DESIGN AND ANALYSIS OF ALGORITHMS

23CSE211

FLOW NETWORKS
CASE STUDY REPORT

AMRITA
VISHWA VIDYAPEETHAM

TEAM DETAILS:

ANAND PRIYADARSHINI	CB.SC.U4CSE23306
KANISHKA S	CB.SC.U4CSE23328
SHREYA B	CB.SC.U4CSE23347
UHASHINI N	CB.SC.U4CSE23352

Table of Contents

1.1 INTRODUCTION

Flow networks are fundamental to solving problems involving directed graphs where resources move between nodes. These networks are applied in traffic routing, communication networks, project scheduling, and resource allocation. This report provides a detailed exploration of flow networks, maximum flow problems, relevant algorithms, and applications.

1.2 FLOW NETWORK

A flow network is a directed graph where each edge has a capacity that determines the maximum flow it can handle. The network consists of:

- A **source node (s)** where flow originates.
- A **sink node (t)** where flow is collected.
- **Edges with capacities** restricting the amount of flow between nodes.

Mathematically, a flow network is represented as **G(V, E, C)**, where:

- **V** is the set of vertices.
- **E** is the set of directed edges.
- **C(u, v)** is the capacity function defining the maximum flow an edge **(u, v)** $\in$ **E** can handle.

Each edge **(u, v)** carries a flow **f(u, v)** that must satisfy:

- **Capacity Constraint:** $0 \leq f(u,v) \leq C(u,v)$
- **Flow Conservation:** For every vertex except the source and sink, the total incoming flow equals the total outgoing flow

A **residual network** is derived from the original flow network and represents the available capacity for additional flow. It contains:

- **Forward edges**, representing remaining capacity **C(u, v) - f(u, v)**.
- **Backward edges**, allowing flow reduction if needed.

Properties of a Flow Network

1. **Augmenting Paths**: Paths from source to sink in the residual network where additional flow can be pushed.
2. **Residual Capacity**: The difference between the edge's capacity and its current flow.
3. **Bottlenecks**: The edges with the smallest available capacity along an augmenting path, restricting total flow.
4. **Flow Reversal**: If needed, flow can be redirected through the residual network by utilizing backward edges.

Flow networks serve as the foundation for various optimization algorithms, including the Ford-Fulkerson method, Edmonds-Karp algorithm, and Push-Relabel approach.

1.3 MAXIMUM FLOW PROBLEM

The **maximum flow problem** aims to find the maximum possible flow from the source to the sink while adhering to capacity constraints. It follows two principles:

- **Capacity Constraint**: Flow on an edge must not exceed its capacity.
- **Flow Conservation**: Except for the source and sink, the total incoming flow must equal the total outgoing flow for each node.

Solving the Maximum Flow Problem

To solve the maximum flow problem, algorithms use a residual graph and augmenting paths to push flow until no further augmentation is possible. The most commonly used approaches include:

1. **Ford-Fulkerson Algorithm**: Uses DFS to find augmenting paths and increase flow iteratively.
2. **Edmonds-Karp Algorithm**: A modified Ford-Fulkerson approach using BFS for efficiency.
3. **Push-Relabel Algorithm**: Uses preflows and height functions for faster convergence in dense networks.

1.4 CUTS IN FLOW NETWORK

A **cut** in a flow network is a partition of the vertices into two disjoint subsets **S** and **T**, such that **S** contains the source and **T** contains the sink. The **capacity of a cut** is the sum of the capacities of the edges that cross from **S** to **T**. The **minimum cut** determines the bottleneck in the network, influencing the maximum flow.

Max-Flow Min-Cut Theorem

This theorem states that the value of the maximum flow in a network equals the capacity of the minimum cut that separates the source and sink. In other words:

- The **maximum flow** that can be pushed from the source to the sink is equal to the **smallest cut capacity** that separates them.
- The **minimum cut** helps in determining the weakest points in the network where additional capacity improvements can enhance flow.

Applications of Network Cuts

- **Network Reliability**: Identifying critical points where failures could disrupt traffic.
- **Graph Partitioning**: Separating communities in social networks.
- **Security Analysis**: Identifying vulnerabilities in network communications.

1.5 MAXIMUM BIPARTITE MATCHING

A **bipartite graph** consists of two disjoint vertex sets **U** and **V**, where edges only connect vertices from different sets. **Maximum Bipartite Matching** aims to find the largest possible set of edges such that no two edges share a vertex.

Transforming Bipartite Matching into a Flow Problem

The problem can be converted into a flow network by:

1. **Adding a source node** connected to all nodes in **U**.
2. **Adding a sink node** connected to all nodes in **V**.
3. **Assigning unit capacities** to all edges (i.e., maximum capacity of 1).

By solving the maximum flow in this constructed network, we obtain the maximum bipartite matching.

Applications of Bipartite Matching

- **Job Matching**: Assigning applicants to suitable job openings.
- **University Admissions**: Matching students with courses based on availability.
- **Sports Scheduling**: Assigning referees to matches without conflicts.

2.1 FORD-FULKERSON ALGORITHM

The **Ford-Fulkerson algorithm** is an iterative approach to finding the maximum flow in a network by repeatedly identifying **augmenting paths**. It works as follows:

1. **Initialize**: Set the initial flow in all edges to zero.
2. **Find an augmenting path**: Search for a path from the source to the sink where additional flow can be pushed without exceeding edge capacities.
3. **Augment flow**: Increase the flow along this path by the smallest available capacity on the path (bottleneck value).
4. **Update residual graph**: Adjust capacities in the residual graph to reflect the new flow.
5. **Repeat** until no more augmenting paths exist.

Time Complexity

- In the worst case, Ford-Fulkerson has **O(E * max_flow)** complexity.
- If augmenting paths are chosen using **Depth First Search (DFS)**, the algorithm can be inefficient.

2.2 EDMONDS-KARP ALGORITHM

The **Edmonds-Karp algorithm** is an implementation of Ford-Fulkerson that uses **Breadth-First Search (BFS)** instead of DFS to find augmenting paths. This ensures a polynomial time complexity of $\mathbf{O(VE^2)}$, making it more efficient for practical use.

Advantages

- Guarantees polynomial runtime.
- More efficient than the naive Ford-Fulkerson implementation.

2.3 PUSH- RELABEL AlGORITHM

The **Push-Relabel algorithm** improves upon Ford-Fulkerson by maintaining a **preflow** rather than strictly adhering to flow conservation. It involves two main operations:

1. **Push operation**: If a vertex has excess flow, push the flow to an adjacent vertex if the edge has remaining capacity.
2. **Relabel operation**: If a vertex has excess flow but no valid outgoing edges, increase its height to enable more flow movement.

Time Complexity

- **$O(V^2E)$** in the general case.
- **$O(V^3)$** in dense graphs, making it efficient for large networks.

3.1 REAL LIFE APPLICATIONS:

1. Transportation Systems

- **Traffic Flow Optimization**: Helps in designing efficient road networks and managing congestion in urban areas.
- **Airline Scheduling**: Allocating flight routes to minimize delays and optimize passenger movement.
- **Public Transit Systems**: Optimizing bus and train schedules to maximize passenger flow and reduce wait times.

2. Computer Networks

- **Data Packet Routing**: Efficiently distributing data across network nodes to prevent congestion.
- **Bandwidth Allocation**: Managing limited bandwidth resources among multiple users and services.
- **Load Balancing**: Ensuring even distribution of data requests among servers to prevent overloading.

3. Supply Chain Management

- **Logistics Optimization**: Improving warehouse storage and transportation planning for goods.
- **Production Line Balancing**: Managing the workflow between different stages of production to minimize bottlenecks.
- **E-commerce Fulfillment**: Optimizing delivery routes and warehouse inventory flow.

4. Project Scheduling and Resource Allocation

- **Workforce Management**: Assigning employees to projects based on skill sets and availability.

- **Task Scheduling**: Ensuring efficient execution of dependent tasks in industrial operations.
- **Construction Planning**: Managing the flow of materials and workforce across multiple project sites.

5. Healthcare Systems

- **Hospital Resource Management**: Allocating ICU beds, doctors, and medical equipment efficiently.
- **Organ Transplant Matching**: Using flow algorithms to optimize donor-recipient pairings based on medical urgency and compatibility.
- **Ambulance Dispatch Optimization**: Ensuring the fastest response time by optimizing emergency vehicle routes.

3.2 INTERESTING APPLICATIONS OF FLOW NETWORKS

1. Sports Tournament Scheduling

Flow networks determine the feasibility of tournament schedules by modeling matches as flow problems.

2. Image Segmentation

Graph cuts in computer vision utilize flow networks to separate foreground and background in images.

3. Social Network Analysis

Max-flow methods help identify influential groups in social networks by analyzing information propagation.

4. Network Reliability Assessment

Determining the reliability of networks by analyzing the minimum cut that would disconnect a system.

4.1 Problem Statement

The increasing demand for rapid and efficient package delivery has led to the adoption of drones as a viable solution. However, optimizing drone deliveries is challenging due to several constraints:

- **Battery limitations:** Not all drones can reach every delivery location.
- **Weight capacity:** Some packages are too heavy for smaller drones.
- **Weather conditions:** Certain drones cannot operate in high wind areas.

To address these challenges, we model the problem as a **flow network** where:

- *Warehouses (sources)* supply packages.
- *Drones (intermediate nodes)* transport packages.
- *Delivery locations (sinks)* receive packages.

The objective is to **maximize the number of successful deliveries** while considering these constraints.

4.2 Objectives

- Implement multiple **maximum flow algorithms** to optimize drone deliveries.
- Compare the performance of different flow algorithms.
- Analyze how algorithmic choices affect **delivery efficiency and energy consumption**.

4.3 Implementation Approach

4.3.1 Graph Representation

The drone delivery system is represented as a directed graph:

- **Nodes:** Represent warehouses, drones, and delivery locations.
- **Edges:** Represent drone routes with capacities corresponding to weight limits, battery range, and weather constraints.
- **Capacities:** Set based on real-world constraints such as drone endurance, package weight, and atmospheric conditions.

Each warehouse connects to multiple drones, which in turn connect to feasible delivery locations. Flow capacities are assigned dynamically based on the drone's limitations.

4.3.2 Algorithms Implemented

To solve the flow optimization problem, four key algorithms were implemented:

Ford-Fulkerson Algorithm

The **Ford-Fulkerson algorithm** computes the maximum possible deliveries by repeatedly finding augmenting paths. Using a **depth-first search (DFS)** approach, it updates flows iteratively until no more paths exist.

- **Advantages:** Simple to implement, works well for small networks.
- **Disadvantages:** Can be inefficient for large networks due to unoptimized path selection.

Edmonds-Karp Algorithm

The **Edmonds-Karp algorithm**, an improvement of Ford-Fulkerson, uses **breadth-first search (BFS)** instead of DFS to find augmenting paths. This ensures shorter paths are chosen first, leading to better convergence.

- **Advantages:** Guarantees polynomial time complexity of O(VE2)O(VE^2).
- **Disadvantages:** Still slow for large-scale networks with many edges.

Capacity Scaling Algorithm

The **Capacity Scaling algorithm** enhances Ford-Fulkerson by prioritizing high-weight packages first. Instead of considering all possible flows, it processes deliveries in decreasing order of capacity constraints.

- **Advantages:** Optimized for handling larger packages more efficiently.
- **Disadvantages:** Requires additional sorting overhead, increasing preprocessing time.

Successive Shortest Path Algorithm

The **Successive Shortest Path algorithm** introduces cost-based optimization to minimize energy consumption. Using **Dijkstra's algorithm**, it finds cost-efficient paths before augmenting flow.

- **Advantages:** Optimizes energy usage, crucial for battery-limited drones.
- **Disadvantages:** Prioritizing cost over flow may slightly reduce total deliveries.

4.4 Performance Analysis

Experimental Setup

To evaluate algorithm efficiency, we tested them under two scenarios:

1. **Small-scale network:** 10 drones, 5 warehouses, 20 delivery points.
2. **Large-scale network:** 1000 drones, 50 warehouses, 500 delivery points.

Metrics analyzed include:

- **Execution time:** Measures computational efficiency.
- **Maximized deliveries:** Evaluates total successful deliveries.

- **Energy efficiency:** Analyzes trade-offs between cost-based and max-flow strategies.

Comparative Results

Algorithm	Execution Time (Large Network)	Maximized Deliveries	Energy Efficiency
Ford-Fulkerson	High	Moderate	Low
Edmonds-Karp	Medium	High	Low
Capacity Scaling	Medium	Very High	Moderate
Successive Shortest Path	Low	Moderate	Very High

4.5 Insights and Understanding

- **Edmonds-Karp** is preferable for scenarios requiring a **balance between speed and accuracy**.
- **Capacity Scaling** works best when **prioritizing high-weight packages**, ideal for logistics systems where large packages are critical.
- **Successive Shortest Path** is optimal when **energy conservation** is a priority, making it ideal for long-range drone deliveries.
- **Ford-Fulkerson**, while useful in small cases, is impractical for large-scale drone networks due to its inefficiency.

4.6 Conclusion

This study explores multiple **flow algorithms** in optimizing drone delivery networks.The optimal algorithm depends on whether delivery throughput, energy efficiency, or computational speed is prioritized. The findings can be extended to **real-world logistics**, enhancing drone-based delivery systems for improved performance and sustainability.