Course Outlines
Year 1
ALGORITHMS FOR DECISION SUPPORT

Course goals
A student that successfully finishes the course will:

know central concepts and algorithms from combinatorial optimization including linear and integer linear programming
be able to model problems from applications as an (integer) linear program
know central concepts and algorithms from online algorithms, and is able to analyse the behavior of an online algorithm
have basic knowledge about online algorithms and competitive analysis (and further with machine learning advice)
have knowledge of important concepts and proof techniques from algorithmic complexity, including undecidability, NP-completeness, etc.
be able to understand and design NP-completeness proofs
Assessment
There is a project done in groups (40% of the final mark) and two exams, both an approximately half of the materials (30% each).
You must have at least a 4.5 for the project and for the average of the exams to pass the course, and at least a 5.5 for the overall average.

If the project was graded less than 4.5, and you have an average of at least 4 for the exams and the project, you can do a new project.
If the average of your exams was graded less than 4.5, and you have an average of at least 4 for the exam and the project, you can do the re-exam.
If your average was at least 4 and less than 5.5, you can do the re-exam.

Details on grading for the 2nd chance exam or project can be found on the Blackboard site of the course.

Prerequisites
Basic logic and reasoning.

Content
The purpose of this course is to teach topics that are:

important for the working area of algorithms (in practice and theory)
prerequisites for other courses in the COSC program
not encountered by all students in the bachelor.
It therefore contains a broad range of topics.

In many real-life decision problems in e.g. (public) transportation, logistics, energy networks, healthcare, computer networks and education we want to select a very good solution from a large set of possible solutions.
In the course you learn how to model such problems and how to solve them by optimization algorithms. We focus on discrete models.
You shall learn about computational complexity and about meaning of exact optimization algorithms, heuristics and what-if analysis.
We study well-known algorithms from combinatorial optimization, in the scope of linear programming, online algorithms, randomized algorithms, etcetera.

Course form
Lectures, self-study, exercises, assignments.

Literature
The material of the course consists of course notes and slides.

The following books are not mandatory but interesting for further reading:
Laurence A. Wolsey, "Integer Programming", Wiley-Interscience publication, 1998, ISBN 0-471-28366-5.
M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, New York, 1979, ISBN 0-7167-1044-7..

CONCEPTS OF PROGRAMMING LANGUAGE DESIGN

Course goals

After completing this course, students will have an understanding of programming language paradigms such as object oriented, imperative and functional programming.
They will understand fundamental concepts, such as static and dynamic semantics of programming languages, type safety, static and dynamic typing, subtyping, inheritance, and higher order functions, as well as manual versus automatic memory management.
They will understand how the semantic of a language can be specified formally, to prove properties of the language in general or individual programs (such as correctness).
They will also understand how these features can be implemented, and how they can affect run time, safety and usability of a language.

Assessment

Students will need to

1. complete a project component, which can be done by either
completing a small research project (in a group) and present this project in a report (in a group), or by
completing a programming assignment component, which may include a written report.
2. take a final exam

The final mark is the average of the exam and project/assignment marks. Both these marks must be at least a 4.

To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Content

There exists a multitude of programming languages, and the top ten list of most popular languages keeps changing continuously.
However, the set of basic building blocks on which these languages are based changes much more slowly.
In this course, we focus on exactly these building blocks and concepts, and look at how they are implemented in concrete programming languages.
We use predicate logic and natural deduction to specify and analyze the static and dynamic semantics of languages.

Course form

Lectures, project, presentations.

Study material

There is no book, but there are lecture notes for most of the topics.
In addition, articles, slides, and background reading will be provided during the course.

For the assignments, we will use the functional programming language Haskell.
For the projects, students can choose the most suitable language.

SCIENTIFIC METHODS FOR COMPUTING SCIENCE

Course goals

Thee learning objectives of this course are

to learn about scientific research in general

to learn about the most common research methods used in Computing Science

to increase practice with working in teams

to increase practice with scientific reporting

to learn to find and judge scientific publications

and also

to learn more about the Computing Science master program

to get to know your fellow COSC students

Assessment

Assignments (80% of the final mark), exam (20%).

To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

ADVANCED ALGORITHMS

After completing this course, the student will have:
knowledge of important combinatorial algorithms
knowledge of important algorithmic techniques and concepts
the ability to model problems from applications mathematically (e.g., as a graph problem)
the ability to apply algorithmic techniques, to solve these problems
the ability to design combinatorial algorithms
the ability to prove correctness of an algorithm
the ability to analyze the running time of complex algorithms

Assessment

There are a number (approximately 6) exercise sets, and two exams.
The exercise sets count for 30 percent of the final grade, and the two exams each for 35 percent. The exercises sets will be peer-reviewed.

In order to pass the course, you need:
an average grade of at least 5.5 (computation of the average is explained at the course website).
at least an average grade of 6.0 on the exercises.
an average of at least 5.0 for the exams.
To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Prerequisites

We expect basic mathematic and algorithmic skills and knowledge.
Besides the basics, we also expect the students to be familiar with basic algorithmic paradigms and the theory of NP-completeness.
This is taught in courses such as INFOAL Algorithmiek (bachelor level 3), or INFOMADS Algorithms for decision support (master).
If you have not taken these courses (or any other equivalent course) it may be hard to keep up with the pace of the course.
Please contact the course coordinator if you’re in doubt.

Content

In this course a number of advanced techniques for efficient algorithm design are studied, often at the hand of combinatorial problems.
Networks (graphs) are often used as a basic theoretical model, both to model practical application, and as a theoretical object to study and design algorithms for.
Also graphs are a main theme, we also consider classical combinatorial problems in a broader sense, including packing, covering, knapsack, subset sum, and some scheduling problems.

This course has two parts.
The first part is on important fundamental algorithms with both practical and theoretical implications, namely (stable) matchings, advanced algorithms for maximum flow/minimum cut, minimum cost flows, Euler tours, planar graphs, and the probabilistic method).

The second part focuses on getting familiar with a broader range of fields of algorithmic study involving fixed parameter tractability, exact exponential-time algorithms, probabilistic/randomized algorithms, treewidth, and approximation algorithms.

Course form
Lectures.

Literature
Most literature will be handed out during the course.
We refer to a series of book for further reading, but these are not required.

GEOMETRIC ALGORITHMS

Course goals

In many areas of computer science -- robotics, computer graphics and virtual reality, and geographic information systems are some examples -- it is necessary to store, analyze, and create or manipulate spatial data.
This course deals with the algorithmic aspects of these tasks: we study the design and analysis of geometric algorithms and data structures.

Assessment

Homework exams that will be distributed thrice (5%, 25% and 25% of the final mark), and in the exam week there will be a final exam (45%).

Each item has to be scored with at least a 5 to pass the course, and the weighted average, rounded, must be at least a 6.
In addition, there will be a bonus assignment worth 0.5pts.
The maximum achievable grade is a 10.
The final exam is "closed book".

If you do not make or fail (<=4) two or more of the homework exams, you fail the course.
If you fail one homework exam, you get a new, third homework exam based on later chapters of the book to replace the failed homework.

To qualify for a repair of the final result the mark needs to be at least a 4.

Content

Computer graphics, robot motion planning, computer games, simulations, geographic information systems, and CAD/CAM systems all make use of geometric algorithms to perform various tasks.
This course on geometric algorithms takes a fundamental viewpoint and discusses the design and analysis of geometric algorithms.
We will study various algorithmic techniques and geometric concepts that are useful to solve geometric problems efficiently.
These include plane sweep, randomized incremental construction, and multi-level data structures; geometric concepts include Voronoi diagrams and Delaunay triangulations, arrangements, and duality.
We will apply these techniques to solve a variety of problems: convex-hull computation, line-segment intersection, polygon triangulation, low-dimensional linear programming, range searching, and point location are some examples.
Each problem we study is motivated by a practical problem from one of the application areas.

Course form

Lectures.

Literature

M. de Berg, O. Cheong, M. van Kreveld, M. Overmars."Computational Geometry: Algorithms and Applications" (3rd edition). Springer-Verlag, Heidelberg, 2008. ISBN 978-3-540-77973-5.

NETWORK SCIENCE

Course goals

After completing this course, the student will have:
knowledge of classic network analytic measures
knowledge of important (random) graph models
knowledge of important dynamics of networks
knowledge of important processes on networks
knowledge of important algorithmic challenges and solutions for the analysis of very large networks
the ability to analyze properties of random graphs
the ability to analyze properties of dynamics and processes on networks
the ability to select and interpret suitable network measures and algorithms for various (real-world) problems
the ability to survey literature of an advanced topic in network algorithms
the ability to experimentally study an advanced topic in network algorithms
the ability to provide/use feedback to/from peers

Assessment

The course is graded as follows:
exam (40% of the final mark)
term paper (50%)
peer reviews (10%).
You can only pass the course if your marks for the exam and the term paper are both strictly larger than 4.
To qualify for a repair, your mark for either the exam or the term paper has to be strictly larger than 4.
The repair consists of a re-exam if your exam mark was at most 4; an update to your term paper if your term paper mark was at most 4; or your choice of one of these two options otherwise.

Prerequisites

The course assumes that you have basic skills in algorithms and mathematics: familiarity with basic graph algorithms (shortest paths, flows), such as offered in INFOAL Algoritmiek, and basic understanding of NP-completeness, such as offered in INFOAL or INFOMADS Algorithms for Decision Support.
Having taken INFOAN Algorithms and Networks is helpful, but not required. During the class, we also work with basic probabilities and some integrals.

Content

Network science is an exciting new field that studies large and complex networks, such as social, biological, and computer networks.
We will study, formalize, and quantify (social) network phenomena such as "the small world", "rich get richer", "birds of a feather flock together", "strength of weak ties" and others.
The class will address topics from network structure and growth to the spread of epidemics.

We also consider the diverse algorithmic techniques and mathematical models that are used to analyze such large networks, and give an in-depth description of the theoretical results that underlie them.

Some topics are centralities, pagerank algorithms, assortativity, random graphs, giant components, power laws, diffusion, percolation, spreading phenomena, community detection, basic algorithms for network science, lower bounds and advanced algorithms for polynomial-time problems, streaming algorithms, temporal networks, graph databases, and graph partitioning algorithms.

Course form
Lectures, tutorials, term paper, peer feedback.

A major component of the class is for students to perform their own experimental study on algorithms for the community detection problem.
The results of this study will be presented in the term paper. This term paper will be reviewed by your student peers.

Literature
Recommended:

A. Barabasi, "Network Science", for free online
D. Easley, J. Kleinberg, “ Networks, Crowds, and Markets: Reasoning About a Highly Connected World”, for free online
M.E.J. Newman, "Networks", 2nd edition (2018).
The class is based on parts of all three books. The Newman book is the most comprehensive reference to the material of the class.

OPTIMIZATION FOR SUSTAINABILITY

Course goals

In this course students learn to:

understand discrete-event simulation models and statistical analysis methods for these models
create a discrete-event simulation model for a given system
perform a scientific sound simulation study including statistical analysis
understand optimization models for sustainable mobility and energy systems
understand applications of optimization algorithms and simulation to sustainable mobility and energy systems
assess and present scientific papers on optimization algorithms and simulation to sustainable mobility and energy systems
identify and describe possibilities for applications of optimization algorithms and simulation to sustainable mobility and energy systems

Assessment

The grading consists of the following parts:

hand-in exercises: 1) simulation model (5% of t he final mark), 2) input analysis (5%), 3) optimization challenge energy and mobility (10%)
simulation assignment (40%)
seminar (40%)

To obtain a pass grade

you have handed in all the assignments and the preparation documents for the seminar
you have given a seminar presentation
the grade for the optimization challenge exercise has to be at least 5.0
you attended the milestone and feedback meeting of the simulation meeting in person. It is not sufficient if the other member of your group attended.
you have attended the seminar sessions (exceptions for valid reasons are possible, contact teacher before the session)

Minimum effort to qualify for 2nd chance exam:

your final grade is at least 4
you delivered the hand-in exercises on time.
you received a pass for the milestone of the simulation assignment

You can take an additional exam for at most one out of hand-in exercises, simulation assignment, seminar. Which one is decided in discussion with the teacher.

Prerequisites

This course in primarily meant for students in the master Computing Science.
The focus is on operations research and computer science.
The required knowledge is mathematics and statistics at the level of the bachelor computer science, algorithms (e.g. Algorithms from the bachelor) and programming at the level of Imperative Programming.

Content

The current energy transition leads to many changes in the energy as well as the mobility system. In this course, we study algorithmic techniques to optimize the performance of future energy systems and discuss topics from sustainable public transportation.

We discuss Mixed Integer Linear Programming (MIP) formulations, branch-and-cut, and simulation.
Students learn how to apply these techniques to different optimization problem related to energy systems, such as network design, unit commitment, load flow, demand response, and storage optimization.
Moreover, we discuss question around electric buses, environmental friendly bus driving, and multi-modal route planning.
The topics in this course are related to recent research in the Algorithms and Complexity group.
In this course you will learn algorithms and methods that are also used by companies to solve real-world problems.

This course will not deal with political or policy issues in the domain of `sustainability', but is devoted entirely to (a selection of) the computational models and optimizing algorithms that are developing in the field.

Course form
Lectures, seminar: discussion and presentation of papers

Literature
Material will be made available through MS Teams.

SCHEDULING AND TIMETABLING

Course goals

After completing this course, you are ready to solve real world problems in the field of Logistics. Moreover, you will train your analytic skills and common sense.

Assessment

There will be an oral exam, which counts for 50% of the final mark. On this you must at least score a 5.0 to pass.
Furthermore, there are take-home exercises, which count for 25%.
Finally, each student has to present a paper, which counts for 25% as well.

To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Prequisites

We advise students to choose this course in their second year of the master, when they have followed INFOMADS Algorithms for decision support.
If you have followed the bachelor courses INFOOPT Optimalisering en Complexiteit, INFOAL Algoritmiek, or INFOB3DW Discrete Wiskunde, then there is no need to follow the course INFOMADS before INFOSTT.

Content

The course consists of two major subjects: Timetabling and Scheduling (which are treated in this order).

In the first part, we discuss problems that occur in practice, like determining the optimal time-table for the Dutch Railways, the planning of buses and drivers in Rotterdam, and the construction of a timetable for school. The focus here is on finding good solutions that can be used in practice. Examples of techniques we use here are ILP models, min cost max flow, and local search.
In the second part we give an introduction into scheduling. Here we discuss questions like: what is the best order to execute a set of tasks in case you want to minimize some given objective, like total weighted completion time? The focus in this part is on theory.

Course form

Lectures.

Literature

Handouts.

ADVANCED ALGORITHMS

Course goals

After completing this course, the student will have:
knowledge of important combinatorial algorithms
knowledge of important algorithmic techniques and concepts
the ability to model problems from applications mathematically (e.g., as a graph problem)
the ability to apply algorithmic techniques, to solve these problems
the ability to design combinatorial algorithms
the ability to prove correctness of an algorithm
the ability to analyze the running time of complex algorithms

Assessment

There are a number (approximately 6) exercise sets, and two exams.
The exercise sets count for 30 percent of the final grade, and the two exams each for 35 percent. The exercises sets will be peer-reviewed.

In order to pass the course, you need:
an average grade of at least 5.5 (computation of the average is explained at the course website).
at least an average grade of 6.0 on the exercises.
an average of at least 5.0 for the exams.
To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Prerequisites

We expect basic mathematic and algorithmic skills and knowledge.
Besides the basics, we also expect the students to be familiar with basic algorithmic paradigms and the theory of NP-completeness.
This is taught in courses such as INFOAL Algorithmiek (bachelor level 3), or INFOMADS Algorithms for decision support (master).
If you have not taken these courses (or any other equivalent course) it may be hard to keep up with the pace of the course.
Please contact the course coordinator if you’re in doubt.

Content

In this course a number of advanced techniques for efficient algorithm design are studied, often at the hand of combinatorial problems.
Networks (graphs) are often used as a basic theoretical model, both to model practical application, and as a theoretical object to study and design algorithms for.
Also graphs are a main theme, we also consider classical combinatorial problems in a broader sense, including packing, covering, knapsack, subset sum, and some scheduling problems.

This course has two parts.
The first part is on important fundamental algorithms with both practical and theoretical implications, namely (stable) matchings, advanced algorithms for maximum flow/minimum cut, minimum cost flows, Euler tours, planar graphs, and the probabilistic method).

The second part focuses on getting familiar with a broader range of fields of algorithmic study involving fixed parameter tractability, exact exponential-time algorithms, probabilistic/randomized algorithms, treewidth, and approximation algorithms.

Course form
Lectures.

Literature
Most literature will be handed out during the course.
We refer to a series of book for further reading, but these are not required.

DOMAIN-SPECIFIC LANGUAGES

Course goals

After successfully completing the course, you will be able to:

discuss programming language techniques to implement domain-specific languages and their tradeoffs.
understand how programming language techniques can be used to improve the security of software systems.
design and implement domain-specific languages, analyses, and transformations to address specific problems, such as security bugs.
understand and explain fundamental aspects of security attacks (e.g., undefined behavior, memory vulnerabilities, sandbox breakout, information leaks, side channels).
understand and explain basic mathematical concepts (e.g., lattices, fixed points, traces) underlying domain-specific languages, analyses, and properties.
develop formal models that describe the behavior and the security requirements of a system.
develop formal specifications of domain-specific languages and prove that they achieve desired security properties using program semantics.

Assessment

Written exam; lab assignments.

A repair test requires at least a 4 for the original test.

Prerequisites

Note that basic knowledge of and experience with the Haskell programming language is important to do the practical assignments; this is a prerequisite to follow the course.
Note, though, also that you are not expected to have done the course on Advanced Functional Programming.

Content

Building secure systems is notoriously hard. Despite every security patch, attackers always seem to find new exploits.
A huge part of the problem is that developers do not have the right tools to write secure software. They build complex software systems that handle sensitive data using programming languages that do not account for security or privacy.
Thus, developers can too easily introduce bugs that attackers can exploit as security vulnerabilities to breach their systems.
Domain-specific languages can help developers address the security and privacy problems of their systems through security-oriented abstractions that make it easy to detect and eliminate security bugs.

This course studies techniques to design and implement domain-specific languages and analyses, with a focus on software security.
In the first part of the course, we will learn basic program analysis techniques and apply them to automatically prevent sandbox breakouts and information leaks in cryptographic code.

We will then cover lightweight techniques to embed domain-specific analyses into general-purpose languages. We will apply these techniques to develop information-flow tracking languages that can enforce data confidentiality by construction in untrusted third-party code.

The course combines theoretical foundations and hands-on experience.
We will learn how to design programming languages and analyses using formal semantics and how to specify and establish their security guarantees.
The practical assignments will provide experience implementing static analyzers and domain-specific languages.
Course form
Combined lectures, exercises, lab sessions.

Literature
Slides and the following book: Nielson, Flemming, Nielson, Hanne R., Hankin, Chris, "Principles of Program Analysis", Corr. 2nd printing, ISBN: 3-540-65410-0.

Other material may be made available throughout the course.

PROBABILISTIC REASONING

Course goals

Upon completing this course, the student

1. recognizes and understands the strengths and weaknesses of probabilistic graphical models (PGMs) in general and Bayesian networks in particular;
2. understands the relation between probabilistic independence and the graphical representations thereof, and is able to draw conclusions from this relation;
3. understands and is able to apply probabilistic inference in Bayesian networks and through probabilistic programs;
4. has knowledge and understanding of methods for constructing probabilistic models for actual applications;
5. understands and is able to apply techniques for evaluating the robustness and quality of probabilistic models.

Assessment

The course is assessed through a set of practical homework assignments (20% of the final mark) and a written test (80%).

Unless you have passed the course, you qualify for a second attempt at the written test if both final grade and assignment grade are at least 4.

The practical assignments can not be re-taken.

Prerequisites

This course is primarily aimed at students of the Computing Science master program (COSC).

To be able to fulfill the learning goals of this course, it is necessary to develop a solid understanding of the mathematics underlying PGMs.

This means that prior to taking the course you should have sufficient skills to understand, apply and manipulate mathematical formulas, and have basic knowledge of algorithms, graph theory and probability theory (at the level of INFODS Datastructuren).

In addition, it is assumed that you are capable of abstracting away from given examples, applying the knowledge and techniques learned to contexts other than those discussed in class.

Basic familiarity with R or Python can be an advantage, but is not required.

Content

Probabilistic models can be used for reasoning and decision support under uncertainty:

Which exercises are most suitable to improve Alex’s calculus skills?

How long after infection will we detect classical swine fever on this farm?

What is the risk of Mr. Johnson developing a coronary heart disease?

Should Mrs. Peterson be given the loan she requested?

Will a "study advisor support tool" advise you to take this course?

In complex domains, people have to make judgments and decisions based on uncertain, and often even conflicting, information; a difficult task, even for experts in the domain.

To support these complex decisions, knowledge-based systems should be able to cope with this type of information.

For this reason, models for representing uncertainty and algorithms for manipulating uncertain information are important research subjects within the field of Artificial Intelligence.
Probability theory is one of the oldest theories dealing with the concept of uncertainty and therefore plays an important role in many decision support systems.

In this course, we will consider probabilistic models for representing and reasoning under uncertainty.
More specifically, we will focus on probabilistic graphical models such as Bayesian networks, their underlying theory, and discuss issues and methods related to the construction of such networks for real-life applications.
In addition, we consider methods for probabilistic inference, including the role of Probabilistic Programming.

More information about the course can be found on the course website:
https://ics.uu.nl/docs/vakken/prob/

Course form
Lectures, self-assessment exercises.

Literature

syllabus 'Probabilistic Reasoning with Bayesian networks'
course slides
selected articles
All are mandatory and will be made available through the course website or Blackboard.

PROGRAM SEMANTICS AND VERIFICATION

Course goals

Learning goals: to become familiar with, and acquire insight on the underlying concepts of:

program semantics: operational, denotational, axiomatic.
formalisms to express programs' correctness: Hoare-style, LTL, CTL, CSP.
automated verification techniques: predicate transformation/symbolic execution, automated theorem proving, model checking (LTL,CTL, symbolic), refinement checking.
Additionally, to acquire hands-on experience with :
implementing and optimizing a verification technique.
using a verification tool to model a problem and conduct a verification of its solution.

Assessment

In principle assignments (20% of the final mark), projects (25%), exams (55%).

This can be changed depending on the composition of the projects, but in any case the exam form and grading will be announced and fixed at the start of the course.

To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Prerequisites

Background in predicate logic, experience with a functional programming language, e.g. Haskell or ML.

Content

Most modern software is quite complex.
The most widely used approach to verify them is still by testing, which is inherently incomplete and hard to scale up to cover the complexity.
In this course we will discuss a number of advanced validation and verification techniques that go far beyond ad-hoc testing.
Exploiting them is an important key towards more reliable complex software.

We will in particular focus on techniques that can be automated, or at least partially automated:
predicate transformation technique, which you can use to symbolically execute a program to calculate its range of input or output. This leads to the bounded but symbolic and fully automated verification technique.
common automated theorem proving techniques, used as the back-end of symbolic execution based verification.
model checking techniques, that can be used to either fully verify a model of a program, even if the number of possible executions is infinite, or to boundedly verify the program itself.
We will also discuss several common ways to define the semantic of programs, from which correctness can be defined and proven.

Course form

Lectures, projects.

Literature

Lecture notes, on-line documentation, and papers.

STATISTICAL LEARNING AND STOCHASTIC PROCESSES

Course goals

After completing this course, the student will be able to:
specify a stochastic process appropriate to model a given dynamic system
conduct analytical derivations on the behavior of the stochastic process
conduct simulations of the stochastic process
estimate the parameters of a stochastic process from data
perform hypothesis tests to test a given stochastic model, or to compare different models
select an appropriate model from a number of candidates using scoring functions (AIC/BIC)

Assessment

The final grade consists of:
written exam (60% of the final mark)
2 practical assignments (20% each)
The minimum grade to pass the exam is 5.5.
For each of the assignments, the minimum grade to pass is 6.
If an assignment is failed, then it can be redone. The maximum grade for a retake is 7.

To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Prerequisites

Knowledge of probability and statistics, multivariable calculus, linear algebra, algorithms and data structures.
Ability to (learn how to) program in Python or R.

Content

This course introduces stochastic processes to model complex, dynamic systems.
First we discuss a number of important stochastic processes such as Markov chains and Poisson processes. We study their properties, and discuss how to determine the probabilities of different events for a given stochastic process
both analytically and through simulation.
Then we turn to the problem of estimating the parameters of stochastic processes, e.g. the transition probabilities of a Markov chain, from data. Here we enter the area of statistical inference.

As a possible area of application, think of an airport where data is available on the number of passengers present, the duration of their stay at the airport premises, the facilities used, etc. All these data can be used to specify a stochastic process,
which supports the analysis of the system (e.g., how busy is the airport expected to be tomorrow) as well as decision-making (e.g., the increase/decrease of the number of check-in desks).

The course provides students with the skills to use data for model specification (stochastic processes), model analysis and decision-making.

List of topics:
1. Monte Carlo simulation
2. Markov Chains, Poisson processes
3. Gaussian processes
4. Markov decision processes
5. statistical inference for Markov chains and Poisson processes
6. linear regression, autoregressive models

Course form
2 lectures every week, and 1 tutorial session every week. The lectures and the tutorials are on campus.

Literature
Recommended:

H.M. Taylor & S. Karlin, "An introduction to stochastic models", third edition. Academic Press, 1998
Dekking, F.M., Kraaikamp, C., Lopuhaa, H.P., Meester, L.E., "A Modern Introduction to Probability and Statistics: Understanding why and how". Springer Science & Business Media, 2005.

EVOLUTIONARY COMPUTING

Course goals

After completing the course, students have

a thorough knowledge of the concepts, techniques, analyses, and algorithms in the field of evolutionary computation and meta-heuristic search algorithms.

theoretical knowledge to understand the behavior of evolutionary and meta-heuristic search algorithms.

a thorough knowledge of state-of-the-art applications of evolutionary computation and meta-heuristic search algorithms.

a thorough knowledge of solving multi-objective optimization problems with metaheuristic search algorithms.

and are capable of

designing efficient and high performance meta-heuristic search problem for diverse discrete optimization problems.

reading and understanding key journal publications in the field of evolutionary computation and meta-heuristic search algorithms.

experimentally comparing different meta-heuristic search algorithms on a set of benchmark problems.

implementing meta-heuristic search algorithms to solve hard, discrete optimization problems.

analyzing the performance and sensitivity of meta-heuristic search algorithms.

performing a statistically sound analysis of the experimental results of different meta-heuristic search algorithms.

working together with other students on designing, building, and testing evolutionary and meta-heuristic search algorithms.

making English language presentations in writing of one's own research.

making English language presentations orally of one's own research.

Assessment

The assessment consists of

a written exam (50% of the final mark)

lab assignments (40%)

paper report (10%).

To qualify for a repair of the final result the mark needs to be at least a 4, or "AANV".

Content

Evolutionary algorithms are population-based, stochastic search algorithms based on the mechanisms of natural evolution.

This course covers how to design representations and variation operators for specific problems. Furthermore convergence behavior and population sizing are analyzed.

The course focuses on the combination of evolutionary algorithms with local search heuristics to solve combinatorial optimization problems like graph bi-partitioning, graph coloring, and bin packing.

Course form

Lectures, lab sessions.

Study material
To be announced.

OPTIMIZATION FOR SUSTAINABILITY

Course goals

In this course students learn to:

understand discrete-event simulation models and statistical analysis methods for these models
create a discrete-event simulation model for a given system
perform a scientific sound simulation study including statistical analysis
understand optimization models for sustainable mobility and energy systems
understand applications of optimization algorithms and simulation to sustainable mobility and energy systems
assess and present scientific papers on optimization algorithms and simulation to sustainable mobility and energy systems
identify and describe possibilities for applications of optimization algorithms and simulation to sustainable mobility and energy systems

Assessment

The grading consists of the following parts:

hand-in exercises: 1) simulation model (5% of t he final mark), 2) input analysis (5%), 3) optimization challenge energy and mobility (10%)
simulation assignment (40%)
seminar (40%)

To obtain a pass grade

you have handed in all the assignments and the preparation documents for the seminar
you have given a seminar presentation
the grade for the optimization challenge exercise has to be at least 5.0
you attended the milestone and feedback meeting of the simulation meeting in person. It is not sufficient if the other member of your group attended.
you have attended the seminar sessions (exceptions for valid reasons are possible, contact teacher before the session)

Minimum effort to qualify for 2nd chance exam:

your final grade is at least 4
you delivered the hand-in exercises on time.
you received a pass for the milestone of the simulation assignment

You can take an additional exam for at most one out of hand-in exercises, simulation assignment, seminar. Which one is decided in discussion with the teacher.

Prerequisites

This course in primarily meant for students in the master Computing Science.
The focus is on operations research and computer science.
The required knowledge is mathematics and statistics at the level of the bachelor computer science, algorithms (e.g. Algorithms from the bachelor) and programming at the level of Imperative Programming.

Content

The current energy transition leads to many changes in the energy as well as the mobility system. In this course, we study algorithmic techniques to optimize the performance of future energy systems and discuss topics from sustainable public transportation.

We discuss Mixed Integer Linear Programming (MIP) formulations, branch-and-cut, and simulation.
Students learn how to apply these techniques to different optimization problem related to energy systems, such as network design, unit commitment, load flow, demand response, and storage optimization.
Moreover, we discuss question around electric buses, environmental friendly bus driving, and multi-modal route planning.
The topics in this course are related to recent research in the Algorithms and Complexity group.
In this course you will learn algorithms and methods that are also used by companies to solve real-world problems.

This course will not deal with political or policy issues in the domain of `sustainability', but is devoted entirely to (a selection of) the computational models and optimizing algorithms that are developing in the field.

Course form
Lectures, seminar: discussion and presentation of papers

Literature
Material will be made available through MS Teams.

SCHEDULING AND TIMETABLING

Course goals

After completing this course, you are ready to solve real world problems in the field of Logistics. Moreover, you will train your analytic skills and common sense.

Assessment

There will be an oral exam, which counts for 50% of the final mark. On this you must at least score a 5.0 to pass.
Furthermore, there are take-home exercises, which count for 25%.
Finally, each student has to present a paper, which counts for 25% as well.

To qualify for a repair of the final result the mark needs to be at least a 4, or “AANV”.

Prequisites

We advise students to choose this course in their second year of the master, when they have followed INFOMADS Algorithms for decision support.
If you have followed the bachelor courses INFOOPT Optimalisering en Complexiteit, INFOAL Algoritmiek, or INFOB3DW Discrete Wiskunde, then there is no need to follow the course INFOMADS before INFOSTT.

Content

The course consists of two major subjects: Timetabling and Scheduling (which are treated in this order).

In the first part, we discuss problems that occur in practice, like determining the optimal time-table for the Dutch Railways, the planning of buses and drivers in Rotterdam, and the construction of a timetable for school. The focus here is on finding good solutions that can be used in practice. Examples of techniques we use here are ILP models, min cost max flow, and local search.
In the second part we give an introduction into scheduling. Here we discuss questions like: what is the best order to execute a set of tasks in case you want to minimize some given objective, like total weighted completion time? The focus in this part is on theory.

Course form

Lectures.

Literature

Handouts.