MA2827
Foundations of discrete optimisation

Professor: Anton Osokin

Language of instruction: English – **Number of hours**: 36 – **ECTS**: 3

Prerequisites: Basic knowledge of graph theory will be helpful, but not required since the course will be self-contained. Programming skills (in any programming language of your choice) will be necessary for completing the lab exercises.

Period: S8 Elective 11 March to June IN28IE4, SEP8IE4

Course Objectives

Discrete optimization is concerned with the subset of optimization problems where some or all of the variables are confined to take a value from a discrete set. Examples include several important problems in various fields of applied mathematics and computer science, such as

- Finding the best (shortest, cheapest, most scenic) route from one place to another.
- Connecting cities using a road network that minimizes the cost.
- Selecting a subset of projects, each requiring a subset of available resources, to maximize profit.
- Finding the best assignment of students to universities.

In this course, we will study the fundamental concepts of discrete optimization such as greedy algorithms, dynamic programming and min-max relationships. Each concept will be illustrated using well-known problems such as shortest paths, minimum spanning tree, min-cut, max-flow and bipartite matching. We will also identify which problems are *easy* and which problems are *hard*, and briefly discuss how to obtain an approximate solution to hard problems.

On completion of the course, students should be able to

- Identify which optimisation problems are easy and which are hard.

- Design efficient algorithms for easy problems.

- Have a basic understanding of how hard problems are approximately solved in practice.

Course Contents

Briefly, the following topics will be covered in the course.

- Shortest Paths

- Minimum Spanning Trees, Disjoint Paths

- Minimum Cut and Maximum Flow

- Bipartite Matching

- P, NP, and NP-Complete Problems

- Convex Relaxations

A more detailed description of the topics follows.

1. Shortest Path

Summary. Given a directed graph (vertices, arcs and arc lengths), the goal is to find the minimum length or shortest path from one vertex to another. This problem can be solved efficiently when the graph does not contain a negative length directed circuit.

Preliminaries:

- Graph preliminaries
- Complexity preliminaries

Shortest Path:

- Breadth-first search
- Djikstra's algorithm
- Bellman-Ford algorithm
- Floyd-Warshall algorithm

2. Minimum Spanning Tree, Disjoint Paths

Summary. Given an undirected graph (vertices, edges and edge lengths), a spanning tree is a subgraph that consists of all vertices and whose edges form a tree. The minimum spanning tree problem requires us to find the spanning tree with the minimum length. The problem can be solved efficiently for arbitrary (real-valued) edge lengths. Given a directed graph, the disjoint paths problem requires us to find the maximum number of arc disjoint paths between two vertices, where two paths are arc disjoint if they do not contain a common arc. The problem is equivalent to finding the maximum number of vertex disjoint paths between two subset of vertices, and the maximum number of internal vertex disjoint paths between two vertices. All the disjoint path problems can be solved efficiently.

Minimum Spanning Tree:

- Prim's algorithm
- Kruskal's algorithm

Disjoint Path:

- Menger's theorem(s)
- Algorithm for finding maximum number of arc-disjoint paths

3. Minimum Cut, Maximum Flow

Summary. Flow is a function on the arcs of a directed graph such that its value is non-negative, less than the length (or capacity) of the arc, and for all vertices other than a source vertex and a sink vertex, the excess flow is 0. The maximum flow problem requires us to find the flow with the maximum value. A cut is a set of arcs from one subset of vertices that contain the source to another subset of vertices that contain the sink. The minimum cut problem requires us to identify the subsets of vertices which minimize the capacity of the cut. The maximum flow problem and the minimum cut problem are equivalent to each other, and both can be solved efficiently for directed graphs with non-negative arc capacities.

- Max-Flow Min-Cut theorem
- Ford-Fulkerson algorithm
- Dinit's algorithm

4. Minimum Cost Flows and Circulations

Summary. Given a directed graph, along with a function for each arc that measures the cost of passing a unit flow through the arc, the minimum cost flow problem requires us to find the flow with the lowest cost. Circulations are a variant of flow with no source or sink. Given a cost function and a demand function, the minimum cost circulation problem requires us to find the circulation with the lowest cost such that the flow in each arc is greater than or equal to the demand. Both the problems can solved efficiently.

- Reducing shortest path and max-flow to minimum cost flow
- Reducing minimum cost flow to minimum cost circulation
- Algorithm for computing minimum cost circulation

5. Bipartite Matching

Summary. Given an undirected graph, a matching is a set of disjoint edges. The weight of a matching is the sum of the lengths or weights of the edges contained in the edge. The matching problem requires us to find the maximum weight matching of a given graph. The problem can be solved efficiently when the given graph is bipartite.

- Konig's theorem
- Algorithm for cardinality bipartite matching
- Hungarian algorithm for weighted bipartite matching

6. P, NP, and NP-Complete Problems

Summary. We define two types of decision problems (problems with a "yes" or "no" answer): those with a polynomial-time algorithm (P), and those with a polynomial-time checkable certificate (NP). Any NP problem can be reduced to the satisfiability (SAT) problem, which makes SAT NP-complete. The NP-completeness of several other problems is proved by reducing SAT to those problems. We also define NP-hard problems, which may not have polynomial-time checkable certificates, and may not even be decision problems.

- ✧ Random access machine
- ✧ Cook's theorem
- ✧ Reductions

7. Convex Relaxations

Summary. Convex relaxations offer an elegant way to obtain approximate solutions for difficult (NP-hard) problems. The idea is to formulate a problem as an integer program and then relax the integer constraints to obtain a convex program, which can be solved efficiently. We show some well-known examples of accurate convex relaxations for the multiway cut and the max-cut problem.

- ✧ Convex programming preliminaries
- ✧ Multiway cut
- ✧ Maximum cut

Course Organization

The course consists of seven lectures (3 hours each), which will cover the aforementioned topics.

In addition, there will be five lab sessions (3 hours each) where the students will write the code for the following problems:

- ✧ 1. Computing Shortest Paths on the Paris Metro
- ✧ 2-3. Interactive Binary Image Segmentation via Minimum Cut
- ✧ 4-5. Automatic Photo-Stitching via Bipartite Matching

Teaching Material and Textbooks

Slides will be provided on the instructor's webpage (http://cvn.ecp.fr/personnel/pawan/teaching.html). Students are encouraged to look at the slides from previous offerings of the course to get an idea of the level of detail.

In addition, the following course notes by Alex Schrijver will also be helpful (http://homepages.cwi.nl/~lex/files/dict.pdf).

Previous exam papers and other example problems will be provided on the instructor's webpage.

Evaluation

Students will be evaluated based on their lab work (50%) and a final exam that will last 3 hours (50%). The use of printed materials such as slides, notes and textbooks will be permitted during the exam. No electronic devices will be allowed. Bonus points will be available for class participation and would likely play a very important role in your final grade