Course content for MT2630, Graphs and Optimisation

Prerequisites:
MT1810 and MT1820

Aims:
This course provides an introduction to graph theory and linear programming. Many networks such as railroad networks, social networks, electrical networks and the worldwide-web can be modelled by graphs. In this course we consider graphs and directed graphs and establish some of their basic properties. We then consider flows in networks and see how flows are related to a more general optimisation concept called linear programming. The course then introduces other linear programming problems and how to solve them algorithmically. To every linear program there exists a dual program and we investigate the relationship between the solutions of these two problems.
Key aims of the course include:

- to provide an introduction to graphs and linear programming;
- to develop the skill to formulate and solve linear programs and their duals.

Learning outcomes:
On completion of the course, students should:

- know the basic notions of graph theory;
- know basic examples of graph classes, like paths, cycles and trees;
- be able to formulate problems as matching and flow problems;
- be able to formulate linear programming problems;
- be able to solve linear programming problems with the simplex algorithm;
- be able to formulate the dual of a linear programming problem;
- know the strong duality theorem.

Course content:
Graphs: Definition of a graph and directed graph; basic examples of graphs such as paths, cycles, trees; basic properties of graphs such as graph isomorphism, the handshaking lemma, degree sequences, characterisations of trees, connectivity and components; matchings and flows in graphs; Hall's theorem.
Optimisation: Examples of formulation of problems as linear programs; flow problem as an LP problem. Unbounded and infeasible LPs; feasible and basic feasible solutions; the simplex algorithm. LP duality; the weak and strong duality theorem; dual of the dual is the primal; complementary slackness and testing for optimality. The branch-and-bound method for integer programming.