Technical University of Košice

Faculty of Mining, Ecology, Process Control and Geotechnologies

Institute of Control and Informatization of Production Processes

Mathematical method for identification, modelling and simulation

Ing. Ales Jandera

11. May 2024

Contents

1 XXX

2 Advanced methods in Graph Theory

Graph theory is a fundamental field of mathematics and computer science that studies the properties and applications of graphs, which consist of vertices connected by edges. It is essential for solving problems in various domains like **telecommunications** used for optimizing the routing of data to maximize throughput. **Transportation and Supply Chain Management** determining the most efficient way to route goods and services. **Utility Networks** to managing water, gas, or electricity distribution to maximize efficiency and reliability. **Project Management** using graph theory to task scheduling where tasks are dependent on the completion of preceding tasks. **Epidemiology** modeling the spread of diseases through dynamically changing human contact networks.

2.1 Network flow

Network flow is a fundamental concept in graph theory that models the flow of quantities through a network, allowing for the analysis and optimization of systems.

2.1.1 Basic Components of Network Flow

Directed Graph used for a network flow problem where each edge has a capacity and each node is either a source, a sink, or an intermediate node. Nodes represent junction points, and edges represent paths that the flow can take. Source are the origin node from where the flow starts and th sink is the destination node where the flow is intended to end. Each edge in the network has an associated capacity, which is the maximum flow that the edge can handle.

Flow is the amount of stuff (e.g., data, goods, traffic) that passes along the edges. It must satisfy **Capacity Constraint condition** that flow on an edge cannot exceed the capacity of the edge and **Conservation of Flow condition** except for the source and sink, the flow into any node must equal the flow out of that node.

2.1.2 Key Concepts

Maximum Flow is the main problem in network flow is to determine the maximum flow from the source to the sink without exceeding the capacities of the edges. Algorithms like the Ford-Fulkerson method, the Edmonds-Karp algorithm, and the Dinic's algorithm are designed to solve this problem.

Flow Value is the total amount of flow that leaves the source or enters the sink (these two numbers are equal due to the conservation of flow).

Residual Network is constructed from the original flow network by including residual capacities which indicate additional possible flow on an edge given the current flow.

Augmenting Path in the residual network is a path from the source to the sink along which additional flow can be pushed to increase the overall flow in the network.

Cut in a network is a partition of the vertices into two disjoint subsets that separate the source from the sink. The capacity of the cut is the sum of the capacities of the edges that are directed from the subset containing the source to the subset containing the sink. The max-flow min-cut theorem states that the maximum value of an source to sink flow is equal to the minimum capacity of an same cut in the network.

2.2 Graph Coloring

Graph coloring is a fundamental concept in graph theory, which involves assigning colors to elements of a graph under certain constraints. The most common form of graph coloring is vertex coloring, where colors are assigned to vertices such that no two adjacent vertices share the same color. This concept is not only pivotal in theoretical mathematics but also has practical applications in areas such as scheduling, networking, and the allocation of resources.

2.2.1 Types of Graph Coloring

The goal of **Vertex Coloring** is to color the vertices of a graph such that no two adjacent vertices share the same color. The minimum number of colors needed to achieve this is known as the graph's chromatic number.
For a proper vertex coloring using k colors in a graph G with vertex set V, the coloring function c can be described as:

$$c : V \to \{1, 2, ..., k\} \tag{1}$$

Such that for every edge $(u, v) \in E$ (the edge set of G):

$$c(u) \neq c(v) \tag{2}$$

Edge Coloring is similar to vertex coloring but applies to edges. Here, no two adjacent edges (edges sharing a common vertex) can have the same color. The minimum number of colors needed for an edge coloring of a graph is called the chromatic index $\chi'(G)$. For a graph $G = (V, E)$ with vertex set V and edge set E, and a given number of colors k, an edge coloring is a function:

$$c : E \to \{1, 2, \ldots, k\} \tag{3}$$

Such that for any two edges e_1 and e_2 that share a common vertex, the following must hold:

$$c(e_1) \neq c(e_2) \tag{4}$$

Face Coloring is relevant in planar graphs where faces (the regions bounded by edges) must be colored in such a way that no two faces sharing a boundary segment have the same color. This is directly related to the Four Color Theorem 2.2.2, which states that four colors are sufficient to color any planar graph.

2.2.2 Key Concepts and Theorems

Four Color Theorem is one of the most famous theorems in graph theory, it states that four colors are sufficient to color the vertices of any planar graph so that no two adjacent vertices share the same color. For any planar graph G can be phrased as:

$$\chi(G) \leq 4 \tag{5}$$

Brooks' Theorem provides a bound on the chromatic number of a graph. It states that any connected graph other than a complete graph or an odd cycle has chromatic number at most Δ (the maximum degree of the graph):

$$\chi(G) \leq \Delta + 1 \tag{6}$$

However, if G is neither a complete graph nor an odd cycle:

$$\chi(G) \leq \Delta. \tag{7}$$

Hall's Marriage Theorem often used in edge coloring and matching problems, it provides a condition that guarantees the existence of a perfect matching in bipartite graphs, which is directly related to the graph coloring problem 2.2.1. It states that a bipartite graph has a perfect matching if and only if for every subset S of one part of the graph, the number of neighbors of S (denoted as $N(S)$) is at least as large as S:

$$|N(S)| \geq |S| \tag{8}$$

This theorem helps in determining if it's possible to match edges (or jobs to timeslots, etc.) without overlap, effectively a coloring problem on the edges.

2.2.3 Algorithms for Graph Coloring

Greedy Coloring is a straightforward and easy-to-implement method where vertices are colored one at a time, using the smallest available color that hasn't been used by adjacent vertices. This method does not generally yield the minimum number of colors needed. The greedy algorithm attempts to color each vertex with the lowest number k where k is the smallest positive integer not used by its adjacent vertices. If $d(v)$ is the degree of vertex v, then a vertex can be colored with one of $d(v) + 1$ colors:

$$c(v) = \min\{k \in \mathbb{N} \mid k \notin \{c(u) \mid u \text{ is adjacent to } v\}\} \tag{9}$$

Backtracking Algorithm is a more exhaustive approach that can find the chromatic number of a graph but may be computationally expensive for large graphs. The chromatic number $\chi(G)$ of a graph G is the smallest number

of colors needed to color the vertices of G so that no two adjacent vertices share the same color. Formally, it's defined by:

$$\chi(G) = \min\{k \mid G \text{ is k-colorable}\} \tag{10}$$

Heuristic Algorithms solve many practical problems use heuristic or approximation algorithms to find a "good-enough" coloring. Examples include the DSATUR algorithm, which orders vertices based on their degree and saturation.

2.3 Dynamic Graphs

Dynamic graphs are an extension of traditional graph theory that deals with graphs in which the structure changes over time. These changes can include the addition or deletion of vertices and edges, as well as changes in the weights or labels of existing vertices and edges. Dynamic graphs are particularly relevant in fields where the underlying relationships between entities are not static but evolve, such as social networks, traffic networks, telecommunications, and biological networks.

2.3.1 Key Concepts

Temporal Dimension incorporate time as a fundamental component, allowing for the exploration of how graph properties evolve. This temporal dimension distinguishes dynamic graphs from static graphs, where the structure is fixed.

Look at three **Types of Changes** in dynamic graphs. **Vertex Dynamics** includes the addition or removal of nodes. Forinstance, new users joining a social network or stations being added to a transport network. **Edge Dynamics** involves the addition or removal of edges, which could represent forming or dissolving relationships in a social network orchanges in routes in a transportation system. **Attribute Changes** to the properties or weights of the existing nodes and edges, such as changing bandwidth in communication networks or fluctuating relationship strengths in social networks.
Discrete Time Models changes are modeled at specific time steps or intervals in oposite Continuous Time Models evolves the graph continuously over time, and changes can occur at any moment.

2.3.2 Analytical Challenges

The dynamic nature adds a layer of **complexity** to traditional graph problems, making them computationally harder to solve.
Keeping track of changes over time requires **efficient memory management** and computational strategies, especially for large-scale networks.
Predicting how a graph will evolve involves understanding past trends and potentially complex dependencies between changes.

2.3.3 Problems and Algorithms in Dynamic Graphs

Dynamic Connectivity determines whether two vertices are connected by a path and how this connectivity changes over time. Efficient algorithms are required to update connectivity information as the graph evolves.

Dynamic Shortest Paths finding the shortest paths in a graph that changes over time is crucial for applications like dynamic routing in transportation and communication networks. The problem of maintaining shortest paths in a dynamic graph, where edges can be added, removed, or have their weights changed, can be mathematically described as follows:

Distance Update After Edge Weight Change:

$$d(u, v) = \min(d(u, v), d(u, k) + w(k, v)), \tag{11}$$

where, $d(u, v)$ represents the shortest distance from u to v, and $w(k, v)$ represents the new weight of an edge that has been updated.

Community Detection in social and biological networks, identifying groups or communities that change over time can provide insights into the underlying structure and dynamics of the network. The modularity Q at any time t can be formulated as:

$$Q_t = \sum_{c \in C} \left[\frac{L_c}{L} - \left(\frac{d_c}{2L} \right)^2 \right], \tag{12}$$

where L_c is the number of edges within community c, d_c is the sum of degrees of the nodes in c, and L is the total number of edges in the graph.

Network Resilience analyzing how structural changes affect the stability and resilience of networks against failures or attacks.
Assessing network resilience in dynamic graphs might involve calculating the algebraic connectivity, represented by the Fiedler value (the second smallest eigenvalue of the Laplacian matrix), which can be influenced by dynamic changes:

$$L = D - A, \tag{13}$$

where D is the degree matrix and A is the adjacency matrix of the graph.
Algebraic Connectivity (Fiedler Value):

$$\lambda_2 = \min_{x \neq 0, x \perp \mathbf{1}} \frac{x^T L x}{x^T x} \tag{14}$$

The smaller λ_2 is, the less connected the graph is, indicating lower resilience.