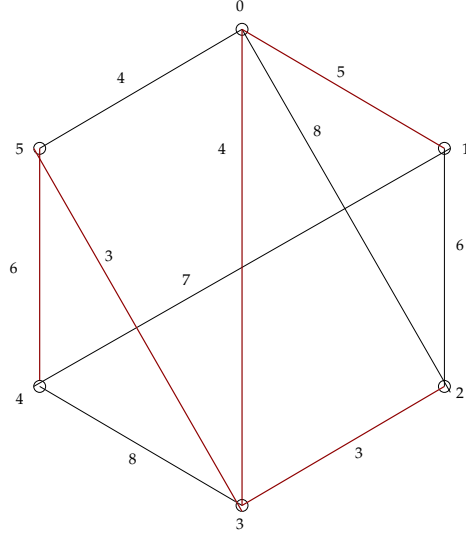


Figure 7: Bird allocation

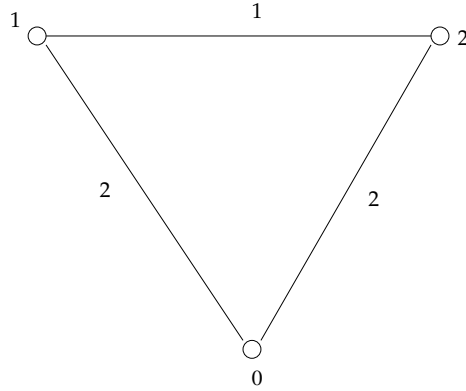


Figure 8: More than one Bird allocation

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 8 illustrates that. There are two MCSTs in Figure 8 - 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 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 2.

THEOREM 2 *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 $(\#N - 1)$ edges as the MCST (N, T) . 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 $\#N - 1$ 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 there are more than one Bird allocations, 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).

5 HALL'S MARRIAGE THEOREM

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 be 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 $E = \{\{i, j\} : i \in B, j \in H_i\}$. We will denote such a bipartite graph 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. We ask whether there exists a matching with $\#B$ edges, i.e., where all the buyers are matched.

Figure 9 shows a bipartite graph with a matching: $\{\{1, b\}, \{2, a\}, \{3, d\}, \{4, c\}\}$.

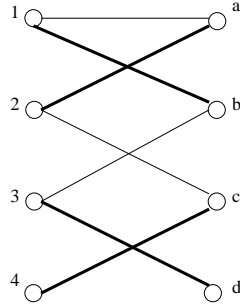


Figure 9: A bipartite graph with a matching

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 3 *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 $D(S) = \cup_{i \in S} H_i$.*

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.

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.

6.1 M -AUGMENTING PATH

We start with the notion of augmenting path in an arbitrary undirected graphs. 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 10 and 11 show two matchings in a graph - dark edges represent a matching.

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

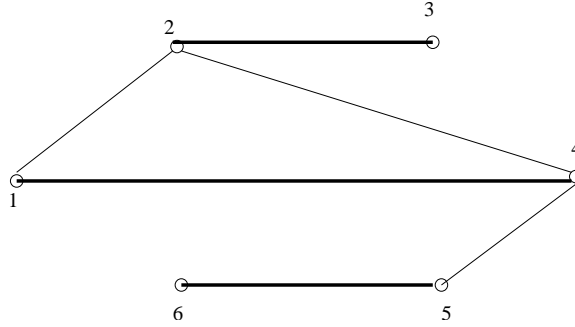


Figure 10: A matching in a graph

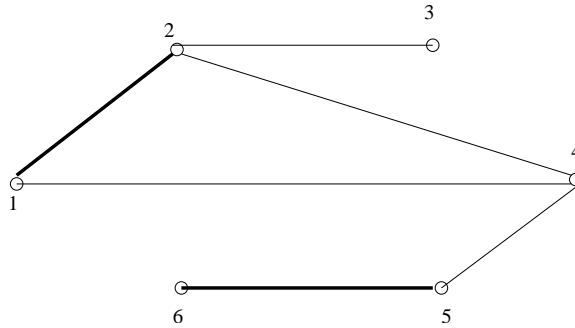


Figure 11: A matching in a graph

in S is an endpoint of some edge in X . If (i_1, i_2, \dots, i_k) is a path, then i_1 and i_k are called endpoints of this path.

DEFINITION 1 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 alternately in and out of M .

Note that an M -augmenting path need not contain all the edges in M . Suppose an M -augmenting path contains k edges from $E \setminus M$. Note that $k \geq 1$ as 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 12 and 13 show two matchings and their respective M -augmenting paths.

DEFINITION 2 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 10 is a maximum matching. There is an obvious connection between maximum matchings and augmenting paths. For example,

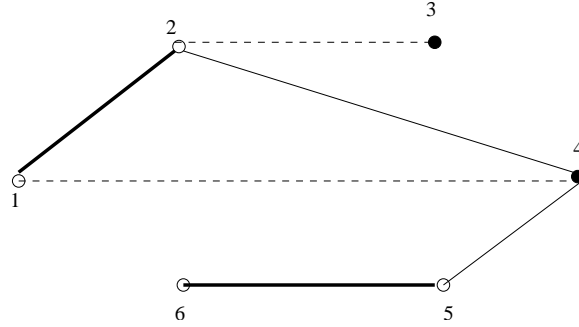


Figure 12: An M -augmenting path for a matching

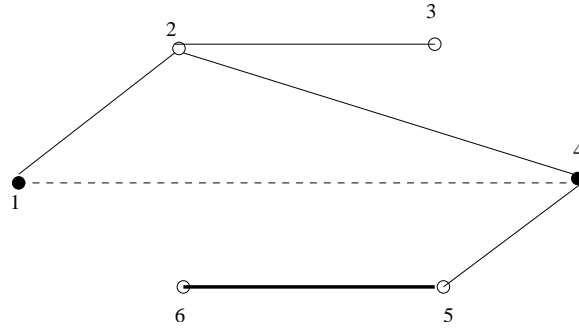


Figure 13: An M -augmenting path for a matching

notice the maximum matching M in Figure 10. We cannot seem to find an M -augmenting path for this matching. On the other hand, observe that the matching in Figure 11 is not a maximum matching (the matching in Figure 10 has more edges), and Figure 12 shows an augmenting path of this matching. This observation is formalized in the theorem below.

THEOREM 4 *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 4 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.

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 4, 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 14 (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 14.

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 .

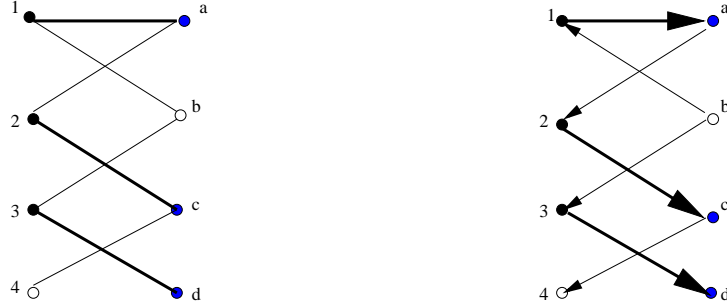


Figure 14: A bipartite graph with a matching

LEMMA 4 *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 14 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 15). 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 15). 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).

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.

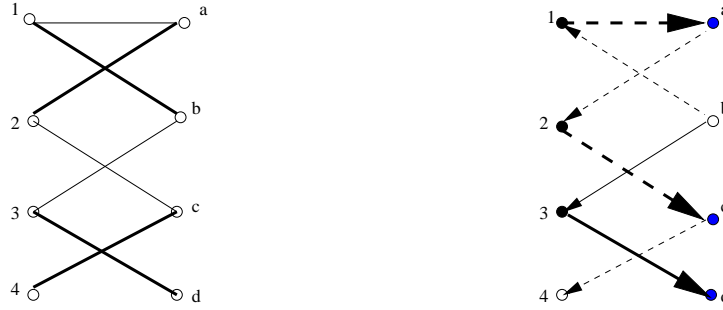


Figure 15: A bipartite graph with a matching

DEFINITION 3 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 16. Figure 16 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.

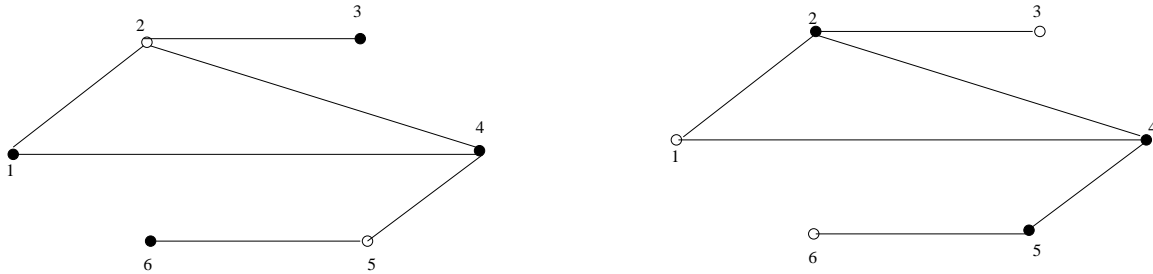


Figure 16: Vertex cover

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 $\nu(G)$ - this is also called the **matching number** of G . Denote the size of minimum cover in G as $\tau(G)$ - this is also called the **vertex cover number** of G .

LEMMA 5 For any graph G , $\nu(G) \leq \tau(G)$.

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 implies that for every graph G , $\nu(G) \leq \tau(G)$. ■

Lemma 5 can hold with strict inequality in general graphs. Consider the graph in Figure 17. A minimum vertex cover, as shown with black vertices, has two vertices. A maximum matching, as shown with the dashed edge, has one edge.

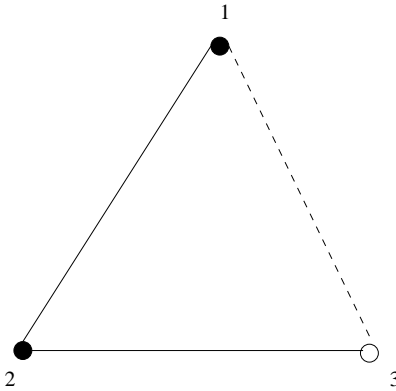


Figure 17: Matching number and vertex cover number

But the relationship in Lemma 5 is equality in case of bipartite graphs as the following theorem, due to Koïig shows.

THEOREM 5 (Koïig’s Theorem) *Suppose $G = (N, E)$ is a bipartite graph. Then, $\nu(G) = \tau(G)$.*

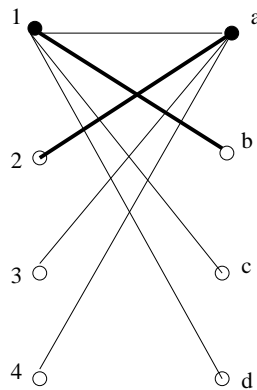


Figure 18: Matching number and vertex cover number in bipartite graphs

Figure 18 shows a bipartite graph and its maximum matching edges (in dark) and minimum vertex cover (in dark). For the bipartite graph in Figure 18 the matching number (and the vertex cover number) is two.

We will require the following useful result for proving Theorem 5. 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 6 *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 $\{i, j\} \in E$. 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 4. 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 4 since M' is a maximum matching. ■

PROOF OF THEOREM 5

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 6, 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, $\nu(G') = \tau(G')$.

We show that $\nu(G') = \nu(G) - 1$. By deleting the edge covering i in any maximum matching of G , we get a matching of G' . Hence, $\nu(G') \geq \nu(G) - 1$. Suppose $\nu(G') > \nu(G) - 1$. This means, $\nu(G') \geq \nu(G)$. Hence, there exists a maximum matching of G' , which is also a matching of G , and has as many edges as the maximum matching 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 $\nu(G') = \tau(G') = \nu(G) - 1$. 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 $\tau(G') + 1$ vertices. Hence, the minimum vertex cover of G must have no more than $\tau(G') + 1 = \nu(G)$ vertices. This means $\tau(G) \leq \nu(G)$. But we know from Lemma 5 that $\tau(G) \geq \nu(G)$. Hence, $\tau(G) = \nu(G)$. ■

Now, we show how to compute a minimum vertex cover from a maximum matching in a bipartite graph. We leave the formal proof of the fact that this indeed produces a minimum vertex cover as an exercise.

Consider the bipartite graph G in Figure 18. Figure 19 shows its maximum matching (in dark edges) M and the directed graph G^M suggested by Lemma 4. We first consider the set of vertices *reachable* from the set of vertices in L^M in G^M - a vertex i is reachable from a vertex j in a directed graph if there is a path from j to i . Call such a set of vertices R^M . In Figure 19, we have $L^M = \{c, d\}$ and $R^M = \{c, d, 1, b\}$.

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

We show that C 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 $j \in C$, 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 $i \in R^M$, and hence $i \in C$.

Note that $R^M \cap B^M = \emptyset$ - because if some vertex $i \in R^M$ 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.

Note that $L^M \subseteq R^M$. Hence, C is disjoint from L^M . Since C is disjoint from B^M , it is disjoint from $B^M \cup L^M$.

Further, no edge in M is contained in C . To see this, suppose $\{i, j\} \in M$ and $i, j \in C$ 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 $j \in C$.

The last two observations imply that $\#C \leq \#M$. Therefore, C is a minimum vertex cover. In Figure 19, we have $C := \{1, a\}$, which is a minimum vertex cover.

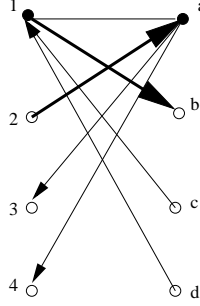


Figure 19: Minimum vertex cover from maximum matching

7 BASIC DIRECTED GRAPH DEFINITIONS

A **directed graph** is defined by a triple $G = (N, E, w)$, where $N = \{1, \dots, n\}$ 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.

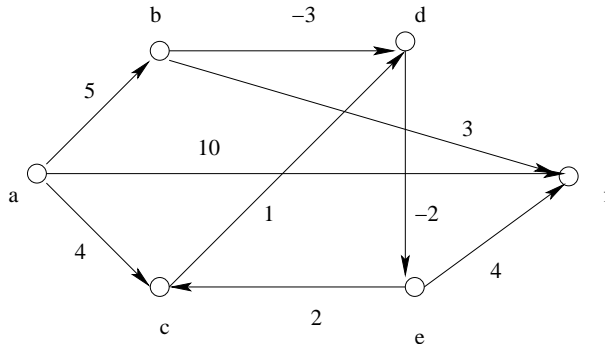


Figure 20: A Directed Graph

A **path** is a sequence of *distinct* nodes (i^1, \dots, i^k) such that $(i^j, i^{j+1}) \in E$ for all $1 \leq j \leq k-1$. If (i^1, \dots, 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 $(i^1, \dots, i^k, i^{k+1})$ such that (i^1, \dots, 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 $P = (i^1, \dots, i^k, i^{k+1})$ is the sum of the edge lengths in the path or cycle, and is denoted as $l(P) = w(i^1, i^2) + \dots + w(i^k, i^{k+1})$. Suppose there is at least one path from node i to node j . Then, the **shortest path** from node i 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 $s(i, j)$. We denote $s(i, i) = 0$ for all $i \in N$.

Figure 20 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 7 *Suppose (i, i^1, \dots, 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 $i^p \in \{i^1, \dots, i^k\}$, $s(i, i^p) + s(i^p, j) = s(i, j)$.*

Proof: Let $P_1 = (i, i^1, \dots, i^p)$ and $P_2 = (i^p, \dots, i^k, j)$. We show that $l(P_1) = s(i, i^p)$ and $l(P_2) = s(i^p, j)$, and this will establish the claim.

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 $l(P_3) + l(P_2) < s(i, j)$. But this contradicts the fact that (i, i^1, \dots, 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 $s(i^p, j) < l(P_2)$. ■

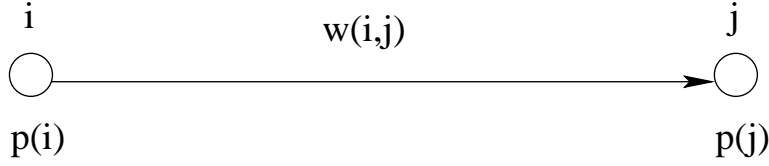
7.1 POTENTIALS

DEFINITION 4 A **potential** of a directed graph G is a function $p : N \rightarrow \mathbb{R}$ such that $p(j) - p(i) \leq w(i, j)$ for all $(i, j) \in E$.

Figure 21 illustrates the 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. The following theorem provides a necessary and sufficient condition.

THEOREM 6 *There exists a potential of directed graph $G = (N, E, w)$ if and only if G has no cycles of negative length.*



$$p(i) + w(i,j) \geq p(j)$$

Figure 21: Idea of potentials

Proof: Suppose a potential p exists of graph G . Consider a cycle (i^1, \dots, 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

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

Adding these inequalities, we get $w(i^1, i^2) + \dots + w(i^{k-1}, i^k) + w(i^k, i^1) \geq 0$. The right hand side of the inequality is the length of the cycle (i^1, \dots, 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 22 shows a directed graph and how the graph with the dummy vertex is created.

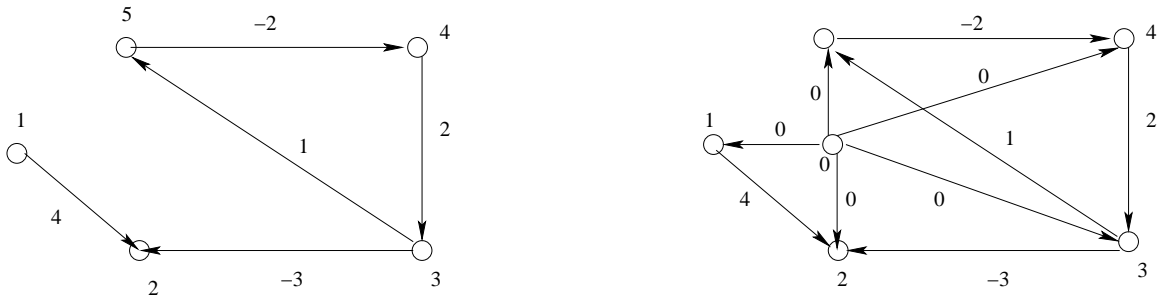


Figure 22: A directed graph and the new graph with the dummy vertex

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 7). 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 6 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*

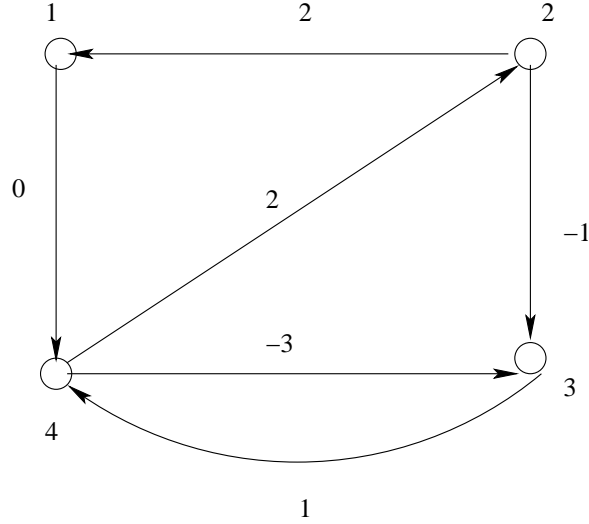


Figure 23: A graphical representation of difference inequalities

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 vertex set $\{1, 2, 3, 4\}$ and an edge for every inequality with weights given by the right hand side of the inequalities. Figure 23 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 6.

From the proof of Theorem 6, 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 7 *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 6 - the only difference being we do not need to construct the new graph G' and work on graph G directly. ■

Figure 24 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 7. 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.

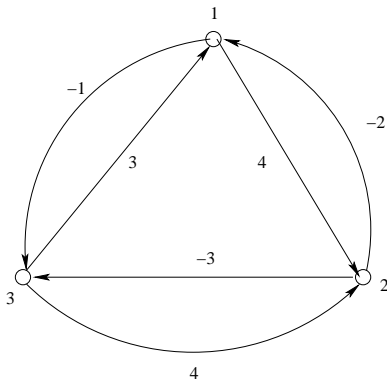


Figure 24: Potentials for a complete directed graph

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 8 *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 25 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.

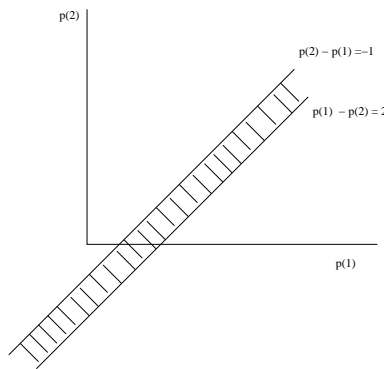


Figure 25: Potentials of a complete graph with two vertices

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 5 *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 8 *A strongly connected digraph $G = (N, E, w)$ 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. 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 $j, k \in N$ (note that such a digraph will always have cycles of non-negative length, and hence will have potentials). Consider the shortest path from j to k . Let it be (j, j^1, \dots, j^q, k) . Consider any potential p of G . We can write

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

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 $p(k) - p(j) = q(k) - q(j) = s(j, k)$. ■

It is instructive to look at the right digraph in Figure 27. One can verify that $p(0) = 0, p(1) = 0, p(2) = -0.5$ is a potential of this digraph. Another potential can be obtained by taking shortest paths from vertex 1: $q(0) = 0.5, q(1) = 0, q(2) = -0.5$. Notice that p and q do not differ by constant. So this digraph fails potential equivalence. This can also be observed by noting that $s(0, 1) = 0$ and $s(1, 0) = 0.5$ - so by Theorem 8 potential equivalence must fail in this digraph.

9 APPLICATION: FAIR PRICING

Consider a market with $N = \{1, \dots, n\}$ agents (candidates) and $M = \{1, \dots, m\}$ indivisible goods (jobs). We assume $m = n$ (this is only for simplicity). Each agent is interested 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 $\mu : N \rightarrow M$, 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 6 A matching μ can be **fairly priced** if there exists p such that for every $i \in N$,

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

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

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

Substituting the values, we get

$$\begin{aligned} p(1) - p(2) &\leq -1 \\ p(2) - p(1) &\leq -1, \end{aligned}$$

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

DEFINITION 7 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. The efficient matching in this example is: agent i gets good i for all $i \in \{1, 2, 3\}$.

THEOREM 9 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

	1	2	3
$v_1.$	4	2	5
$v_2.$	2	5	3
$v_3.$	1	4	3

Table 1: Fair pricing example

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, we take μ to be the efficient matching, and construct G^μ . The edge lengths are:

$$w(1, 2) = v_{22} - v_{21} = 3$$

$$w(2, 1) = v_{11} - v_{12} = 2$$

$$w(1, 3) = v_{33} - v_{31} = 2$$

$$w(3, 1) = v_{11} - v_{13} = -1$$

$$w(2, 3) = v_{33} - v_{32} = -1$$

$$w(3, 2) = v_{22} - v_{23} = 2.$$

The digraph G^μ is shown in Figure 26.

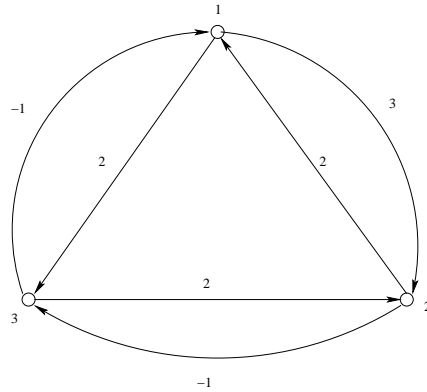


Figure 26: Digraph G^μ

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

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

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 $C = (i^1, i^2, \dots, i^k, i^1)$ in G^μ . Denote by $R = N \setminus \{i^1, \dots, i^k\}$. The length of the cycle C in G^μ is

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

Note that $\sum_{r=1}^k v_{i^r i^r} + \sum_{h \in R} v_{hh}$ 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 $h \in \{2, 3, \dots, k\}$ and $\mu'(i^1) = i^k$. Note that μ' is the matching implicitly implied by the cycle, and $V(\mu') = \sum_{h \in R} v_{hh} + [v_{i^1 i^k} + v_{i^2 i^1} + v_{i^3 i^2} + \dots + v_{i^k i^{k-1}}]$. 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 $C = \{i^1, \dots, i^k\}$, and suppose that $\mu'(i^r) = i^{r-1}$ for all $r \in \{2, \dots, k\}$ and $\mu'(i^1) = i^k$. Then, the length of the cycle (i^1, \dots, 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. 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 $p(3) = s(1, 3) = 2$.

10 APPLICATION: ONE AGENT IMPLEMENTATION

Consider an agent coming to an interview. The agent has a private information, his ability, which is called his *type*. Let us assume that type of the agent is a real number in a finite set $T = \{t_1, t_2, \dots, t_n\}$, where $t_i \in \mathbb{R}$ for all $i \in \{1, \dots, n\}$ and $t_1 < t_2 < \dots < t_n$. The interviewer does not know the type of the agent. The agent reports some type and given this type, the interviewer assigns a number in $[0, 1]$, called the allocation. The allocation reflects the probability of getting the job or the fraction of the responsibility assigned to the agent. The interviewer knows the set T but does not know what the actual type of the agent. This

leaves the room open for the interviewer to report any type to increase his utility. So, an allocation rule is a mapping $a : T \rightarrow [0, 1]$.

Given an allocation and his ability, the agent derives some value or cost from the job. This is captured by the valuation (cost) function $v : [0, 1] \times T \rightarrow \mathbb{R}$.

Along with the allocation, there is a payment done to the agent. This is again a function of the reported type. So, a payment rule is a mapping $p : T \rightarrow \mathbb{R}$. We assume that if an agent reports type s to an allocation rule a and payment rule p when his true type is t , his net utility is

$$v(a(s), t) + p(s).$$

DEFINITION 8 *An allocation rule $a : T \rightarrow [0, 1]$ is **implementable** if there exists a payment rule $p : T \rightarrow \mathbb{R}$ such that reporting true type is a dominant strategy, i.e.,*

$$v(a(t), t) + p(t) \geq v(a(s), t) + p(s) \quad \forall s, t \in T.$$

In this case, we say payment rule p implements allocation rule a .

Define for every $s, t \in T$ and for allocation rule a

$$w(s, t) = v(a(s), s) - v(a(t), s).$$

The constraints of implementability admit a potential form. We can say an allocation rule a is implementable if there exists a payment rule p such that

$$p(s) - p(t) \leq w(t, s).$$

Consider the complete directed graph G^a corresponding to allocation rule a with set of vertices being T and length of edge (s, t) being $w(s, t)$. Then, the payment rule defines the potential of G^a . Such a payment rule exists if and only if G^a has no cycles of negative length. Indeed, we can find a particular payment rule, if it exists, by fixing a particular type, say the lowest type t_1 , and taking the shortest path from t_1 to every other type in G^a .

We give two examples of allocation rules for $T = \{0, 1, 2\}$ and $v(a(s), t) = a(s) \times t$ for all $s, t \in T$.

EXAMPLE 1: Suppose $a(0) = 1, a(1) = 0, a(2) = 1$. Then, the edge lengths of G^a is as shown below.

$$\begin{aligned} w(0, 1) &= v(a(0), 0) - v(a(1), 0) = 0 \\ w(0, 2) &= v(a(0), 0) - v(a(2), 0) = 0 \\ w(1, 0) &= v(a(1), 1) - v(a(0), 1) = -1 \\ w(1, 2) &= v(a(1), 1) - v(a(2), 1) = -1 \\ w(2, 0) &= v(a(2), 2) - v(a(0), 2) = 0 \\ w(2, 1) &= v(a(2), 2) - v(a(1), 2) = 2. \end{aligned}$$

The digraph G^a is shown in figure 27. It is easily seen that $(0, 1, 0)$ is a negative length cycle of this digraph. So, a is not implementable in this example.

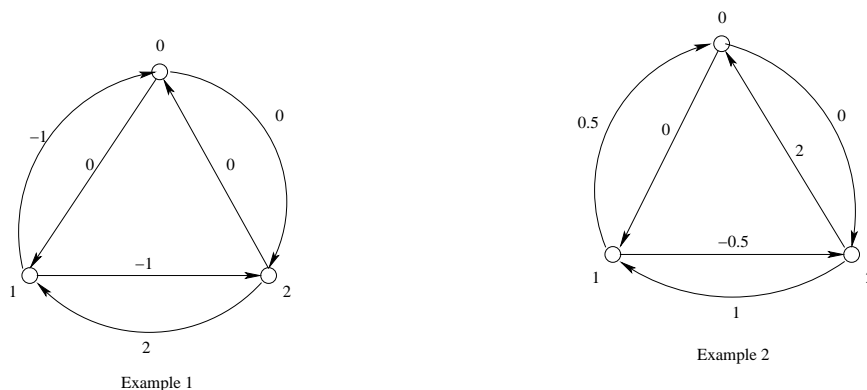


Figure 27: G^a for Example 1 and Example 2

EXAMPLE 2: Suppose $a(0) = 0, a(1) = 0.5, a(2) = 1$. Then the edge lengths of G^a is as shown below.

$$\begin{aligned}
 w(0, 1) &= v(a(0), 0) - v(a(1), 0) = 0 \\
 w(0, 2) &= v(a(0), 0) - v(a(2), 0) = 0 \\
 w(1, 0) &= v(a(1), 1) - v(a(0), 1) = 0.5 \\
 w(1, 2) &= v(a(1), 1) - v(a(2), 1) = -0.5 \\
 w(2, 0) &= v(a(2), 2) - v(a(0), 2) = 2 \\
 w(2, 1) &= v(a(2), 2) - v(a(1), 2) = 1.
 \end{aligned}$$

The digraph G^a is shown in figure 27. It is easily seen that there is no negative length cycle of this digraph. So, a is implementable in this example. One particular payment rule which implements a in this example is to set $p(0) = 0$ and consider the shortest paths from 0, which gives $p(1) = 0, p(2) = -0.5$.

11 THE MAXIMUM FLOW PROBLEM

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:

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

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.

DEFINITION 9 *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 28 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 rail-ways. 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 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.

11.1 ANALYSIS OF THE MAXIMUM FLOW PROBLEM

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

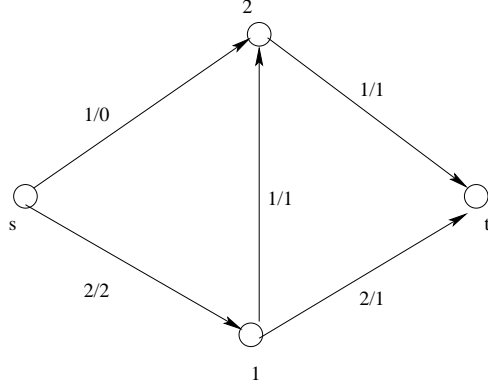


Figure 28: Feasible flow

LEMMA 9 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 10 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 29 shows a saturated cut: $(\{s, 2\}, \{1, t\})$.

The second part of the definition of saturated cut is equally important. Figure 30 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 $f(1, 2) \neq 0$.

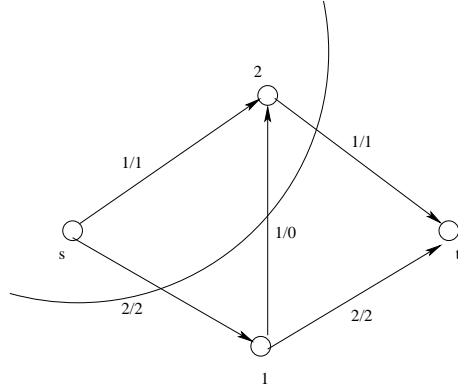


Figure 29: Saturated cut

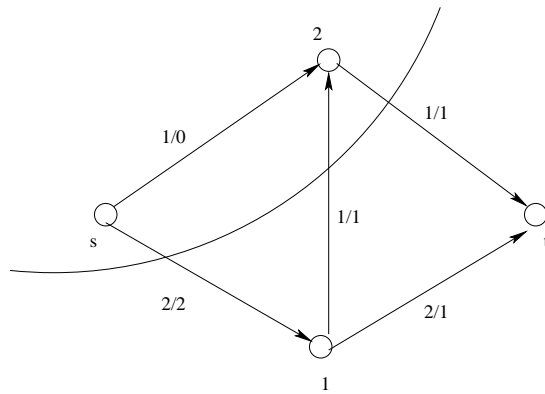


Figure 30: A cut which is not saturated

LEMMA 10 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 , $\nu(f) = \kappa(S)$.
3. if $(S, N \setminus S)$ is saturated for flow f , then f is a maximum flow.

Proof: We prove (1) first. Using Lemma 9, 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 10 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 31 shows a maximum flow and an (s, t) -cut which is not saturated.

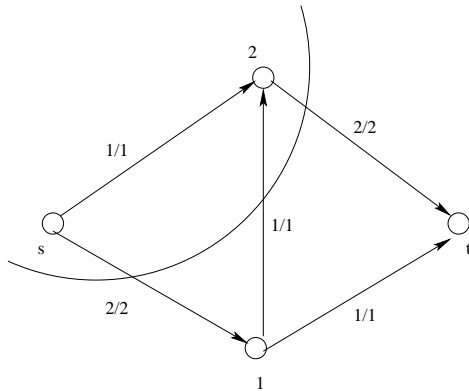


Figure 31: A maximum flow f does not imply every (s, t) -cut is saturated for f

11.2 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 32 and 33.

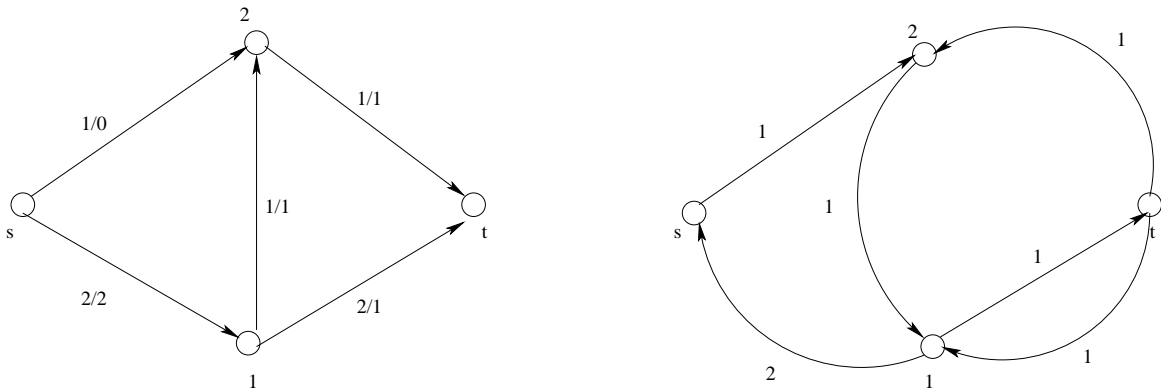


Figure 32: Residual digraph of a flow

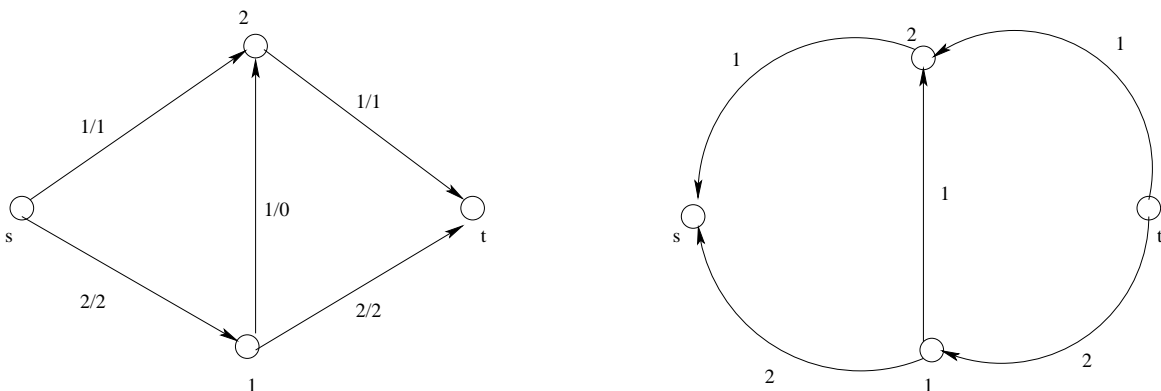


Figure 33: Residual digraph of a flow

The next theorem illustrates the importance of a residual digraph.

THEOREM 10 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 32, 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 10, 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 11 *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 10). By Theorem 10, 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 10). This implies that $\nu(f) = \min_{S \in S^G} \kappa(S)$. ■

11.3 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 10 (feasibility is maintained). Iterate from Step S1.

If the algorithm terminates, then from Theorem 10 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 34. You can verify that the maximum flow amount is 7.

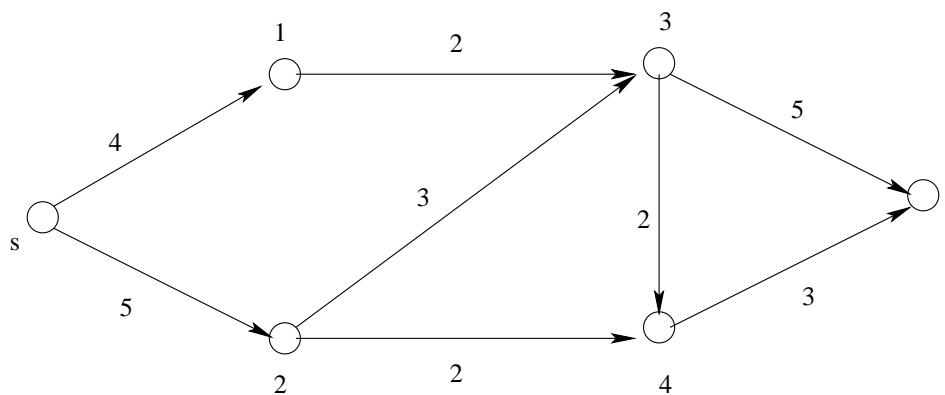


Figure 34: Maximum flow

We show some of the steps. Let us start from a feasible flow as shown in Figure 35. Then, Figure 36 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 37. 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.

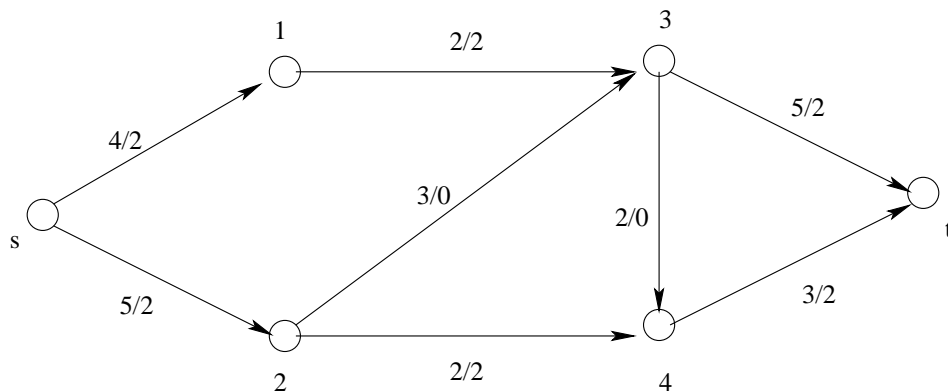


Figure 35: A feasible flow

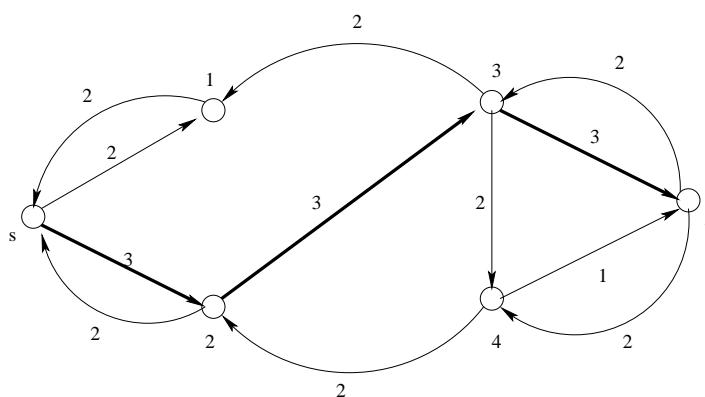


Figure 36: Residual digraph for flow in Figure 35

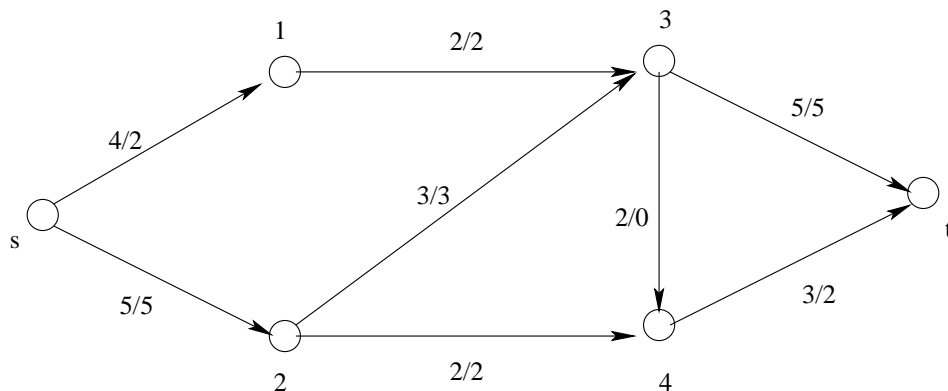


Figure 37: Maximum flow for the flow graph in Figure 34

THEOREM 12 *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. ■

However, the algorithm may not terminate in general for irrational capacities. You are encouraged to think of an example with irrational capacities.

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.

The premise of this problem is a directed graph $G = (N, E)$ (not weighted) and two special vertices s (source - with no incoming edges) and t (terminal - with no outgoing edges). 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 38.

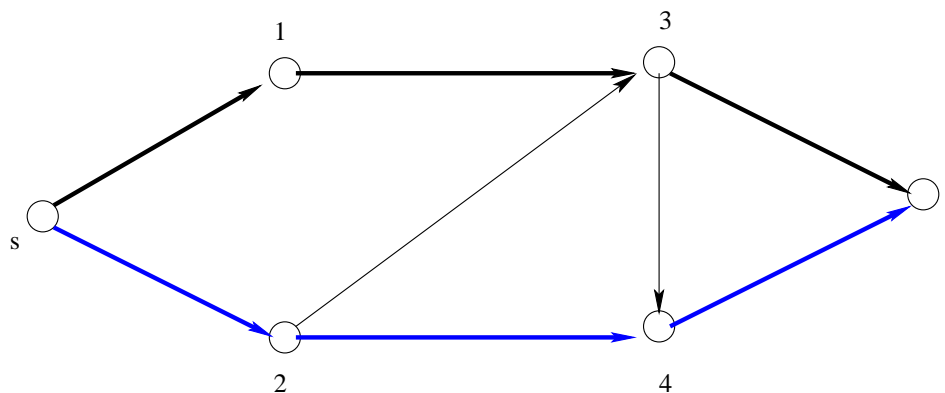


Figure 38: Two disjoint paths

Consider a 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 ensuing corollary, show that the two problems are related.

THEOREM 13 (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 $(i, j) \in E$. 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 . Now, note that any feasible integral flow in G is either (a) a set of paths from s to t or (b) a cycle (not involving s or t). Also, an integral maximum flow exists. Consider such an integral maximum flow. If a maximum flow involves a cycle, we can set the flow in this cycle to zero without loss of generality, and still get an integral maximum flow. Hence, without loss of generality, a maximum flow consists of a set of paths from s to t . Since the capacity of every edge is just 1, 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 . ■

An immediate corollary to this result is the following.

COROLLARY 1 *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 13, $k \geq \ell$. Suppose $k > \ell$. Then, again by Theorem 13, 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: $(s, i^1), (s, i^2), \dots, (s, i^k)$. 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 39 illustrates that. There are two disjoint paths from s to t in the graph in Figure 39. Take for example the pair of disjoint paths $(s, 1, 2, t)$ and $(s, 3, 4, t)$ (there are other pairs of disjoint paths). If we remove two edges $(s, 1)$ and $(s, 3)$, there are still paths from s to t .

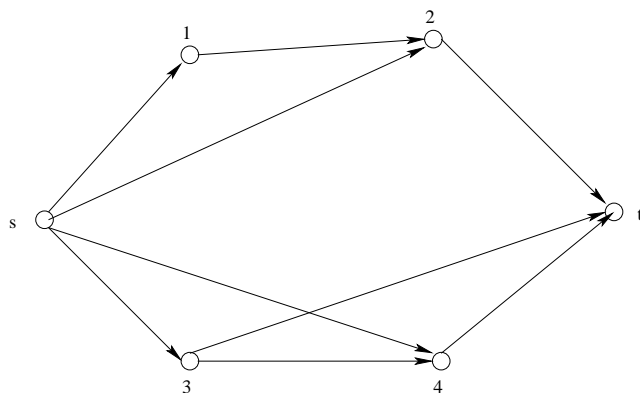


Figure 39: Two disjoint paths