A Review of Flow-Capacitated Networks: Algorithms, Techniques, and Applications

Abstract

This paper presents a review of flow network concepts, including definition of some graph-theoretic basics and a discussion of network flow properties. It also provides an overview of some crucial algorithms used to solve the maximum-flow problem such as the Ford and Fulkerson algorithm (FFA), supplemented with alternative solutions, together with the essential terminology for this algorithm. Moreover, this paper explains the max-flow min-cut theorem in detail, analyzes the concepts behind it, and provides some examples and their solutions to demonstrate this theorem. As a bonus, it expounds the reduction and transformation techniques used in a capacitated network. In addition, this paper reviews one of the popular techniques for analyzing capacitated networks, which is the "decomposition technique". This technique is centered on conditioning a complicated network on the possible states of a keystone element K_1 or on the possible combinations of states of many keystone elements. Some applications of capacitated network problems are addressed based on each type of problem being discussed.

Keywords: Capacitated networks; Max-flow min-cut theorem; Ford and Fulkerson algorithm; Reduction and transformation techniques; Decomposition technique; Flow network applications.

1. Introduction

A capacitated network (also called a flow network) in graph theory is a directed graph where each directed edge has a specified capacity, which is the maximum value of a specific entity, commodity, or 'through' quantity that can flow through the edge. For example, we can state that the capacity of a wire is 30 amperes if the maximum electric current that can flow through the wire (that the wire can withstand) is 30 amperes. Moreover, each edge in the capacitated network supports a flow that moves through the edges without breaching capacity constraints from the source node to the sink node. Notably, a directed graph in Operations Research (OR) is called a network, which has nodes named vertices. The arcs connecting these vertices are called edges or branches. Understandably, a flow should adhere to the constraint that the inflow should be equal to the outflow at any vertex. The source and sink nodes, which have only outgoing flow and incoming flow, respectively, are not restricted by this constraint. This constraint, called "the law of flow conservation", is similar to Kirchhoff's Current Law when an electric current is the subject of the flow. Capacitated networks have many applications when modelling real-world problems. For instance, the commodity flowing in a capacitated network might be the traffic in a computer network, the liquid flowing through pipes, the current in an electric circuit, the information passing through communication networks, or anything similar that moves through a network of vertices.

The maximum-flow problem seeks a solution, where we can compute the highest flow from a source to a sink within the maximum capacity limits of the network branches. Notably, an effective algorithm like that of Ford-Fulkerson can solve this problem easily since it is proven correct and tested for various capacitated networks. Moreover, we can adapt certain techniques that apply particular maximum-flow algorithms (such as those based on the reduction and transformation rules or the decomposition technique) so as to cope (at least partially) with the increase in complexity for larger networks.

The rest of this paper is arranged as follows. A review of flow network concepts is presented in Section 2, which details network properties, and the problem definition, while viewing the maximum flow aspects. Section 2 also provides an overview of some crucial algorithms used to solve the maximum-flow problem, in addition to some useful notation. Section 3 presents the Ford and Fulkerson method or the Ford and Fulkerson algorithm (FFA), supplemented with alternative

solutions, together with the essential terminology for this algorithm. Additional examples to explain the algorithm are also featured in this section. Section 4 explains the max-flow min-cut theorem in detail, analyzes the concepts behind it, and provides some examples and their solutions to demonstrate this theorem. Section 5 explains the reduction and transformation techniques used in a capacitated network. Moreover, Section 6 extends this work by presenting one of the popular techniques for analyzing capacitated networks, which is the "decomposition technique". This technique is centered on conditioning a complicated network on the possible states of a keystone element K_1 or on the possible combinations of states of many keystone elements. Subsequently, Section 7 highlights some applications of the flow network problem. Finally, Section 8 concludes the paper.

2. Flow-Capacitated Networks:

The section introduces flow networks, including definition of some graph-theoretic basics, a discussion of network flow properties, and a detailed definition of the maximum-flow problem. Besides, some useful notation is clarified in this section.

2.1. Definition of Flow Networks and Flows

A *flow network* [1, 2, 3, 63] $G = (V, E)$, is a directed graph, where V is a set of vertices and E is a set of directed edges, where each edge (u, v) is characterized by the ordered set of the two nodes u and v it connects. Without loss of generality we assume that (u, v) denotes at most one edge extending between the two nodes u and v, because if there are several such edges then they can be combined into one. Each edge $(u, v) \in E$ has a nonnegative *function* $c(v, u) \geq 0$, called the capacity function. If $(u, v) \notin E$, then we can add (u, v) to E provided we set $c(u, v) = 0$. Self-loops (edges connecting a vertex to itself) are disallowed in the graph, which means that the graph is a simple one. If we isolate two specific vertices, a **source** s and a **sink** t, then the four-tuple (G, c, s, t) is called a flow network [1, 2, 3]. Now presume that each node falls on some path when the network is traversed from the source node to the sink node. This means that the flow network involves at least a path $s \rightarrow v \rightarrow t$ for each node $v \in V$. Accordingly, the network is assumed to be connected and because each node other than the source has at least one edge incident on it $|E| \geq |V| - 1$; as can be seen in the examples of the flow networks in Figures 2 and 3.

Flows

There are many ideas for a flow function that might be used to describe the behavior of a flow network [2, 3, 5]. In fact, the net flow between any two vertices is conveniently modeled by such a flow function. One of the important examples of a useful flow function is derived via what is called a pseudo-flow algorithm, which is used to resolve the maximum flow and minimum cut problems [2, 3, 5].

A pseudo-flow in $\boldsymbol{G}$ [59-61] satisfies the properties stated below for all vertices u and v with the real-valued function $f: V \times V \rightarrow \mathbb{R}$:

- **Capacity constraint**: For all $u, v \in V$, we require $0 \leq f(u, v) \leq c(u, v)$. This constraint indicates that flow through a specific branch (extending between nodes u and v) should not exceed the flow limits for this branch and must be non-negative.
- **Skew symmetry**: This is the only requirement that involves encoding the total units of flow between each pair of vertices u and$v$. This is due to the fact that $f(u, v) = -f(v, u)$, i.e., the net flow from node u to node v is the opposite of the total flow from node v to node u.

It is necessary to verify the net flow into a specific vertex u for a particular pseudo-flow f in a flow graph. An additional function $x_f : V \rightarrow \mathbb{R}$ is needed to represent the net flow entering node u, and hence it is stated by $x_f(u) = \sum_{v \in V} f(v, u)$. Accordingly, the vertex u is considered active if $x_f(u) > 0$, deficient if $x_f(u) < 0$ or conservative if $x_f(u) = 0$.

The aforementioned explanations naturally bring us to mention two crucial definitions related to that of a pseudo-flow:

A pre-flow is a pseudo-flow in which all $v \in V \backslash \{s\}$, satisfies the extra property:

- **Non-deficient flow**: The total inflow in any vertex v other than the source is non-negative. The exception of the source is necessary, since, by definition, it is a node that "produces" flow. Mathematically speaking, a non-deficient flow means: $x_f\ (v)\ \geq\ 0$ for all $v\ \in\ V\setminus\{s\}$.

For all $v\ \in V\setminus\{s,\ t\}$, **a feasible flow**, or only a **flow** is the pseudo-flow that fulfills the extra property.

- **Flow conservation [**2, 3, 63**]:** For all $u \in V - \{s, t\}$, we require

$$\sum_{v\in V} f(v,u) = \sum_{v\in V} f(u,v)\ .$$

The conservation property states that, for a node other than the source or the sink, the total outflow from a vertex should be equivalent to the total inflow into that vertex. Roughly speaking, "what flows in must come out" for a node other than the source node or the sink node. Whenever $(u, v) \notin E$, nothing flows from u to v, and $f(u, v) = 0$.
The flow from vertex u to v is the non-negative quantity $f(u, v)$, where the ***value*** $|f|$ of a flow f is defined as

$$|f| = \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s),$$

Notably, this is the net outflow from the source node minus the inflow towards it, where the $|.|$ notation indicates flow value and not cardinality or absolute value. Usually, a flow network cannot have any edges entering the source, and the flow entering the source, defined by $\sum_{v\in V} f(v,s) = 0$. Nevertheless, it is essential to introduce the inflow concept at the source at this stage (though we set it identically to zero later) since this conforms to our brief introduction to residual networks, which comes later in the paper.

2.2. Intuitive Analysis of Flow Networks

The method of transferring the flow units between vertices is a critical issue in the analysis of a flow network. Notably, distinguishing various edges between two vertices is not warranted here. We recall that we assumed, without loss of generality, that $(u,\ v)$ denotes at most one edge extending between the two nodes u and v. We now explain this further.

- For any pair of vertices u and v, let us tentatively assume that we have two edges in parallel from node u to node v whose branch capacities are 4 and 6 units, respectively. Instead of using these two edges, we can replace them with an equivalent single edge between u and v with a capacity of 10 units. It is not necessary to know how these two edges might share the actual flow that can be transferred, but the important thing is to know that we can, as a maximum, transfer 10 units from u to v.
- Again, given a pair of different vertices u and v, assume we have a flow of 7 units from the direction u to v, and an additional opposite flow of 2 units from the direction v to u. These correspond to 5 units of net flow from u to v, which equates to a negative 5-unit flow in the opposite direction v to u. Therefore, the sign is used to indicate the actual direction measured with respect to the u to v direction as a reference direction.

Thus, the capacity function$c$: $V \times V \rightarrow \mathbb{R}_{\infty}$ (which avoids having several edges originating and terminating at the same set of two vertices) is adequate for successful flow analysis. Correspondingly, imposing the skew symmetry property on flow functions is justified since there is a guarantee that the flow between two nodes is uniquely coded by a distinct non-negative numerical value (to show magnitude), together with a sign (for direction designation w.r.t. a reference direction). Once you know the flow between u and v, then through skew symmetry you have known the flow between v and u. Usually, the immediate impact of network solutions are not intuitive, but they become more realistic during the actual analysis of the flow network. The capacity constraint ensures that a flow on any edge in the flow graph will not exceed the capacity of that edge.

2.3. Networks with Multiple Sources and Sinks

Many sources and many sinks can exist in any maximum flow problem, which rules out the necessity of existence of only one source and one sink [2]. For example, the telecommunication wireless network may actually have multiple base stations $\{s_1, s_2, \dots, s_m\}$, each servicing particular locations in a certain best way and numerous users $\{t_1, t_2, \dots, t_n\}$ with varying expectations of the (network traffic) communication rates, as shown in Figure 1(a).

It is possible to replace multiple flows involving multiple sources and sinks by a single basic maximum flow problem. Figure 1 shows the conversion of multiple sinks and sources in part (a) to an ordinary single sink and an ordinary single source, respectively, in part (b). To achieve this purpose, we added a **super-source** s to the flow-graph. We also added a directed arc (s, s_i) whose branch capacity is $c(s, s_i) \to \infty$ that connects the super-source s with each of the initial multiple sources s_i for each $i = 1, 2, \dots, m$. Moreover, we added a new **super-sink** t too to the flow graph, together with a directed arc $(\mathrm{t_i}, \mathrm{t})$ with capacity $\mathrm{c}(\mathrm{t_i}, \mathrm{t}) \to \infty$ that connects each of the initial multiple sinks t_i for each $\mathrm{i} = 1, 2, \dots, \mathrm{n}$ to the super-sink t. Actually, there are noticeable similarities between Fig. 1(a) and Fig. 1(b). The only source s releases the sufficient flow required to satisfy the multiple sources s_i , while the one sink t consumes the flow delivered to the several sinks t_i. We observe that the infinite value for the capacities $c(s, s_i)$ and $\mathrm{c}(\mathrm{t_i}, \mathrm{t})$ might not be necessary as it suffices to secure only the following finite values $c(s, s_i) = \sum_{(s_i, u) \in E} c(s_i, u)$, and $\mathrm{c}(\mathrm{t_i}, \mathrm{t}) = \sum_{(v, \mathrm{t_i}) \in E} c(v, \mathrm{t_i})$.

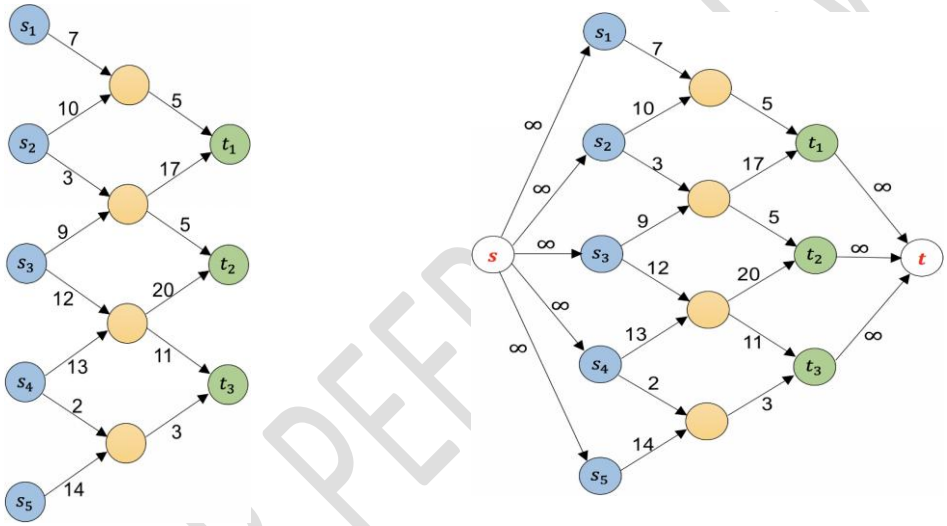


Figure 1. Conversion of a multiple-source, multiple-sink maximum-flow problem for the telecommunication wireless network into a problem with a single source and a single sink. The figure (a) above shows how to transform a multiple-source, multiple-sink maximum-solution problem for the telecommunication wireless network into a single-source and a single-sink problem. It has a five-source $S = \{s_1, s_2, s_3, s_4, s_5\}$ flow network with three sinks $T = \{t_1, t_2, t_3\}$. Diagram (b) shows the corresponding network flow for a single-source and a single-sink. The significant changes made in going from (a) to (b) include the addition of a super-source s and an edge with an unlimited (or sufficient) capacity from s to each of the multiple sources, the addition of a super-sink t and an edge with an unlimited (or sufficient) capacity from each of the multiple sinks to t.

2.2.4. Example

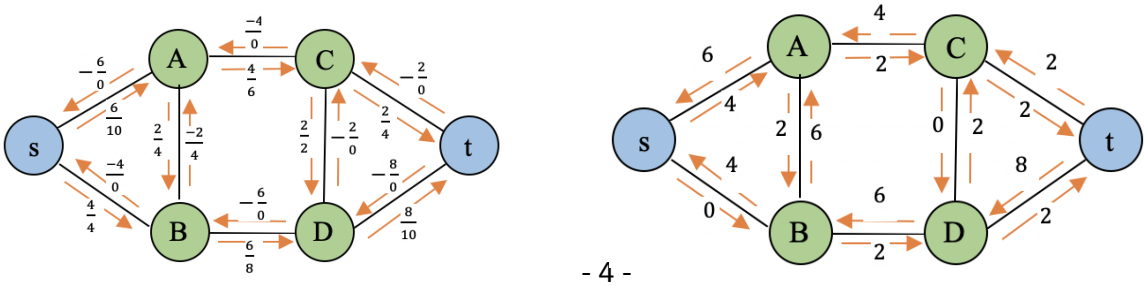


Figure 2. Illustration of flow and capacity in a flow network

Figure 3. The flow network in Fig. 2 redrawn to show the residual capacities.

Figure 2 shows the source depicted as s, the sink identified as t , and four additional vertices for a flow network. The flow and capacity are indicated via an ordinary-fraction $\frac{f}{c}$ notation, where f and c denote the flow and capacity, respectively. The way this network conserves flow, upholds both capacity and skew symmetry is evident in Figure 2. For instance, the total amount of flow from s to t is 10, which can be easily seen from the fact that the total outgoing flow from s is 10, which is also the incoming flow to t. Notably, every flow is accounted for in all the vertices, since the flowing commodity is neither generated nor consumed at any of these vertices.

Figure 3 shows the residual network for the given flow network. Notice how a positive residual capacity $(c - f)$ appears on some branches where the original capacity is zero, for instance, for the branch (D, C). This flow is not a maximum flow. There is available capacity along the paths $(s, A, C, t), (s, A, B, D, t)\ and\ (s, A, B, D, C, t)$, which are labelled as augmenting paths. The residual capacity of the first path is

$\min\big(c(s,A) - f(s,A),\ c(A,C) - f(A,C),\ c(C,t) - f(C,t)\big) = \min(10-6, 6-4, 4-2) =$
$\min(4, 2, 2) = 2.$

Notice that as long as there exists some path with a positive residual capacity, the flow will not be maximum. The residual capacity for some path is the minimum residual capacity of all edges in that path.

Finding the maximum flow [64] is the simplest and most popular problem using flow networks. The maximum flow provides the largest possible total flow from the source to the sink in a given network. Understandably, the maximum flow algorithm can solve many other problems. This is particularly true if you model those problems so as to be pertaining to flow networks, such as the bipartite matching problem, the assignment problem and the transportation problem. We can effectively solve such problems by relabeling them as max-flow ones. The max-flow min-cut theorem states that finding a maximal network flow is equivalent to finding a cut of minimum capacity that separates the source and the sink, where a cut is the division of nodes such that the source is in one division and the sink is in another.

Some of the well-known Maximum Flow Algorithms include the following:

- **The Ford-Fulkerson algorithm,** which was published in 1956 by L.R. Ford Jr. and D.R. Fulkerson [4]. (Refer to the next section for more details about this algorithm).
- **The Dinic's algorithm** or Dinitz's algorithm, which is a strongly polynomial algorithm for computing the maximum flow in a flow network. This algorithm was conceived in 1970 by A. Dinitz [6]. The algorithm runs in $O(V^2E)$ time; where V represents the number of vertices or nodes of the capacitated network and E represents the number of edges or arcs.
- **The Edmonds–Karp algorithm,** which is an implementation of the Ford – Fulkerson method for computing the maximum flow in a flow network in $O(V\,E^2)$ time. It was first published by Jack Edmonds and Richard Karp in 1972 [7].
- **The James Orlin algorithm,** which was published in 2013 [8]. The algorithm runs in $O(VE)$ time.

A multi-commodity flow problem consists of many distinct "commodities" that flow from the source(s) to the sink(s). For instance, there might be many goods that are produced at many factories, and are to be delivered to many given customers across the same transportation network. A minimum cost flow problem has multiple branches, where each branch (u, v) has its cost per unit of flow of $k(u, v)$, so that the cost of transmitting the flow $f(u, v)$ through the branch is $f(u, v).k(u, v)$. The aim is to transport particular value at minimum total cost from the source to the sink.

A circulation problem is another aspect of a flow network, which has a lower bound $l(u, v)$ for flow on the branch (u, v), as well as an upper bound $c(u, v)$ on it. Every branch also has a certain cost per unit of flow. In this problem, the sink is linked back to the source and the flow conservation rule holds for all vertices. Therefore, it is possible to control the total flow with the lower and upper bounds $l(t, s)$ and $c(t, s)$. The flow therefore *circulates* over the network, a fact which accounts for the name of this type of problems.

A graph with gains is a generalized network in which each link has a gain, which is a non-zero real number. In such a graph, if a link has a gain g, and an amount x flows into the edge at its tail, then an amount gx flows out at the head of this edge.
Another aspect of flow network is the source localization problem, where the pertinent algorithm anticipates localities of excessive flow through knowledge of flow distribution across a moderately monitored graph. Such an algorithm is useful for tracking mobile phone users and identifying sources of disease outbreaks since it can be implemented in arbitrary networks in cubic time or in tree networks in linear time.

3. The Ford and Fulkerson Algorithm

3.1. Definition

This section presents the Ford and Fulkerson method (also called the Ford and Fulkerson algorithm (FFA)), which is used in a flow network to solve the maximum flow problem [2, 3, 12, 63, and 65].
The approach pertaining to finding the augmented paths in a residual network is referred to simply as a "method" rather than an "algorithm" because this approach is not fully specified [9]. It is a broad approach that encompasses multiple implementations with differing execution times [2]. Ford–Fulkerson is a combination of names of the publishers of this algorithm, L.R. Ford Jr. and D.R. Fulkerson [4], who published it in 1956. The Edmonds–Karp algorithm is a common terminology used to define the Ford–Fulkerson method implementation. The method proposed by Ford and Fulkerson is dependent on three crucial concepts, which makes it a superior method among other flow algorithms and problems. These concepts include cuts, augmented paths and residual networks. Typically, these concepts or ideas are necessary and crucial aspects of the max-flow min-cut theorem. This section ends with detailed analysis of the Ford and Fulkerson method and some essential examples [2, 3, 12].

The Ford-Fulkerson method repeatedly increases the value of the flow. It begins with $f(u, v) = 0$ for all $u, v \in V$, giving a zero-initial value of the flow. At each step or iteration, the value of the flow in G is increased by finding an "augmenting path" in an associated "residual network" G_f . A path is called an augmenting path if there is some available capacity on all edges belonging to this path. Once we know the branches of an augmenting path in G_f , we can easily identify specific branches in G for which we can update the flow so as to increase the flow value. Although each step of the Ford-Fulkerson method increases the flow value, we should notice that the flow on any specific branch of G can increase or decrease. In fact, decreasing the flow on some branches might be necessary in order to enable the algorithm to transmit more flow from the source node to the sink node. We iteratively augment the flow till the algorithm terminates when the residual network has no more augmenting paths. The max-flow min-cut theorem shows that, upon termination, this procedure produces a maximum flow [2, 3, 12]. In other words, the concept behind the algorithm is as follows: as long as there is a path from the source (start node) to the sink (end node), with available capacity on all edges in the path, the source-to-sink flow might be augmented by utilizing the available capacity along this augmenting path. Then we find another augmenting path, and so on till no path can be found with available capacity on all its edges.

3.2. Some useful terminology

In order to implement and analyze the Ford-Fulkerson method, we need to introduce several additional concepts such as:

- **Source/Sink**: The source/sink vertex is such that all its edges are outward/inward ones, and none of them is inward/outward.
- **Residual networks**: Given a flow network G and a flow f , the residual network G_f consists of links with capacities that represent how we can change the flow on links of the flow network G. A link (edge) of the flow network G can permit an amount of additional flow equal to the link's capacity minus the actual flow f on that link. If that value is positive, we

place that link into the residual network G_f with a "residual capacity" of $c_f(u,v) = c(u,v) - f(u,v)$. The only links of the flow network G that are in the residual network G_f are those that can permit more flow. Those links (u,v) whose flow equals their capacity have $c_f(u,v) = 0$, and they are not in the residual network G_f. Moreover, we can define the **residual graph** as a graph which indicates additional possible flow. If there is a path from the source to the sink in the residual network, then there is a possibility to add more flow along that path that equals the minimum residual capacity on the edges of this path.

- **Minimal cut**: Also known as the "bottleneck capacity," which decides the maximum possible flow from s to t through an augmented path. In other words, the bottleneck capacity of the path is the minimum capacity of any edge on the path.
- **Augmenting paths**: An augmenting path p is a basic link from node s to node t in a residual network, provided there is a flow network $G = (V,E)$ and a flow f. Possibly, we might increase the link (u,v) flow of an augmenting path by the amount $c_f(u,v)$, following the definition of a residual network. That is possible with no violation of branch capacity on whichever of (u,v) and (v,u) is in the original flow network G. Actually, augmenting paths is achievable by considering any of the two kinds of edges: (1) Non-full forward edges, and (2) Non-empty backward edges.

Augmenting path theorem: A flow f is a maximum flow if and only if no augmenting paths exist.

3.3. Algorithm

Consider $G(V,E)$ as a flow network, and for every branch from u to v, let $c(u,v)$ be the capacity and $f(u,v)$ be the flow. Therefore, it possible to determine the maximum flow between the sink node t and source-node s. Table 1 shows the prerequisites that must be fulfilled after each step in the algorithm. Each of these prerequisites does not change after every stage of the algorithm:

Table 1: Prerequisites that must be fulfilled after each step in the algorithm

Capacity constraints	$\forall(u,v) \in E : f(u,v) \leq c(u,v)$	Edge Capacity defines the bounds for maximum flow along the path.
Skew symmetry	$\forall(u,v) \in E : f(u,v) = -f(u,v)$	Flow from u to v equals the negative of flow in the reverse direction.
Flow conservation	$\forall u \in V : u \neq s \; and \; u \neq t \rightarrow \sum_{w \in V} f(u,w) = 0$	Unless the node is a source (producer) or a sink (consumer), the net flow is zero.
Value (f)	$\sum_{(s,u) \in E} f(s,u) = \sum_{(v,t) \in E} f(v,t)$	Exiting flow at s equals flow entering t.

Table 1 shows that every round in the algorithm results in a legal flow within the network. Therefore, the residual network $G_f(V,E_f)$ can be defined as a network with a capacity of $c_f(u,v) = c(u,v) - f(u,v)$ and a zero flow. Notably, it is possible that flow from v to u be permitted in the residual network, but prohibited in the initial network. This happens when $f(u,v) > 0 \; and \; c(v,u) = 0$ for then $c_f(v,u) = c(v,u) - f(v,u) = -f(v,u) = f(u,v) > 0$.

3.4. Entire algorithm

The following steps describe the Ford-Fulkerson algorithm used to solve maximum flow problems:

Start

Inputs: Given is a network $G = (V, E)$ with flow capacity c, a source node s, and a sink node t
Output: Compute a flow f from s to t of maximum value

1. $f(u, v) \leftarrow 0$ for all edges (u, v) . [Start with initial flow as zero].
2. While there is a path p (to be denoted as an augmenting path) from s to t in G_f, such that $c_f(u, v) > 0$ for all edges $(u, v) \in p$:
   a) Find $c_f(p) = \min\{c_f(u, v): (u, v) \in p\}$
   b) For each edge $(u, v) \in p$
      i) $f(u, v) \leftarrow f(u, v) + c_f(p)$ (Send flow along the path)
      ii) $f(v, u) \leftarrow f(v, u) - c_f(p)$ (The flow might be "returned" later)

- "$\leftarrow$ " denotes assignment. For instance, "largest $\leftarrow$ item" means that the value of largest changes to the value of item.
- "return" terminates the algorithm and outputs the following value.

End

Note that s will never reach t whenever no more paths exist in the residual network as shown in step 2. Assuming S repsresnets the set of nodes reachable by s in the residual network, then the original network's total capacity of edges beginning from S is equivalent to the net flow found from s to t. It also denotes the upper limit for all potential flows. Therefore, the final flow is maximum. Further details on Max-flow Min-cut theorem are reported in the next section.
If the network $G(V, E)$ consists of many sources and many sinks, then it is worked out as follows: Assume that $T = \{t \mid t \text{ is a sink}\}$ and $S = \{s \mid s \text{ is a source}\}$. Add a new source s^* with an edge (s^*, s) from s^* to every node $s \in S$, with capacity $c(s^*, s) = d_s\big(where\ d_s = \sum_{(s,u)\in E} c(s, u)\big)$. Similarly, add a another sink $t^*$with an edge (t, t^*) originating from each node $t \in T$ to t^*, with capacity $c(t, t^*) = d_t\ \big(where\ d_t = \sum_{(v,t)\in E} c(v, t)\big)$. Finally, use the Ford-Fulkerson's algorithm. Similarly, replace a node u if it has a capacity constraint d_u with two nodes u_{in}, u_{out}, connected by an edge (u_{in}, u_{out}), which possesses a capacity $c(u_{in}, u_{out}) = d_u$. After this, apply the Ford–Fulkerson algorithm.

3.5. The Algorithm Complexity

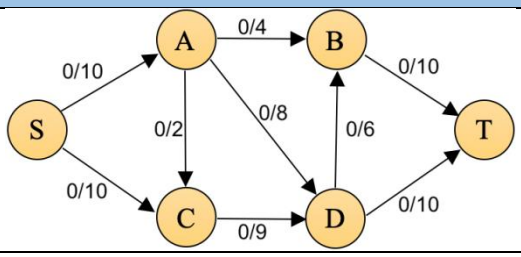
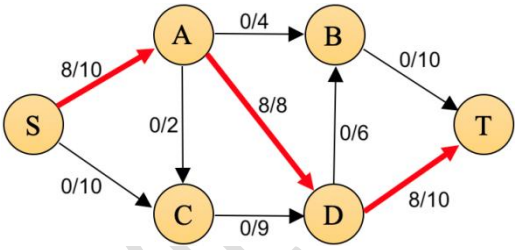
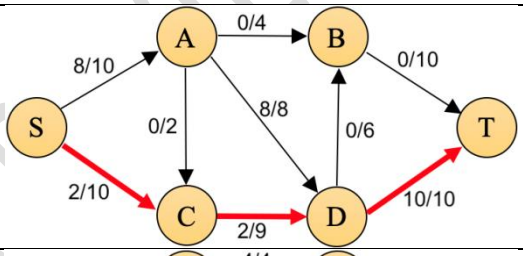
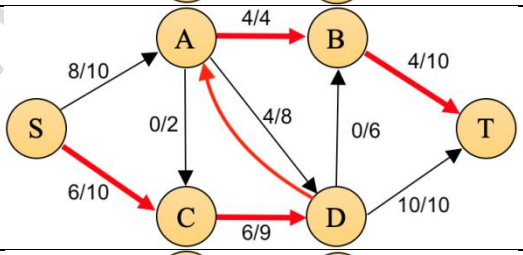
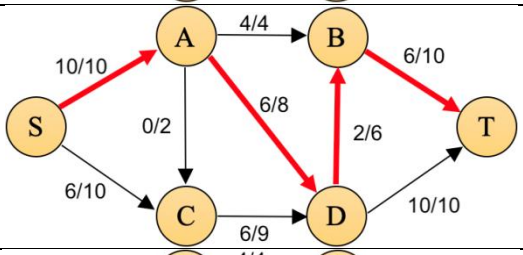
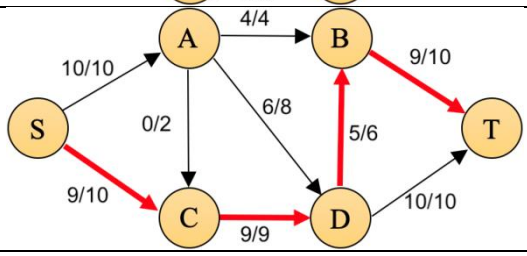
Adding more augmenting paths to an already established network flow has a limit, where no more augmenting paths are available in the network. Nevertheless, there is no guarantee this limit may be achieved; therefore, the correct results are achievable when the algorithm terminates. Issues regarding the termination of this algorithm will come later in this paper. In the case that the algorithm runs forever, the flow could not even converge towards the maximum flow. However, this situation only happens with irrational flow values. When the capacities are integers, the runtime of Ford–Fulkerson is bounded by $O(Ef)$ where E is the number of branches in the network and f is the maximum flow in the network. This is because each augmenting path can be found in $O(E)$ time and it increases the flow by an integer amount of at least 1, with the upper bound of f.

The Edmonds–Karp algorithm is another variant of the Ford–Fulkerson' algorithm, which executes in $O(VE^2)$ regardless of the maximum flow value, and guarantees termination.

3.6. Examples:

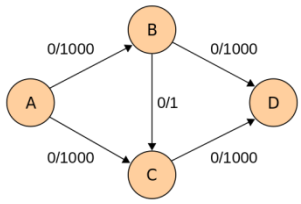
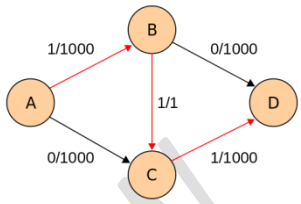
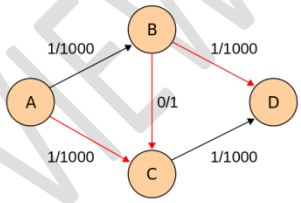
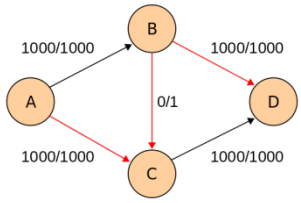
Example 1

Table 2. Example 1

Augmenting path	Bottleneck Capacity	Resulting Flow Network
	Initial flow network	
S, A, D, T	$min\left(c_f(S,A), c_f(A,D), c_f(D,T)\right)$ $= \min\big(c(S,A) - f(S,A), c(A,D) - f(A,D),\ c(D,T) - f(D,T)\big)$ $= \min(\ 10 - 0{,}8 - 0, 10 - 0) = 8$	
S, C, D, T	$min\left(c_f(S,C), c_f(\ C,D), c_f(D,T)\right)$ $= \min\big(c(S,C) - f(S,C), c(C,D) - f(C,D),\ c(D,T) - f(D,T)\big) = 2$	
S, C, D, A, B, T	$\min\ (c_f(S,C), c_f(\ C,D), c_f(D,A),$ $c_f(\ A,B), c_f(B,T)\ ) = 4$	
S, A, D, B, T	$\min\ (c_f(S,A), c_f(\ A,D), c_f(D,B),$ $c_f(\ B,T))\ = 2$	
S, C, D, B, T	Final flow network: $\min\ (c_f(S,C), c_f(\ C,D), c_f(D,B),$ $c_f(\ B,T))\ = 3$	
	Max. Flow = (8+2+4+2+3) = 19	

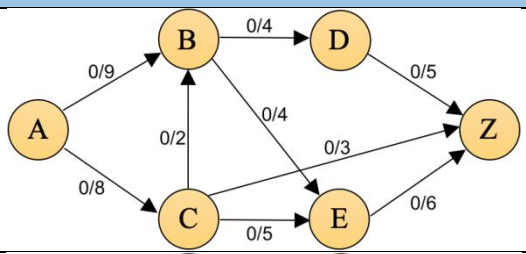
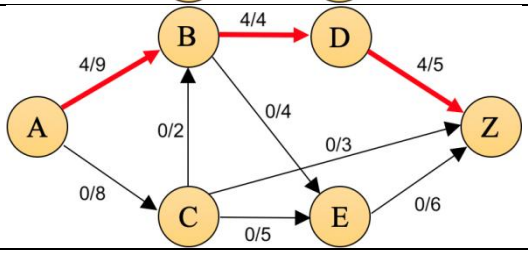
Example 2

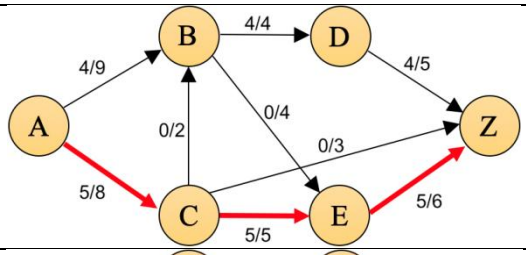
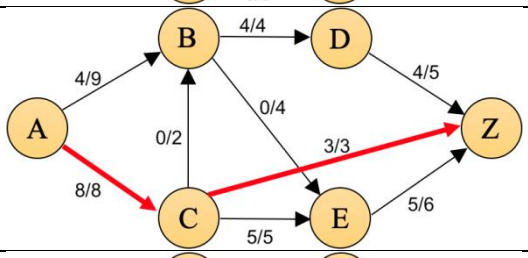
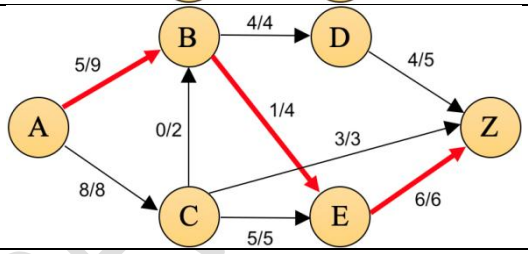
Table 3. Example 2

Path	Capacity	Resulting Flow Network
	Initial flow network	
A, B, C, D	$min\left(c_f(A,B), c_f(B,C), c_f(C,D)\right)$ $= \min\left(c(A,B) - f(A,B), c(B,C) - f(B,C),$ $c(C,D) - f(C,D)\right)$ $= min(1000 - 0, 1 - 0, 1000 - 0) = 1$	
A, C, B, D	$min\left(c_f(A,C), c_f(C,B), c_f(B,D)\right)$ $= \min\left(c(A,C) - f(A,C), c(C,B) - f(C,B) -\right.$ $\left.c(B,D) - f(B,D)\right)$ $= min(1000 - 0, 0 - (-1), 1000 - 0) = 1$	
After 1998, more steps…		
	Final flow network	

Example 3

Table 4. Example 3

Augmenting path	Bottleneck Capacity	Resulting Flow Network
	Initial flow network	
A, B, D, Z	$min\left(c_f(A,B), c_f(B,D), c_f(D,Z)\right)$ $= \min\left(c(A,B) - f(A,B), c(B,D) -\right.$ $\left.f(B,D),\ c(D,Z) - f(D,Z)\right)$ $= \min(9 - 0{,}4 - 0{,}5 - 0) = 4$	

A, C, E, Z	$min\left(c_f(A,C), c_f(C,E), c_f(\ E,Z)\right)$ $= \min\left(c(A,C) - f(A,C), c(C,E) - f(C,E),\ c(E,Z) - f(E,Z)\right) = 5$	
A, C, Z	$min\left(c_f(A,C), c_f(C,Z)\right)$ $= \min\left(c(A,C) - f(A,C), c(C,Z) - f(C,Z)\right) = 3$	
A, B, E, Z	Final flow network $min\left(c_f(A,B), c_f(B,E), c_f(\ E,Z)\right)$ $= 1$	
	Max. Flow = (4+ 5+ 3+ 1) = 13	

3.7. The Non-terminating Case

Understandably, it is widely accepted that the Ford-Fulkerson algorithm does not need to terminate for so as to find the maximum flow, specifically, when the arc capacities take irrational values. Although all non-terminating instances converge to a limit flow, this limit flow does not necessarily imply the maximum flow of the network. Therefore, it is possible to exceed the limit flow and restart the algorithm, which justifies the classification of this method as a transfinite algorithm. Based on transfinite runtime-time analysis of the Ford-Fulkerson algorithm by Backman & Huynh [11], the worst-case execution time is $w^{\theta(|E|)}$ using ordinal numbers. Similarly, Backman & Huynh [11] show it is viable to model an Euclidean algorithm via Ford-Fulkerson on an auxiliary network. They determined that running the above example on a pair of incommensurable numbers could yield a robust non-terminating example.

Another non-terminating example is given below [10]. This example considers the flow network that is indicated on Figure 4. It has a source s, a sink t, and the edge capacities $e_1, e_2\ and\ e_3$, respectively with values $1, r = \frac{\sqrt{5}-1}{2} and\ 1$, while the capacity of each of the other edges is an arbitrary integer $M \geq 2$. The constant r is selected carefully so that $r^2 = 1 - r$ $(r^{-1} = 1 + r)$. Table 5 defines how we apply augmenting paths: $p_1 = \{s, v_4, v_3, v_2, v_1, t\}$, $p_2 = \{s, v_2, v_3, v_4, t\}\ and\ p_3 = \{s, v_1, v_2, v_3, t\}$.

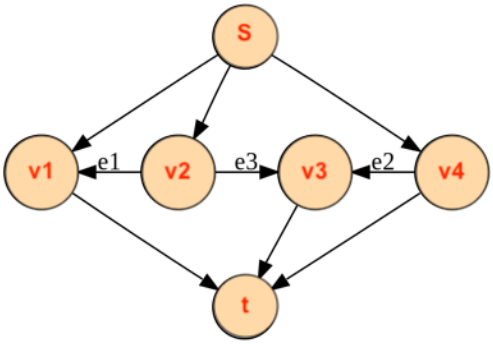


Figure 4. A flow network for which the Ford-Fulkerson does not terminate

Table 5. Relation between augmenting paths and residual capacities for the network in Fig. 4

Step	Augmenting path	Sent flow	Residual capacities		
			e_1	e_2	e_3
0			$r^0 = 1$	r	1
1	$\{s, v_2, v_3, t\}$	1	r^0	r^1	0
2	p_1	r^1	r^2	0	r^1
3	p_2	r^1	r^2	r^1	0
4	p_1	r^2	0	r^3	r^2
5	p_3	r^2	r^2	r^3	0

For some $n \in N$, the residual capacities for edges e_1, e_2 *and* e_3 take the values r^n, r^{n+1} *and* 0, respectively, after each of step 1 and step 5.This implies that the residual capacities of these edges will take similar format regardless of whether we augment paths p_1, p_2 *and* p_3 infinitely. The net flow of the network at step 5 takes the value $1 + 2(r^1 + r^2)$. The total flow converges towards

$$1 + 2\sum_{i=1}^{\infty} r^i = 1 + 2r/(1-r) = 1 + 2r/r^2 = 1 + 2r^{-1} = 1 + 2(1+r) = 3 + 2r$$

if we use the augmenting paths repeatedly. Nevertheless, it results in flow value of $2M + 1$, if we direct M units of flow along sv_1t ,1 unit of flow along sv_2v_3t , and M units of flow along sv_4t . Consequently, the algorithm runs infinitely with no convergence at the maximum flow [10].

4. The Max-Flow Min-Cut Theorem

4.1. Definitions and statements

The max-flow min-cut theorem holds when maximum flow between the source s and the sink t is equate to the total capacity of the links (edges) in a minimum cut [2, 3, 5, 13, 63, 64]. The theorem is considered a unique instance of duality theorem for linear optimization, which might be applied to develop the Kőnig's theorem and Menger's theorem [13]. In other words, The Ford-Fulkerson method iteratively augments the flow along the augmenting paths until it reaches a maximum value of the flow [4]. Therefore, the following questions are necessary: When do we actually get the maximum flow in a flow network? In addition, how do we know when to terminate the algorithm? To answer such questions, the max-flow min-cut theorem expresses that a flow is maximum if its residual network has no augmenting path. Actually, the theorem has two significant parts: the maximum flow through a flow network, and the minimum capacity of a cut of the flow network. To express the theorem, each of these parts should be defined first.
Let $N = (V, E)$ be a directed flow network, where V represents the set of nodes (vertices) and E the set of links (edges). Let $s \in V$ and $t \in V$ be the source and the sink of N, respectively. A link (edge) (u, v) is characterized by a mapping $c: E \to \mathbb{R}^+$ indicated by c_{uv} or $c(u, v)$ where $u, v \in V$. It indicates the maximum value of flow that can pass through this link.

4.2. Analyzing the Algorithm: Flows and Cuts

The next objective is to illustrate that the Ford-Fulkerson Algorithm returns flow that is possibly the maximum value in any flow in G.
***Flows*:** The function $f: E \to \mathbb{R}^+$ represents a flow mapping f_{uv} or $f(u, v)$, subject to the following two restrictions:

1. Capacity Constraint: Given any link (u, v) in $E, f_{uv} \leq c_{uv}$.
2. Flow Conservation: The equality below holds given any node v other than s (the source) and t (the sink):

$$\sum_{\{u:(u,v)\in E\}} f_{uv} = \sum_{\{w:(v,w)\in E\}} f_{vw}.$$

Flow is typically a representation of transportation of a commodity or fluid in the direction of each link through a network. The flow through the link cannot exceed its maximum limit, which is called the capacity constraint. In addition, the conservation constraint holds, which states that amount of flow through each node (other than the source and sink nodes) must equal the amount leaving it.
The definition of the magnitude of flow through node v is shown below:

$$|f| = \sum_{\{v:(s,v)\in E\}} f_{sv} = \sum_{\{v:(v,t)\in E\}} f_{vt}$$

Where the node s is the source node and t is the sink node. Based on the fluid or commodity comparison, it denotes the volume of fluid or quantity of commodity that enters the network at the source, and then leaves at the sink. Note that the same amount of flow entering at the source must leave at the sink to uphold the conservation rule. For any given network, the maximum flow problem seeks to achieve the maximum flow [2, 3].

Maximum Flow Problem. Maximize $|f|$, which aims to achieve the highest possible flow from s to t.

Cuts: The second part of the max-flow min-cut theorem refers to another side of the network: which is a group of cuts. For instance, the s-t cut $C = (S, T)$ in a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$. In other words, an s-t cut represents partitioning of network vertices into two disjoint parts: one part including the source node, with the other containing the sink node. The cut set X_C of a cut C constitutes the set of links that joins the source part S of the cut to its sink part T:

$$X_C := \{(u, v) \in E : u \in S, v \in T\} = (S \times T) \cap E$$

Therefore, if we remove all the edges in the cut set of C, then we cannot have a possible flow, since no link exists to establish a connection from the source part S of the cut to its sink part T. If f is a flow, then the **net flow** $f(S, T)$ across the cut (S, T) is expressed as

$$f(S,T) = \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u)$$

The sum of capacities of its links (edges) represents the capacity of the corresponding s-t cut,
$c(S,T) = \sum_{(u,v)\in X_C} c_{uv} = \sum_{(i,j)\in E} c_{ij} d_{ij}$,
where $d_{ij} = 1\ if\ i \in S\ and\ j \in T$, 0 otherwise.
Alternatively, we could state that the **capacity** of the cut (S, T) equals the following:

$$c(S,T) = \sum_{u\in S}\sum_{v\in T} c(u,v).$$

Usually, there are various cuts in the network, so that it becomes inherently complex to find the cuts with smaller capacities. Notably, a cut with the smallest cut capacity all over a network is called a minimum cut [2, 3]. Such a minimum cut might not be unique.

Minimum s-t Cut Problem. Minimize $c(S, T)$, that is, determine S and T such that the capacity of the *S-T cut* is minimal.

For the sake of simplicity and clarity, let us elaborate the difference between flow and capacity of a cut. In the case of capacity, we count only the capacities of branches passing from S to T, while disregarding branches going in the opposite direction. In the case of flow, it is the flow from S to T minus the flow in the opposite direction (i.e. from T to S) [2].

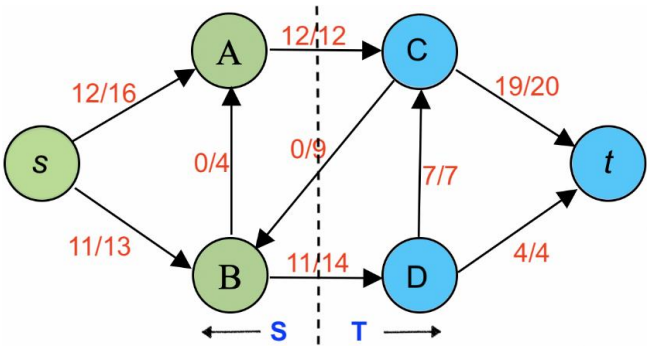


Figure 5. A cut (S,T) for a specific flow network, where $S=\{s,A,B\}$ and $T=\{C,D,t\}$. The vertices in S are green, and the vertices in in T are blue. The net flow across (S,T) is $f(S,T)=23$, and the capacity is $c(S,T)=26$.

Figure 5 shows the cut $(\{s,A,B\},\{C,D,t\})$ in a certain flow network. Its total flow across the cut is $f(A,C)+f(B,D)-f(C,B)=12+11-0=23$, while the capacity is: $c(A,C)+c(B,D)=12+14=26$.

The lemma below shows that the total flow across any cut is uniform for any given flow, which is the same as value of the flow $|f|$.

Lemma: Assume that a flow f goes through a flow network G, where s is the source and (S,T) represents some cut in G. Then, the total flow across (S,T) is given by: $f(S,T)=|f|$.

The important max-flow min-cut theorem, stipulates that the maximum flow value and the minimum cut capacity are equivalent.

4.3. Main theorem

Notably, this theorem shows that there is a link between the capacity of a minimum cut and the maximum flow through a network. In other words, a maximum flow *s-t* is equal to the smallest *s-t* cut capacity. In fact, if f is a flow in a flow network $G=(V,E)$ with source s and sink t, then the following conditions are equivalent [2]:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f|=c(S,T)$ for some cut (S,T) of G.

4.4. Linear program formulation

The max-flow problem and the min-cut problem can be expressed as two primal-dual linear programs, as expressed mathematically in Table 6.

Table 6. The LP of maximum flow and its dual (minimum cut)

	Max-flow (Primal)	Min-cut (Dual)
variables	$f_{uv}\ \forall(u,v)\in E$ [a variable per edge]	$d_{uv}\ \forall(u,v)\in E$ [a variable per edge] $z_v\ \forall\ v\in V\backslash\{s,t\}$ [a variable per non-terminal node]
Objectives	Maximize $\sum_{v:(s,v)\in E} f_{sv}$ [max total flow from source]	Minimize $\sum_{(u,v)\in E} c_{uv}d_{uv}$ [min total capacity of arcs in cut]
Constraints	Subject to $f_{uv}\le c_{uv} \quad \forall(u,v)\in E$ $\sum_u f_{uv}-\sum_w f_{vw}=0 \quad v\in V\backslash\{s,t\}$ [every edge and non-terminal node have a constraint]	Subject to $d_{uv}-z_u+z_v\ge 0 \quad \forall(u,v)\in E,\ u\neq s, v\neq t$ $d_{sv}+z_v\ge 1 \quad \forall(s,v)\in E$ $d_{ut}-z_u\ge 0 \quad \forall(u,t)\in E$ [a constraint per edge]
Sign constraints	$f_{uv}\ge 0 \quad \forall(u,v)\in E$	$d_{uv}\ge 0 \quad \forall(u,v)\in E$ $z_v\in\mathbb{R} \quad \forall v\in V\backslash\{s,t\}$

In a straightforward manner, the max-flow linear program can be obtained from the primal column of Table 6. Likewise, the dual linear program is obtained by applying the algorithm explained in the dual column of Table 6. The resulting linear programs need some clarifications. To understand the variables in the min-cut linear program consider:

$$d_{uv} = \begin{cases} 1, & if\ u \in S\ and\ v \in T\ (the\ edge\ uv\ is\ in\ the\ cut) \\ 0 & ,otherwise \end{cases}$$

$$z_u = \begin{cases} 1, & if \quad u \in S \\ 0, & otherwise \end{cases}$$

Overall, the aim of this minimization entails summing of the capacities of all edges (branches) that are contained in the cut.
These variables represent a valid cut due to the defined constraints:

- The constraints $d_{uv} - z_u + z_v \geq 0$ (equivalent to $d_{uv} \geq z_u - z_v$) warrants that the edge (u, v) is located in the cut ($d_{uv} \geq 1$) , for every non-terminal nodes $u, v,$ only when u is present in S and v is present in T.
- The constraints $d_{sv} + z_v \geq 1$ (equal to $d_{sv} \geq 1 - z_v$) warrants that, if v is in T, then the edge (s, v) is located in the cut (i.e. s is present in S by default).
- The constraints $d_{ut} - z_u \geq 0$ (equivalent to $d_{ut} \geq z_u$) guarantee that, if u is in S, then the edge (u, t) is located in the cut because t is present in T by default.

Notably, there is no guarantee that a specific edge must be present in a cut since this is a minimization problem. The only guarantee is that each edge present in a cut is taken into consideration in the summation in the objective function. Therefore, the duality theorem in linear programming follows from the max-flow min-cut theorem of equality. It states that an optimal solution y^* exists for every primal program that has an optimal solution x^*, where the resulting optimal values from both solutions are equal.

4.5. Examples

Example 1:

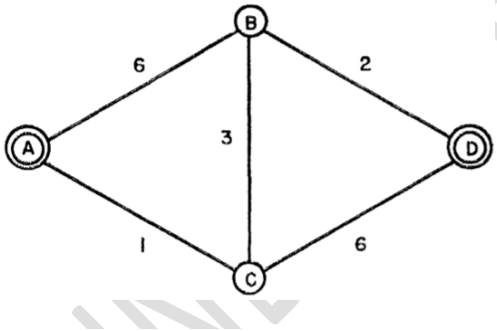


Figure 6. A simple capacitated (flow) network

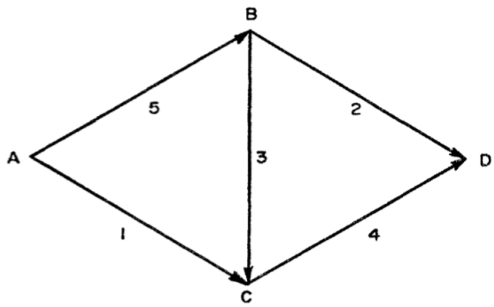


Figure 7. The maximum flow for the network in Figure 6

To demonstrate the aforementioned concepts, consider the simple flow network of Figure 6. Assume that the capacities of the edges are as shown in Figure 6, and that it is required to find the maximum flow from node A to node D. First, we need to define a cut. Generally, a cut is any collection of edges, which totally separates A from D. Therefore, in this example there are four possible cuts: {AB and AC}; {BD and CD}; {AB, BC and CD}; or {BD, AC, and BC}. The value of a cut is the sum of the capacities of its edges, and the Min Cut Theorem simply says that the value of the minimum cut is exactly equal to the maximum flow. Therefore, for Figure 6, it can be noticed that there is a single minimum cut, which is that composed of edges BD, AC, and BC with a value of 6. The corresponding maximum flow is shown in Figure 7. It can be observed that each edge of the minimum cut is saturated (i.e., used to full capacity) as would naturally be necessary. By contrast, since the minimum cut is unique, none of the branches not belonging to it is saturated. It is intuitively clear that the minimum cut definitely gives an upper bound on the maximum flow, but the fact that it is also a lower bound is not nearly as evident.

Example 2:

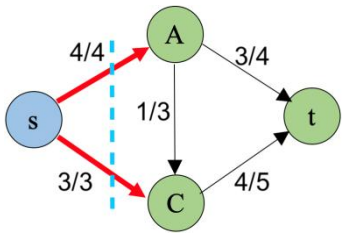


Figure 8. A network with a value of flow that equals the capacity of an s-t cut

The network shown in figure 8 has a flow value of 7, where a numerical expression x/y is formatted along each arrow. As usual, this format represents flow (x) and capacity (y), respectively and the flow across any of the four cutsets of the network has same value of 7. In particular, the flow originating from the source (the flow across the source vertex cutset) has a value of 4+3=7, as does the flow into the sink (the flow across the sink vertex cutset) (3+4=7).

Notably, blue and pale green vertices form the subsets S and T of an s-t cut, whose cutset has red edges intersected with a vertical dashed blue line denoting the cut. Following the max-flow min-cut theorem, (equivalence of flow value and capacity of s-t cut equal to 7), the value of the flow and the capacity of the s-t cut are equally optimal in the graph. Another crucial observation is that the flow through the red edges is at full capacity (saturated) since this cutset is one of the 'bottlenecks' in the network. On the contrary, some extra or residual capacity exists on the right side of the network. Particularly, figure 8 depicts a scenario where flow from vertex A to vertex C is arbitrarily set as 1 although this should not be necessarily the case. If no flow existed between vertices A and C, then the sink inputs will adjust to 4/4 and 3/5 while the total flow remains unchanged (4+3=7). Otherwise, doubling the inputs from vertex A to vertex C will cause a change in the sink' inputs (to become 2/4 and 5/5), but the total flow will remain the same (2+5=7).

Example 3 (Example 1 of Section 2 revisited):

Figure 9 demonstrates that there exists a unique minimal cut $\{SA, AC, CD\}$ whose branch AC is irrelevant since it points in the opposite direction. The capacity of this minimum cut is the sum of edge capacities of branches SA and CD. Therefore, we obtain

Min. Cut: Value of Max. flow = Capacity of Min. cut = **19.**

in agreement with our earlier solution in Example 1 of Section 2.

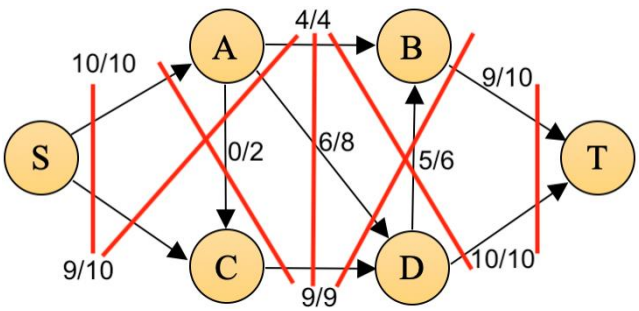


Figure 9. A 9-branch flow network with all 7 possible cuts for Example 3 with a maximum flow value of 19 across each of them, with only one of them saturated.

Example 4 (Example 3 of Section 3 revisited):

From Example 3 in the previous section 3, we obtain
Min. Cut Value of Max. flow = Capacity of Min. cut = **13**

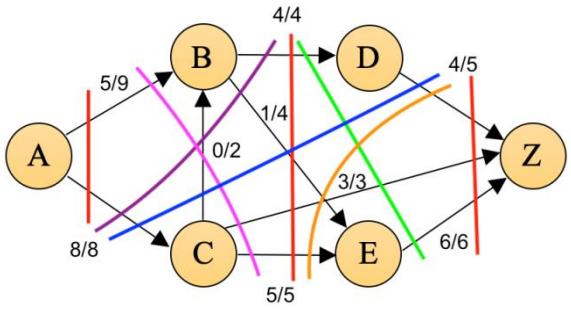


Figure 10. A 9-branch flow network with all 8 possible cuts for Example 4

The network has a unique minimum cut-set: $C = \{BD, CZ, EZ\}$ **of minimal capacity**$= (4 + 3 + 6) = 13$. Each of the other cutsets shown in Fig. 10 has a flow of 13 and is unsaturated.

5. Reductions and Transformations Techniques

5.1. Introduction

In a flow network problem like the maximum flow problem and the shortest path problem, it is often required to try to simplify the given flow network before using different techniques or algorithms available for its solution. This section presents various types of reductions and transformations [14-16] that can lead to considerable simplification. In particular, we present a star-delta transformation that is similar to the one used in electrical circuits [66]. The network in the maximum flow problem contains edges, each of which having a certain flow capacity (indicated by a positive integer value) associated with it. We are required to find the maximum possible flow between the source node s and the terminal node t [4, 18]. Our target in the maximum flow transformations is to prove that in many cases the original flow network can be simplified by the star-delta and other analogous transformations. Such simplifications can significantly reduce the computations involved in using the Max-flow Min Cut theorem, and in particular when the maximum flow between several pairs of points is required. Moreover, these simplifications might also reduce the network to a tree, in which the maximum flow between any pair of terminal points is readily observable.
To begin, let us mention the three most crucial reduction rules (flow network simplifications):

5.2. Rule 1

Two edges in series with capacities C_a *and* C_b can be replaced by a single edge with a capacity equal to $min\ (C_a, C_b)$ as can be seen in Figure 11

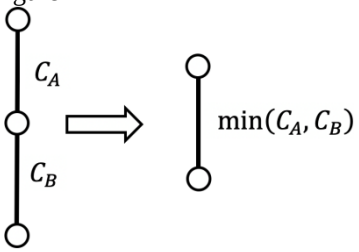


Figure 11. Network simplification of branches in series

5.3. Rule 2

Two edges in parallel with capacities C_a and C_b can be replaced by a single edge with capacity equal to $(C_a + C_b)$ as can be seen in Figure 12.

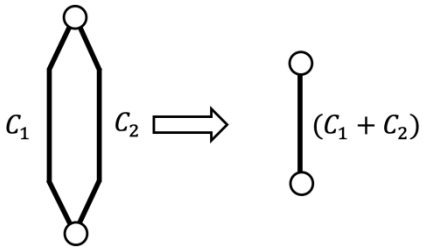


Figure 12. Network simplification of branches in parallel

5.4. Rule 3

An edge between two points might be 'shorted' if its capacity exceeds or equals the sum of capacities of all other edges incident on one of the two points.

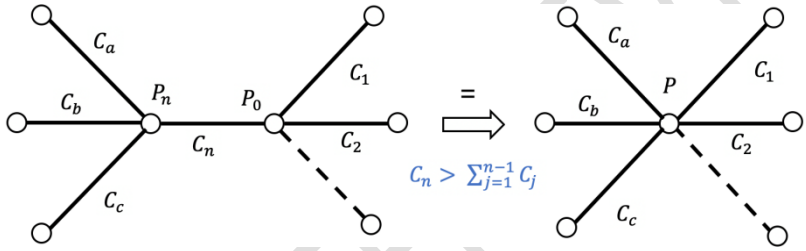


Figure 13. Network simplification of branches for Rule 3

5.5 Star-Delta Transformation

Now consider the star-delta transformation [14-17]. Let us consider a given flow network that contains a 'delta network'- viz, three points that are joined together, two at a time, with three edges, each of which having a finite capacity (See Figure 14). The task is to replace this delta network by a star network (dotted lines in Figure 14) and to select the capacities $C_a, C_b, and\ C_c$ such that the maximum flow through this new network remains the same as before. Assume the flow through the delta is as shown in Figure 14. Then,

$$C_1 \geq f_1 \geq -C_1, \quad (1)$$
$$C_2 \geq f_2 \geq -C_2, \quad (2)$$
$$C_3 \geq f_3 \geq -C_3. \quad (3)$$

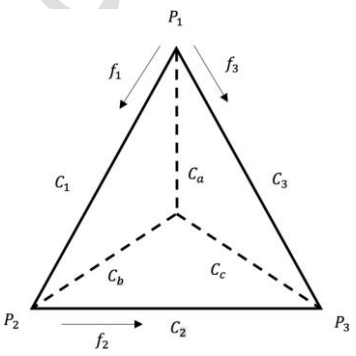


Figure 14. The flow through the delta network

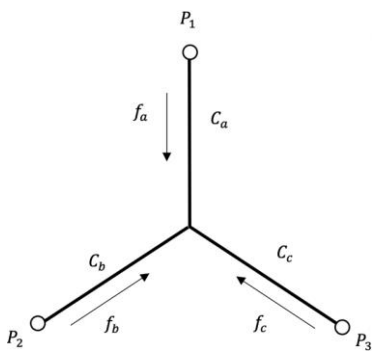


Figure 15. The flow through the wye network

The sign of the flow should be allowed to be either negative or positive since the actual direction of flow is unknown. Adding equation (1) to equation (3) we obtain,

$$C_1 + C_3 \geq f_1 + f_3 \geq -(C_1 + C_3). \quad (4)$$

Therefore, if in the star we let

$$C_a = C_1 + C_3, \quad (5)$$

Then it can be noticed from equation (4), that this edge can handle the flow from (or to) P_1 that originally passed through edges 1 and 3 of the delta.
The same argument applies for P_2 and P_3, so that we let

$$C_b = C_1 + C_2, \quad (6)$$
$$C_c = C_2 + C_3, \quad (7)$$

Therefore, if we select capacities $C_a, C_b, and\ C_c$ according to equations (5), (6), and (7) we notice that any flow through the original 'delta-network' can also be handled by the 'star-network'. Likewise, any flow that we select for the star-network can similarly be handled by the original delta-network.
Let us assume that the flow through the star-network is as shown in Figure 15 such that

$$f_a + f_b + f_c = 0, \quad (8)$$
$$C_a \geq f_a \geq -C_a, \quad (9)$$
$$C_b \geq f_b \geq -C_{b,} \quad (10)$$
$$C_c \geq f_c \geq -C_c. \quad (11)$$

Now, we can find $f_1, f_2, and\ f_3$ in Figure 14 where

$$f_a = f_1 + f_3, \quad (12)$$
$$f_b = f_2 - f_1, \quad (13)$$
$$f_c = -f_2 - f_3, \quad (14)$$

and such that equations (1), (2), and (3) are satisfied. Note that if equations (12) and (13) are satisfied it will follow from equation (8) that equation (14) is also satisfied.
Firstly, we solve for f_2 and f_3 in equations (12) and (13) and then rewrite equations (1), (2), and (3) to get

$$C_1 \geq f_1 \geq -C_1, \quad (1)$$
$$C_2 \geq f_1 + f_b \geq -C_2, \quad (2')$$
$$C_3 \geq f_a - f_1 \geq -C_3, \quad (3')$$

Then, it follows that we should choose f_1 where

$$\min(C_1, C_2 - f_b, C_3 + f_a) \geq f_1 \geq \max(-C_1, -C_2 - f_b, -C_3 + f_a). \quad (15)$$

This can be done provided each term in the right-hand side of equation (15) is less than or equal to each term on the left-hand side of it. Let the right terms be $R_1, R_2, and\ R_3$ correspondingly and the left terms be $L_1, L_2, and\ L_3$. Subsequently, we must satisfy each of the nine inequalities that are shown below

$L_1 \geq R_1$ (definition)	$L_2 \geq R_1$ (6), (10)	$L_3 \geq R_1$ (5), (9)
$L_1 \geq R_2$ (6), (10)	$L_2 \geq R_2$ (definition)	$L_3 \geq R_2$ (7), (8), (11)
$L_1 \geq R_3$ (5), (9)	$L_2 \geq R_3$ (7), (8), (11)	$L_3 \geq R_3$ (definition)

Hence, f_1 (and thus f_2 and f_3) can be found and indeed f_1 might be selected as max (R_1, R_2, R_3) or min (L_1, L_2, L_3). It can be shown that from the aforementioned relations, by applying equations (.5), (6), and (7) any delta network can be transformed into an equivalent star network. As a result, the transformations from a wye to a delta is given by:

$$C_1 = {}^1\!/_2\,[C_a + C_b - C_c], \quad (16)$$

$$C_2 = {}^1\!/_2\,[C_b + C_c - C_a], \quad (17)$$
$$C_3 = {}^1\!/_2\,[C_c + C_a - C_b], \quad (18)$$

5.6 Example

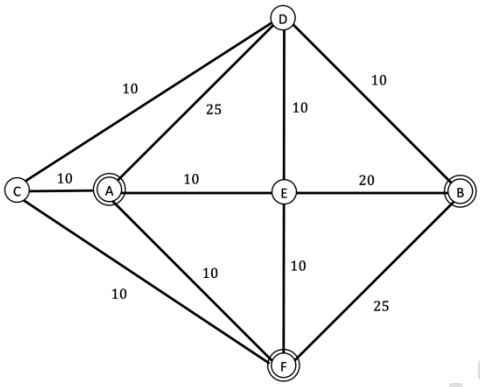


Figure 16. A capacitated network of 11 branches

In this example, it is required to find the maximum flow between node A and node B in the network shown in Figure 16. As an offshoot of the solution procedure, we will be able to go further and consider F as a terminal point in the network and ask to find the maximum flow between node A and node F and between node B and node F.

It can be noticed that, the node C can be eliminated since this node is an intermediate point at the center of a wye that can be transformed to a delta comprising three branches of capacity 5 each. This transformation is done by applying equations (16), (17), and (18), and after combining the resulting parallel edges between node A and node D and between node A and node F, the result is as shown in Figure 17a. It can be observed from the network that the capacity of an edge AD is greater than the sum of the other edges into D; thus, we will apply *rule 3* and A *and* D will be combined into a single point, and the emerging parallel edges are combined according to *rule 2* as can be seen the result in Figure 17b. Now node E becomes a logical choice to eliminate as the center of a wye to be transformed into a delta, with the emerging parallel edges being combined as shown in Figure 17c. Finally, this delta is replaced by an equivalent wye as shown in Figure 17d. The maximum flows between pairs of terminal points can be found since there is a prominent path between the ends of each pair of terminal points. The value of each maximum flow is equal to the minim capacity of the edges of the corresponding path where:

$f_{AB} = 50, \quad f_{Af} = 50, \quad f_{BF} = 55.$

Note that all of the three transformations so far discussed were employed in obtaining this result. To find the actual paths that a flow should follow it is necessary merely to start with Figure 17d and work successively backwards.

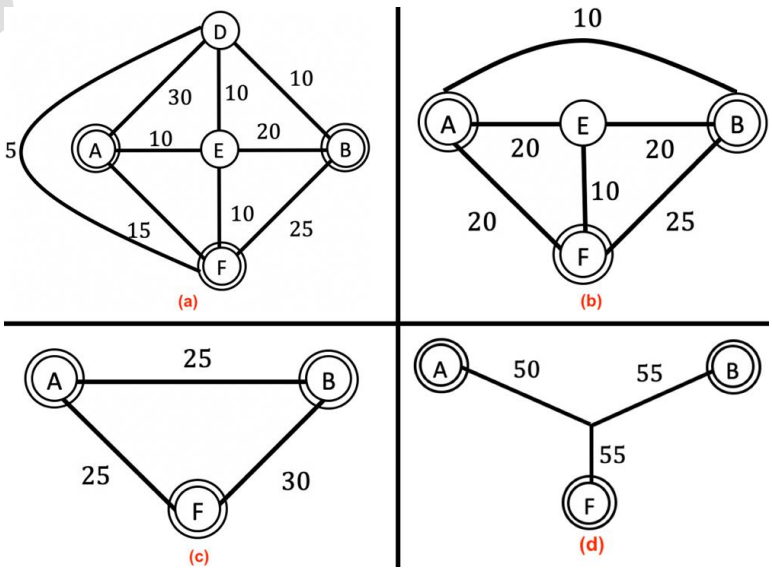


Figure 17. Transformation procedure for the flow network in Figure 16

6 The Decomposition Technique

The decomposition technique involves tuning a complicated network (complex system) to adopt the possible states of a keystone element K_1 or the possible combinations of states of many keystone elements [5, 21-25, 62]. In other words, this method entails a single application or multiple applications of the law of total probability. In its simplest form, it involves selecting a keystone element and then computing the reliability of the network twice: once as if the keystone element did not succeed ($R = 0$) and once as if the keystone element were good ($R = 1$). After that, the method combines these two probabilities to get the reliability of the system, since at any given time the key component will be failed or operating. The Venn graph in Figure 18 shows the event S, which signifies that the system is working successfully. This event can be partitioned in the form of a union of two dependent events that are mutually exclusive, namely (i) $K_1 \cap S$, which indicates that the keystone branch is in the operating state and the system is working and (ii) $\overline{K}_1 \cap S$, which indicates that the keystone branch is in the unsuccessful state and the system is functioning correctly.

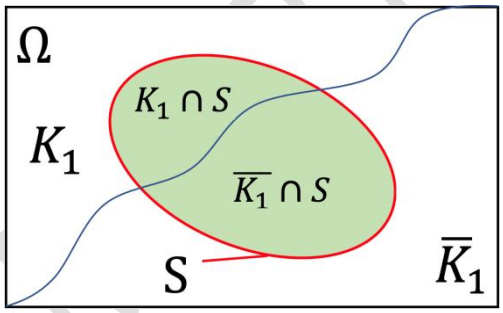


Figure 18. Event S, which indicates a fully functional system, partitioned as the union of a pair of mutually exclusive events: $K_1 \cap S$ and $\overline{K}_1 \cap S$

The technique's main objective is to decompose the system graph into two sub graphs $K_1 \cap S$ and $\overline{K}_1 \cap S$, each of which is with a simpler topology than that of the original graph. Based on the total probability theorem (see, e.g., DeGroot [19] & Ross [20]), the probability $Pr(S)$ of event S that the system is operating, is obtained as the sum of probabilities of the afore-mentioned two mutually-exclusive events:

$$\Pr(S) = \Pr(S \cap K_1) + \Pr(S \cap \overline{K}_1) \quad (19)$$

Equation (19) can be simplified via

$\Pr(S \cap K_1) = \Pr(S|K_1)\ \Pr(K_1)$ and $\Pr(S \cap \overline{K}_1) = \Pr(S|\overline{K}_1)\ \Pr(\overline{K}_1)$, to find:

$$\Pr(S) = \Pr(S|K_1)\ \Pr(K_1) + \Pr(S|\overline{K}_1)\ \Pr(\overline{K}_1) \quad (20)$$

In equation (20) $\Pr(S|K_1)$ is the probability that the network is working given that the keystone element is in the operating state and $\Pr(S|\overline{K}_1)$ is the probability that the network is working given that the keystone branch is in the failed state, while $\Pr(K_1)$ and $\Pr(\overline{K}_1)$ are the probabilities that the keystone element is in the operating state and in the failed state, respectively. Likewise, if a pair of independent keystone branches K_1 and K_2 have been chosen (rather than just one keystone branch), the probability of network success $\Pr(S)$ (the reliability of the system) is determined as the summation of probabilities of four events, which are all mutually exclusive

$$\Pr(S) = \Pr(S|K_1K_2)\Pr(K_1)\Pr(K_2) + \Pr(S|K_1\overline{K}_2)\Pr(K_1)\Pr(\overline{K}_2) + \Pr(S|\overline{K}_1K_2)\Pr(\overline{K}_1)\Pr(K_2) + \Pr(S|\overline{K}_1\overline{K}_2)\Pr(\overline{K}_1)\Pr(\overline{K}_2)$$

The network in Figure 19, for instance, cannot be simplified by using series-parallel reduction. However, its analytical reliability is computed through the afore-mentioned decomposition technique. The system includes eight unreliable branches with availabilities $0 \leq p \leq 1$. Since a directed s-t path exists between the source s and the sink t, the throughput flow is greater than zero.

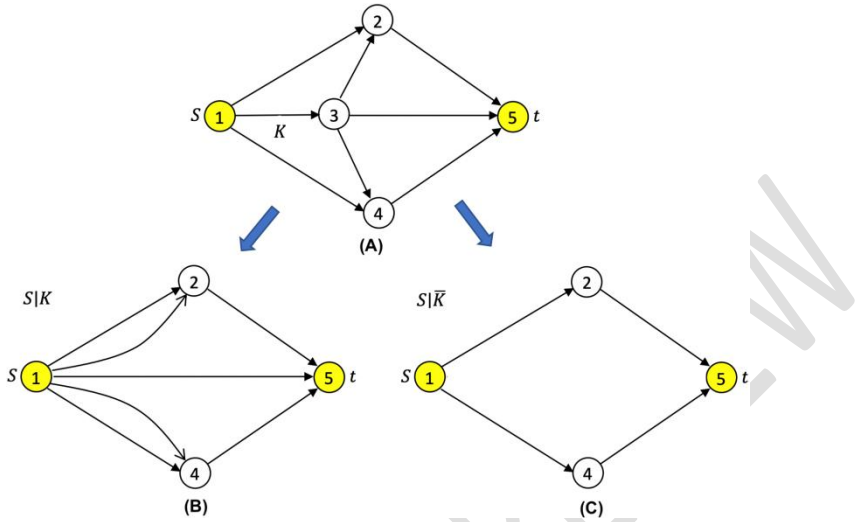


Figure 19. Decomposition technique for evaluating complex systems' reliability

If branch (1, 3) is chosen as a keystone element K, the probability of network success $Pr(S)$ is obtained based on the following equation

$$\Pr(S) = \Pr(S|K)\Pr(K) + \Pr(S|\overline{K})\Pr(\overline{K}) \quad (21)$$

The probability $\Pr(S|K)$ of the network in Figure 19B can be obtained through a series-parallel reduction. The parallel sections (1, 2) and (1, 4) dictates the success probability of two parallel components, which is equivalent to $p_d = 1 - (1-p)^2$. Therefore, the success probability for the system in Figure 19B becomes

$$\Pr(S|K) = 1 - (1-p) * (1-p_d p)^2 \quad (22)$$

Figure 19C, which shows the network's success when the keystone element fails, yields

$$\Pr(S|\overline{K}) = 1 - (1-p^2)^2 \quad (23)$$

Since $\Pr(K) = p$ and $\Pr(\overline{K}) = 1 - p$, the substitution in equation (21) yields the probability of network success $Pr(S)$ for the initial network in Figure 19A:

$$\Pr(S) = p * [1 - (1-p)(1-p_d p)^2] + (1-p) * [1 - (1-p^2)^2] \quad (24)$$

Substituting in equation (24) with $p = 0.65$ yields $p_d = 0.877$ and $\Pr(S) = 0.84$.

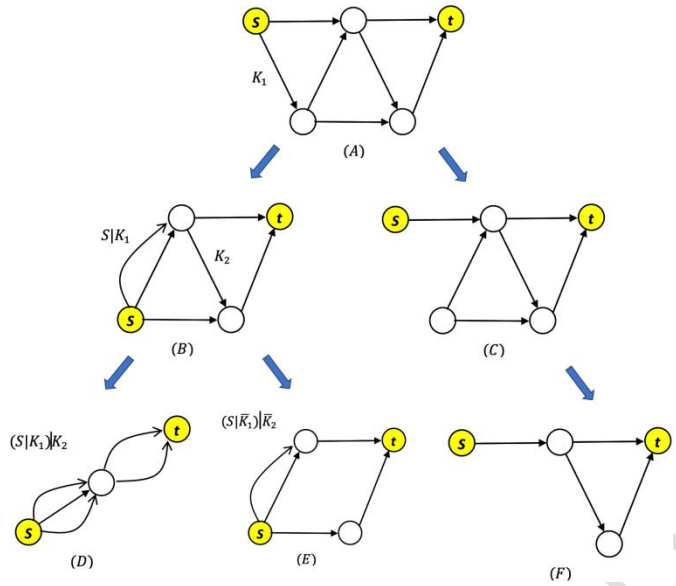


Figure 20. Decomposition-based technique for determining a topologically complex network's reliability on sequential selection of two keystone components

If the probability of success for any basic network is hard to compute, an alternative decomposition is selected by choosing another keystone branch K_2 and so on. The process is iterated until networks are obtained whose probability of success can be assessed effortlessly via series-parallel reductions. Consider the complex system S in Figure 20A which contains seven independent identically distributed branches each with reliability p. The network works if an s-t path between the source node and the sink node exists.

Although this network is not trivial, it can be simplified if a keystone component K_1 is selected. The reliability of the system $Pr(S)$ (The probability of network success) can be obtained from equation (20). Since the probability $\Pr(K_1) = p$ and $\Pr(\bar{K}) = 1 - p$, the probability of network success becomes

$$\Pr(S) = p * \Pr(S|K_1) + (1-p) * \Pr(S|\bar{K}_1) \qquad (25)$$

The probability of network success $\Pr(S|\bar{K}_1)$ shown in Figure 20C can be readily obtained because it is equivalent to the network shown in Figure 20F, whose branches have a simple series-parallel reduction. Consequently, the probability of a directed s-t path for the system shown in Figure 20C is

$$\Pr(S|\bar{K}_1) = p * [1 - (1-p)(1-p^2)] \qquad (26)$$

For $p = 0.7$, equation (26) evaluates to $\Pr(S|\bar{K}_1) = 0.59$.

The subgraph shown in Figure 20B which results from the decomposition of the sub graph shown in Figure 20A is not a trivial subgraph. However, by picking another keystone branch K_2, it can be decomposed further into two trivial subgraphs (Figures 20D and 20E) whose probability of success can be evaluated easily. Thus,

$$\Pr(S|K_1) = \Pr[(S|K_1)|K_2] * \Pr(K_2) + \Pr[(S|K_1)|\bar{K}_2] * \Pr(\bar{K}_2) \qquad (27)$$

For $p = 0.7$, we obtain

$\Pr[(S|K_1)|K_2] = [1 - (1-p)^3] * [1 - (1-p)^2] = 0.885,$

$\Pr[(S|K_1)|\bar{K}_2] = 1 - [1 - (1 - (1-p)^2)p](1-p^2) = 0.81,$

$\Pr(K_2) = p = 0.7 \; and \; \Pr(\bar{K}_2) = 1 - p = 0.3.$

Substituting these values in equation (27) yields

$$\Pr(S|K_1) = 0.885 * p + 0.81 * (1 - p) = 0.864 \quad (28)$$

Finally, the substitution in equation (25) gives

$$\Pr(S) = p * 0.862 + (1 - p) * 0.59 = 0.78$$

for the probability of success, in the initial graph of Figure 20A. This example demonstrates that the decomposition technique can be applied for the analysis of a complex system, and might be adapted for computations pertinent to capacitated networks [20, 67]. This technique, however, has substantial constraints. For instance, it is inappropriate for large systems. A keystone selection splits large system into two sub-systems, which in turn are split again, and the process continues. For a large number n of branches in the initial system, the number of sub-systems generated through the decomposition with respect to keystone elements quickly increases exponentially and becomes uncontrollable for large n. Regardless of the considered techniques, the reliability analysis of a system based on minimal cut-sets and minimal paths can be applied for determining the probability of system success. Likewise, the main problem of reliability analysis is the increase in network size with every additional minimal path and cut-set.

In conclusion, the discussed analytical technique for the analysis of capacitated networks are not appropriate for sizeable and complex systems. The system reduction rule is not appropriate for topologically complex systems while the decomposition technique is not appropriate for large systems.

7. Flow Networks Applications

Flow networks have useful applications in real-world scenarios, such as distribution of electricity and transportation networks. The same principle applies in all the applications, where the inflow at a specific node must equal the outflow at that node. The conservation constraint is analogous to Kirchhoff's current law in electric circuits. In this section, various applications of flow network problems are explained based on all kinds of problems discussed earlier. Section 7.1 highlights maximum flow problem applications in various categories like web communities, image segmentation, telecommunications, wireless networks and transportation. Additionally, Section 7.2 discusses the implementation of minimum-route problems, and their utilizations in various areas, such as very large-scale integrations, facility layout design, facility location and robotics. Besides, we will introduce in each of these two subsections a brief explanation about each application, supported by appropriate reference to pertinent research papers.

7.1. Applications of the Maximum Flow Problem

Capacitated networks play a vital role in our modern society. Their applications cut across a variety of areas such as transportation systems and manufacturing networks. Notably, they comprise important aspects of flow of basic items from suppliers to customers. They are also instrumental in the implementation of telephone networks, which enable cross-border communication. Similarly, computer network technology uses this flow network problem to implement broadband communication and internet services, which allow easy communication within local or global communities.

7.1.1. Telecommunication Wireless Networks

Till today, many people might think that in telecommunication wireless networks, there are no edges between terminal points or vertices. Other common network structures include wired networks that use wires or cables or transportation networks that have easily observable physical links between vertices. However, flows in wireless networks are mainly expressed as electromagnetic waves broadcast by the communication system through a communication channel that might be a material medium or vacuum. Generally, telecommunication networks have several applications in the wireless-communication areas such as satellites, the Internet, and cellphones, etc. Like many other network structures, the transmission of the maximum possible flow (waves) through these frameworks is a crucial task in the telecommunication systems; thus, the operation of the maximum flow methods

remains useful for telecommunication systems [56]. In general, an overview of the performance of a telecommunication is presented below.
A typical broadband connection has several users (nodes or vertices) with various needs. These needs vary based on the traffic usage in particular; therefore, prioritizing internet services based on utility rates is crucial for efficient service delivery. Therefore, offering maximum service means creating maximum flow (waves) across areas with the most signal interruptions by end users. While this may imply installing additional base stations, it also increases noise, which may affect the quality of waves. So, the providers need to constantly monitor the quality of their transmissions. Rushdi and Alsalami [52] examined two simples, albeit useful, methods used to evaluate the reliability of two-terminal multistate flow networks in communication systems. Their target is the evaluation of the probability mass function (pmf) in a wide array of cases, in which they consider flow in a capacitated network from a source node to a sink node with a multistate capacity model for the links. Each network link has a varying capacity, which is assumed to exist in a mutually exclusive sense. The reliability of the system is wholly dependent on its ability to transmit successfully at least a certain required system flow from the source (transmitter) to the sink (receiver) station. The max-flow min-cut theorem is critical in obtaining all successful states.

7.1.2. Image Segmentation

One of the most prominent research topics in the field of medical systems is that of an image segmentation system. This system delineates the parts of the body or organ images in such a way that the infected part of the body (e.g. cancer tissues) can be obviously shown. For instance, examination of kidney tumor offers an ideal application of the maximum flow problem to produce images for various tissue slices in such a way that the important part of the tumor tissue can be clearly evaluated. Any two parts of organ has special connection in biological organ examination. Therefore, the image segmentation for body organs is greatly different from other kinds of images [54]. Notably, there are similarities between the image segmentation problems and the maximum flow concept, since it replicates the theorem of the max-flow min-cut, which is the core of the maximum flow problem. The corresponding nodes and links might also be defined in medical parameter terms if the graph or image is configured as two portions. Therefore, the best cut separating the graph into two parts exhibits the minimum cut capacity or analogously, the maximum flow delivered from part 1 to part 2.

7.1.3. Extraction of Web Communities

The web is a common application of maximum flow problem, which is a directed graph or flow network. Each web page is a node while the hyperlinks are its branches in graph theory. As a normal browser of the Internet, when you are looking for a particular item via the Internet (e.g. certain items on the Amazon site or a research paper in Google Scholar, etc.), obviously, you expect to find out your item that you are looking for quickly. This accessibility depends on several crucial considerations; one of them is the connection between web pages, or so-called nodes in graph theory, to each other via hyperlinks (branches). Therefore, the interlinked web pages will significantly affect access speed by the end-users. Actually, one of the crucial factors in this subject is the significance of this web analogy. This is crucial since the Internet browser (user) normally would not be expected to go to shopping sites while he is looking for a research paper! According to this assumption, a community of the Web can be described as a group of websites. These websites have more connections or hyperlinks than websites outside the scope of the community [43]. The difference is glaring when you compare scientific research or academic related web communities with shopping communities. Notably, applying the maximum flow problem and its theorem (the max-flow min-cut theorem) on these web communities can help extract related web communities. Flake et al. [41] showed that applying the maximum flow algorithm on these communities yields a pair of nodes, such as a source node and a sink node, which are accessible from the source node across augmenting routes that satisfy the web community definition. However, if the minimum cut links remain unsaturated, in which case the flow along these links might be augmented, such links might be added to the web community. If this is the case, then the process will be repeated until all links become saturated (the flow along all edges cannot be augmented anymore). During this step, the web community will be recognized and the links that connect it to another web community will remain unsaturated.

7.1.4. Transportation

There are several cases of a transportation system where the theorem of the maximum flow can be applied like the following cases. The first case concerns the popular system structure, where maximization of flow between a pair of nodes is the top priority. Consider a supply chain structure, which is comprised of clients, retailers, wholesalers, manufactures and service providers. Notably, each entity denotes a node or a vertex in graph theory. In such a system, one of the basic and important criteria is to convey the product elements and merchandise between every two nodes (such as moving goods from distributors to clients), which should involve moving maximum possible items. Moreover, another case concerns urban planning systems such as street networks. This network is expected to apply the theorem of the maximum possible capacities, where they can utilize the possible maximum vehicles to move goods through the streets. Additionally, the case of emergency and high-priority circumstances can be considered as one in which the maximum flow becomes useful whenever high-priority or emergency scenarios exist, Such as a scenario of evacuation of people during emergencies like natural disasters. Notably, evacuation of people from affected areas is one of the most difficult problems to solve. Such a situation can have a flow-network interpretation, as the evacuees (standing for flow "commodity") must leave the danger area, denoted as the source node, towards a secured area, represented as the sink node.

7.1.5. Ecosystems

The flow network has some other applications in ecology, particularly in nutrients and energy flow among various organisms in a food web. Nevertheless, such an application differs from the normal traffic or fluid flow due to differences in the mathematical problem associated with it. Ulanowicz and Wolff [57] developed a network analysis for the ecosystem field, which applies thermodynamics and information theory concepts to examine evaluation of such a field over time. Rushdi and Alsalami [53] attempted to set the stage for a prospective interplay between ecology and reliability theory concerning the common issue of the concept of a capacitated or flow network. They treat the problem of species survivability, which pertains to the ability of a specific species to avoid local extinction by migrating from a critical habitat patch to more suitable destination habitat patches via perfect stepping stones and heterogeneous imperfect corridors. Their paper proposes various types of techniques for analyzing a capacitated ecological network for the process of migration in a meta-population landscape network that arises when paths to destination habitat patches share common corridors. Their techniques include (a) Karnaugh maps, which are crucial in providing not only the visual insight necessary to write better future software but also constitute an adequate means of verifying such software and, (b) a generalization of the max-flow min-cut theorem that is applicable through the identification of minimal cut-sets and minimal paths in the ecological flow network.
At the end of this subsection, Table 7 classifies some papers that pertain to some of the afore-mentioned applications:

Table 7. Maximum flow problem applications and their classification

Type of application	Article	The solution approach
Telecommunication Wireless Networks	Azar et al. [31]	Involves an algorithm and a linear program to model a two-part flow problem in infrastructure wireless networks with adaptive channel width. Notably, the solution is adaptable to fit requirements by particular applications.
	Caillouet et al. [35]	Utilizes the max-flow min-cut theorem to address bandwidth allocation issues in a network (wireless). This theorem uses the more common version of graph theory, particularly in the development of the Cut Covering Problem (CCP).
	Hu et al. [45]	Is mostly applicable in solving wireless mesh network issues, which include maximum

		flow and bandwidth routing problems. It also, proposes a heuristic algorithm and derives an optimization bound.
	Thulasiraman & Shen [56]	Entails design of OFDMA based hybrid hierarchical wireless networks to solve interference aware resource allocation problems.
	Rushdi & Alsalami [52]	Examines two simple (albeit useful) methods used to evaluate the reliability of two-terminal multistate flow networks in communication systems.
Image Segmentation	Freedman & Zhang [42]	Highlights automatic segmentation on multiple applications using the interactive segmentation algorithm. It seeks to implement alternative semi-automatic segmentation to enable diversity in automation.
	Song et al. [54]	Applied in prostrate and bladder imaging to produce shape and appearance information in 3-D graphs. Notably, the study aims at enhancing the quality of medical imaging and overcome simultaneous segmentation problems.
	Zeng et al. [58]	Designs and implements a novel graph-based min-cut/max-flow algorithm. It is also useful in extracting web communities based on maximum flow aspects.
Extraction of Web Communities	Asano et al. [30]	Shows various uses of maximum flow in the extraction of web communities in web-based sites.
	Horiike et al. [43]	Proposes an algorithm to extract research communities from bibliography data, and discusses a case study dealing with the web mining from Cite Seer bibliography data. The technique applied there is the maximum flow concept.
	Imafuji & Kitsuregawa [46]	Uses the maximum flow algorithm to extract a sub graph, which can be recognized as a good web community in qualitative and quantitative contexts.
Transportation	Anderson et al. [27]	Applies a case study in mapping modern roads using sensors against terrorist attacks. Moreover, it is useful for detection and prevention of attacks, which target populated areas.
	Brede & Boschetti [34]	Highlights a weighted passenger flow network for passengers in European countries. In addition, it uses the maximum flow network concept.
	Çalıskan [36]	Presents a simplex algorithm to solve constrained transportation issues.

	Rebennack et al. [51]	Depicts a specific solution for emergency transport management using the maximum contraflow concept.
Ecosystem	Rushdi & Alsalami [53]	Attempts to set the stage for a prospective interplay between ecology and reliability theory concerning the common issue of the concept of a capacitated or flow network.

7.2. Applications of the Minimum-Route Problem

7.2.1. Very-Large-Scale Integration (VLSI)

Mostly in network design, millions of very small components including transistors, resistors, and diodes are assembled on a microchip to come up with an integrated circuit on a very large scale. Very-large-scale-integrations (VLSI) have improved a lot over the years due to various technological advancements. Today, microchips that can store millions of megabytes of data exist due to integrated circuits that consist of millions of very small transistors. Thus, the very-large-scale integration system is mostly troubled with the discovery of suitable paths and layouts that enable transmission of data or nay information over the minimum routes. The VLSI problems become easier since technology can help humans generate the minimum routes required to create a grid graph consisting of millions of nodes [49]. Actually, the minimum route notion in this context indicates the minimum-wired distance between two metal elements, which are supposed to be connected in a grid graph. The Euclidean and rectilinear distances are the common means for measuring the gaps between nodes or vertices (chip's components).

7.2.2. Robotic Systems

Robots are rapidly becoming indispensable essential elements in industries and home applications. Therefore, scientist continue to explore routing solutions for these robots to integrate them in our daily operations. The need to solve the routing problem is even more urgent for autonomous robots. Notably, we do not fancy robot accidents that perform important functions aside from the need to operate efficiently without wasting time during movement [50, 55]. Notably, route planning for robots can also benefit from the minimum route problem. Similar applications that utilize the maximum route planning include computer aided manufacturing (CAM), computer aided design (CAD), and so on. Therefore, implementing routing algorithms in robotics will not be a new phenomenon.

7.2.3. Hazardous Materials Transportation (HMT)

Today, transportation and disposal of hazardous materials is a sensitive issue since it involves transporting things like chemicals, cryogens, gas cylinders, etc. Since these materials are oversensitive as they are excessively or abnormally responsive or susceptible to specific stimuli or agents, they should be transferred from a source node to a sink node through the minimum route. Therefore, in order to reduce the primary risk encountered when shipping these materials, the best k-minimum routes are factored in during routing for emergency purposes. Therefore, transporters can shift from one optimal route to the other during emergencies [38]. In fact, the route from the source node to the sink node can be the minimum route, but it passes through many residential areas. Thus, such a route is rather omitted and replaced by other nearly-optimal routes that expose less people to risks.

7.2.4. Facility Location and Facility Layout

Moving between facilities can be a hard task, especially when it involves traversing multiple routes. Therefore, facilities can be considered as network nodes, where the minimum route problem can be used to find the minimum possible routes between two locations. The case study of the manufacturers, suppliers, distributors and customers provides an ideal example when minimizing distance. For

instance, decision-makers will prioritize minimizing distance between manufactures and suppliers, then distributors in that order. Similarly, the route between machines and other essential items are crucial in the layout problem. The proximity, for example between a machine and another piece of equipment, should be easy to adjust using the minimum routing to facilitate operations and efficiency. At the end of this subsection, a few useful papers that are relevant for each of the afore-mentioned applications are summarized in Table 8.

Table 8. Sources related to the shortest path problem

Type of application	Article	The solution approach
Very-Large-Scale Integration (VLSI)	Aggarwal et al. [26]	Presents a solution for determining the shortest path in the VLSI layout problem using a multi-layer grid model.
	Brazil et al. [33]	Solves the VLSI wiring design problem by determining the shortest length or a network (Steiner-tree network).
	Peyer et al. [49]	Obtains the shortest path in VLSI problems with the help of a Dijkstra's-based algorithm.
Robotic Systems	Asano et al. [29]	Employs pseudo ε-approximate approaches to plan (Euclidean) shortest path for robots.
	Kala et al. [47]	Highlights various approaches to find best robot routes though fuzzy-based and heuristic shortest path algorithms.
	Priya et al. [50]	Introduces a solution for mapping the best (shortest) paths between a pair of mobile devices with Field-programmable Gate Arrays (FPGA)
	Sun et al. [55]	Proposes a Dijsktra-like algorithm based on the tabu restriction concept. It is mainly used for robotic path planning.
Transportation of Hazardous Materials	Androutsopoulos and Zografos [28]	Poses a simultaneous routing and scheduling problem used in minimizing routes during transit of hazardous materials.
	Boulmakoul [32]	Applied in transport of toxic materials and uses the geographical information system (GIS) to get optimal k-shortest routes.
	Carotenuto et al. [37]	Is an equitable approach towards transport of hazardous substances using a risk-based mathematical model.
	Dadkar et al. [38]	Depicts transport of toxic materials in US roads using the k-shortest path algorithm.
	Diaz-Banez et al. [39]	Describes transportation of toxic wastes based on the continuous decision-based shortest route problem
Facility Location and Facility Layout	Dong et al. [40]	Proposes a shortest path concept for a multi-stage facility layout using the auction algorithm.
	Huang et al. [44]	Provides useful information on getting the shortest route between two facilities using the

		Manhattan-distanced heuristic algorithms
	Lee et al. [48]	Gives a shortest path solution for multiple floor layouts using the Dijkstra's algorithm.

8. Conclusions

This paper provides an overview of flow network notions which details network properties, and some problem definitions, while viewing the maximum flow aspects. It also presents a review of some important algorithms used to solve the maximum-flow problem such as the Ford and Fulkerson algorithm supplemented with an additional example to explain the algorithm. The max-flow min-cut theorem is presented in detail, the concepts behind it are analyzed, and some examples and their solutions are provided to demonstrate this theorem. Moreover, the paper explains the reduction and transformation methods used in a flow network as wel as the decomposition technique which is one of the old and common techniques for analyzing capacitated systems. Some applications of flow network problems (including the maximum flow problem and the minimum routing problem) have been also discussed.

References

1. Goldberg, A. V., Tardos, É., & Tarjan, R. E. (1989). *Network flow algorithms* (No. CS-TR-216-89). PRINCETON UNIV NJ DEPT OF COMPUTER SCIENCE.
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.
3. Kleinberg, J., & Tardos, E. (2006). *Algorithm design*. Pearson Education India.
4. Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian journal of Mathematics*, *8*, 399-404.
5. Todinov, M. T. (2013). *Flow Networks: Analysis and optimization of repairable flow networks, networks with disturbed flows, static flow networks and reliability networks*. Newnes
6. Dinitz, Y. (2006). Dinitz'algorithm: The original version and Even's version. In *Theoretical computer science* (pp. 218-240). Springer, Berlin, Heidelberg.
7. Edmonds, J., & Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, *19*(2), 248-264.
8. Orlin, J. B. (2013, June). Max flows in O (nm) time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing* (pp. 765-774).
9. Wang, L. T., Chang, Y. W., & Cheng, K. T. T. (Eds.). (2009). *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann. pp. 204. ISBN 0080922007
10. Zwick, U. (1995). The smallest networks on which the Ford-Fulkerson maximum flow procedure may fail to terminate. *Theoretical computer science*, *148*(1), 165-170
11. Backman, S., & Huynh, T. (2018). Transfinite Ford–Fulkerson on a finite network. *Computability*, *7*(4), 341-347
12. Jungnickel, D., & Jungnickel, D. (2005). *Graphs, networks and algorithms*. Berlin: Springer.
13. Dantzig, G., & Fulkerson, D. R. (2003). On the max flow min cut theorem of networks. *Linear inequalities and related systems*, *38*, 225-231.
14. Akers Jr, S. B. (1960). The use of wye-delta transformations in network simplification. *Operations Research*, *8*(3), 311-323.
15. Rushdi, A. M. (1990). Star-delta transformations of bidirectional branches in probabilistic flow networks. *Microelectronics and Reliability*; *30*(3):525-535.
16. Rushdi, A. M. (1987). Capacity function-preserving star-delta transformations in flow networks. *Reliability engineering*, *19*(1), 49-58.

17. Rushdi, A. M. A., & Alsalami, O. M. A Tutorial Exposition of Various Methods for Analyzing Capacitated Networks. *Journal of Advances in Mathematics and Computer Science*. 2020;35(6):1-23.
18. Ford LR, Fulkerson DR. Flows in Networks. Princeton university press; 2015.
19. DeGroot, M. H., & Schervish, M. J. (2012). *Probability and statistics*. Pearson Education.
20. Ross, S. (2009). A first course in probability. *Upper Saddle River*, *6*.
21. Rushdi, A. M. (1984). On reliability evaluation by network decomposition. *IEEE transactions on reliability*, *33*(5), 379-384.
22. Nakazawa, H. (1976). Bayesian decomposition method for computing the reliability of an oriented network. *IEEE Transactions on Reliability*, *25*(2), 77-80.
23. Singh, B., & Ghosh, S. K. (1994). Network reliability evaluation by decomposition. *Microelectronics Reliability*, *34*(5), 925-927.
24. Misra, K. B. (Ed.). (2012). *New trends in system reliability evaluation*. Elsevier.
25. Aggarwal, K. K., Chopra, Y. C., & Bajwa, J. S. (1982). Reliability evaluation by network decomposition. *IEEE Transactions on Reliability*, *31*(4), 355-358.
26. Aggarwal, A., Kleinberg, J., & Williamson, D. P. (2000). Node-disjoint paths on the mesh and a new trade-off in VLSI layout. SIAM Journal on Computing, 29(4), 1321–1333. doi:10.1137/ S0097539796312733.
27. Anderson, L. B., Atwell, R. J., Barnett, D. S., & Bovey, R. L. (2007). Application of the maximum flow problem to sensor placement on urban road networks for homeland security. Homeland Security Affairs, 3(3), 1-15. Retrieved February 15, 2011, from http://www.hsaj.org/?article=3.3.4.
28. Androutsopoulos, K. N., & Zografos, K. G. (2010). Solving the bicriterion routing and scheduling problem for hazardous materials distribution. Transportation Research Part C, Emerging Technologies, 18(5), 713–726. doi:10.1016/j. trc.2009.12.002.
29. Asano, T., Kirkpatrick, D., & Yap, C. (2004). Pseudo approximation algorithms with applications to optimal motion planning. Discrete & Computational Geometry, 31(1), 139–171. doi:10.1007/s00454-003-2952-3.
30. Asano, Y., Nishizeki, T., Toyoda, M., & Kitsuregawa, M. (2006). Mining communities on the Web using a max-flow and a site-oriented framework. IEICE - Transactions on Information and Systems. E (Norwalk, Conn.), 89-D(10), 2606–2615.
31. Azar, Y., Mądry, A., Moscibroda, T., Panigrahi, D., & Srinivasan, A. (2011). Maximum bipartite flow in networks with adaptive channel width. Theoretical Computer Science, 412(24), 2577–2587. doi: 10.1016/j.tcs.2010.10.023.
32. Boulmakoul, A. (2006). Fuzzy graphs modelling for HazMat telegeomonitoring. European Journal of Operational Research, 175(3), 1514–1525. doi: 10.1016/j.ejor.2005.02.025.
33. Brazil, M., Thomas, D. A., & Weng, J. F. (2006). Locally minimal uniformly oriented shortest networks. Discrete Applied Mathematics, 154(18), 2545–2564. doi: 10.1016/j.dam.2005.04.021.
34. Brede, M., & Boschetti, F. (2009). Analysing weighted networks: An approach via maximum flows. In Zhou, J. (Eds.), Complex sciences (pp. 1093–1104). Berlin, Germany: Springer-Verlag. doi:10.1007/978-3-642-02466-5_109.
35. Caillouet, C., Perennes, S., & Rivano, H. (2010). Cross line and column generation for the cut cov- ering problem in wireless networks. Electronic Notes in Discrete Mathematics, 36, 255–262. doi:10.1016/j.endm.2010.05.033.
36. Çalıskan, C. (2011). A specialized network simplex algorithm for the constrained maximum flow problem. European Journal of Operational Research, 210(2), 137–147. doi:10.1016/j.ejor.2010.10.018.
37. Carotenuto, P., Giordani, S., & Ricciardelli, S. (2007). Finding minimum and equitable risk routes for hazmat shipments. Computers & Operations Research, 34(5), 1304–1327. doi:10.1016/j. cor.2005.06.003.
38. Dadkar, Y., Jones, D., & Nozick, L. (2008). Identifying geographically diverse routes for the transportation of hazardous materials. Transportation Research Part E, Logistics and Transportation Review, 44(3), 333–349. doi:10.1016/j. tre.2006.10.010.
39. Diaz-Banez, J. M., Gomez, F., & Toussaint, G. T. (2005). Computing shortest paths for transportation of hazardous materials in continuous spaces. Journal of Food Engineering, 70(3), 293–298. doi: 10.1016/j.jfoodeng.2004.05.076.
40. Dong, M., Wu, C., & Hou, F. (2009). Shortest path based simulated annealing algorithm for dynamic facility layout problem under dynamic business environment. Expert Systems with Applications, 36(8), 11221–11232. doi:10.1016/j. eswa.2009.02.091.

41. Flake, G. W., Lawrence, S., & Giles, C. L. (2000). Efficient identification of web communities. In the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 150–160). Boston, MA.
42. Freedman, D., & Zhang, T. (2005). Interactive graph cut based segmentation with shape priors. IEEE Computer Society Conference on Computer Vision and Pattern Recognition: Vol. 1 (pp. 755–762). IEEE Computer Society Conference.
43. Horiike, T., Takahashi, Y., Kuboyama, T., & Saka- moto, H. (2009). Extracting research communities by improved maximum flow algorithm. In J. D. Velasquez, et al., (Eds.), KES 2009 Proceedings of the 13th International Conference on Knowledge- Based and Intelligent Information and Engineering Systems: Part II: Vol. 5712 (pp. 472–479). Berlin, Germany: Springer-Verlag.
44. Huang, S., Batta, R., Klamroth, K., & Nagi, R. (2005). The K-connection location problem in a plane. Annals of Operations Research, 136(1), 193–209. doi:10.1007/s10479-005-2045-1.
45. Hu, C. C., Kuo, Y. L., Chiu, C. Y., & Huang, Y. M. (2010). Maximum bandwidth routing and maximum flow routing in wireless mesh networks. Telecommunication Systems, 44(1-2), 125–134. doi:10.1007/s11235-009-9217-2.
46. Imafuji, N., & Kitsuregawa, M. (2004). Finding Web communities by maximum flow algorithm using well-assigned edge capacities. The Institute of Electronics, Information and Communication Engineers. E (Norwalk, Conn.), 87-D (2), 407–415.
47. Kala, R., Shukla, A., & Tiwari, R. (2010). Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning. Artificial Intelligence Review, 33(4), 307–327. doi:10.1007/ s10462-010-9157-y.
48. Lee, K. Y., Roh, M. I., & Jeong, H. S. (2005). An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages. Computers & Operations Research, 32(4), 879–899. doi: 10.1016/j.cor.2003.09.004.
49. Peyer, S., Rautenbach, D., & Vygen, J. (2009). A generalization of Dijkstra's shortest path algorithm with applications to VLSI routing. Journal of Discrete Algorithms, 7(4), 377–390. doi:10.1016/j. jda.2007.08.003.
50. Priya, T. K., Kumar, P. R., & Sridharan, K. (2006). A hardware-efficient scheme and FPGA realization for computation of single pair shortest path for a mobile automaton. Microprocessors and Microsystems, 30(7), 413–424. doi:10.1016/j. micpro.2006.02.021.
51. Rebennack, S., Arulselvan, A., Elefteriadou, L., & Pardalos, P. M. (2010). Complexity analysis for maximum flow problems with arc reversals. Journal of Combinatorial Optimization, 19, 200–216. doi:10.1007/s10878-008-9175-8.
52. Rushdi, A. M. A., & Alsalami, O. M. Reliability evaluation of multi-state flow networks via map methods. Journal of Engineering Research and Reports. 2020;13(3):45-59.
53. Rushdi, A. M. A., & Alsalami, O. M. Reliability analysis of flow networks with an ecological perspective. Network Biology. 2021;11(1).
54. Song, Q., Liu, Y., Liu, Y., Saha, P. K., Sonka, M., & Wu, X. (2010). Graph search with appearance and shape information for 3-D prostate and bladder segmentation. In T. Jiang, et al., (Eds.), MIC- CAI, Part III, LNCS: Vol. 6363. Medical Image Computing and Computer-Assisted Intervention (pp. 172–180). Berlin, Germany: Springer-Verlag.
55. Sun, L., Liu, X., & Leng, M. (2006). An effective algorithm of shortest path planning a static environment. In K. Wang, G. Kovacs, M. Wozny, & M. Fang (Ed.), International Federation for Information Processing (IFIP): Vol. 207. Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management (pp. 257-262). Boston, MA: Springer.
56. Thulasiraman, P., & Shen, X. (2010). Interference aware resource allocation for hybrid hierarchical wireless networks. Computer Networks, 54(13), 2271–2280. doi: 10.1016/j.comnet.2010.03.012.
57. Ulanowicz, R. E., & Wolff, W. F. (1991). Ecosystem flow networks: loaded dice? Mathematical Biosciences, 103(1), 45-68.
58. Zeng, Y., Dimitris Samaras, D., Chen, W., & Peng, Q. (2008). Topology cuts: A novel min-cut/max- flow algorithm for topology preserving segmentation in N–D images. Computer Vision and Image Understanding, 112(1), 81–90. doi:10.1016/j. cviu.2008.07.008.
59. Hochbaum, D. S. (2008). The pseudoflow algorithm: A new algorithm for the maximum-flow problem. Operations research, 56(4), 992-1009.
60. Hochbaum, D. S. (1998, June). The pseudoflow algorithm and the pseudoflow-based simplex for the maximum flow problem. In International Conference on Integer Programming and Combinatorial Optimization (pp. 325-337). Springer, Berlin, Heidelberg.

61. Chandran, B. G., & Hochbaum, D. S. (2009). A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. Operations research, 57(2), 358-376.
62. Todinov, M. T. (2006). Risk-based reliability analysis and generic principles for risk reduction. Elsevier.
63. Thulasiraman, K., & Swamy, M. N. (2011). Graphs: theory and algorithms. John Wiley & Sons
64. Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1988). Network flows, Working Paper 2059-88, Sloan School of Management, MIT, USA, Available: file:///C:/Users/Dr%20ALI/Downloads/networkflows00ahuj.pdf
65. Biswas S , Sundar S , Paul B. (2007). A Review on Ford Fulkerson Graph Algorithm for Maximum Flow. International Journal of Scientific & Engineering Research . Volume 8, Issue 3. ISSN 2229-5518.
66. Rushdi, A. M. (1982).Wye-delta and delta-wye transformations: An instructive derivation, IEEE Transactions on Education 1982;E-25(4):157-159.
67. Rushdi AM. Performance indexes of a telecommunication network. IEEE transactions on reliability. 1988;37(1):57-64.