INDIAN STATISTICAL INSTITUTE

Students' Brochure

PART II

Master of Technology (M.Tech.) in Computer Science

(Effective from Academic Year 2026-27)

(See PART I for general information, rules and regulations applicable to all programmes)

The Headquarters
203 BARRACKPORE TRUNK ROAD
KOLKATA 700108

INDIAN STATISTICAL INSTITUTE

Master of Technology (M.Tech.) in Computer Science

Contents

The structure of the M.Tech. in Computer Science (MTCS) programme differs from that of the other degree programmes mentioned in Part I, which lays out general information, rules and regulations applicable to all programmes. Some information/rules specific to this programme supersede the corresponding information/rules in the general brochure; see Section 3 for details.

1 CS and Non-CS Streams

Every student who is admitted to the programme is categorised as either a **CS-stream** student or a **Non-CS-stream** student.This categorisation is based on the general structure of the admission process that is followed for the MTCS programme.[1] Typically, students qualify based on

- **EITHER** their performance in two written tests conducted by the institute,

  OR a valid GATE score;
- **AND** their performance in a subsequent interview.

One or more of the written tests generally have separate sections, comprising questions on (i) Mathematics and (ii) Computer Science.

To be categorised as a **CS-stream** student, the candidate must be a graduate in Computer Science / Information Technology / any other discipline regarded as equivalent by the admission committee[2], **AND** must have qualified based on

- **EITHER** their answers to questions from the Computer Science section(s) of the written test(s);
- **OR** a valid GATE score (above a threshold as decided by the admission committee) in Computer Science / Information Technology / any other discipline regarded as equivalent by the admission committee.

All other candidates will be categorised as **Non-CS-stream** students.

The stream of each student is recorded in the final selection list by the MTCS admission committee. The courses in this programme are categorised as **compulsory**, **formative** and **elective**. The curricula for the two streams differ in terms of the number of courses a student

[1] If the admission process followed in a particular year has a significantly different structure, the process for assigning streams to incoming candidates will be specified in the Admission Prospectus for that year.

[2] Likewise, in case of any dispute / lack of clarity about whether a candidate satisfies this eligibility criterion, the admission committee will take the final decision, using the relevant year's Admission Prospectus as a guideline.

has to pass in each category. Students are required to follow the curriculum specified for their respective stream.

2 Structure of the Programme

2.1 Structure for CS stream

A student of the CS stream has to pass the courses listed below. The requirements of Compulsory and Pool A Formative courses for the CS stream students, as mentioned in the brochure, have to be completed within the first year of the programme.

1. **Compulsory non-credit course:**

   - Introduction to Programming

   This requirement may be waived if a student passes a programming test conducted within the first two weeks of the first semester.

2. **Three compulsory courses** for the CS stream:

   - Discrete Mathematics
   - Data Structures and Algorithms
   - Design and Analysis of Algorithms

3. **Four formative courses** from the list of formative courses for the CS stream given below.

   Pool A

   - Abstract Algebra and Number Theory
   - Linear Algebra
   - Probability and Stochastic Processes
   - Statistical Methods

   Pool B

   - Compiler Construction
   - Computer Networks
   - Computing Laboratory
   - Database Management Systems
   - Digital Design and Computer Architecture
   - Operating Systems
   - Theory of Computation

   At least two formative courses must be from Pool A.

4. **Eight elective courses** from the list of elective courses

5. **Two more courses**, which may be chosen from the list of formative courses (see 3 above), and/or from the list of elective courses.

   An eligible student (see Section 3.5 for eligibility criteria) may choose to do a **Minor Project** in lieu of one of these two courses.

6. **Dissertation** (equivalent to three courses, one of which is typically taken in the third semester, and the remaining two in the fourth semester)

2.2 Structure for non-CS stream

A student of the non-CS stream has to pass the courses listed below. The requirements of Compulsory and at least two Formative courses for the non-CS stream students (both with and without a Master's degree in Mathematics/Statistics), as mentioned in the brochure, have to be completed within the first year of the programme.

1. **Compulsory non-credit course:**

   - Introduction to Programming

   This requirement may be waived if a student passes a programming test conducted within the first two weeks of the first semester.

2. **Five compulsory courses** for the non-CS stream:

   - Discrete Mathematics
   - Data Structures and Algorithms
   - Design and Analysis of Algorithms
   - Computing Laboratory
   - Digital Design and Computer Architecture

3. **Four formative courses** from the list of formative courses for the non-CS stream:

   - Abstract Algebra and Number Theory
   - Linear Algebra
   - Probability and Stochastic Processes
   - Statistical Methods

- Compiler Construction
- Computer Networks
- Database Management Systems
- Operating Systems
- Theory of Computation

The following restrictions are applicable when choosing formative courses.

- **Students with a Master's degree in Mathematics** will not be permitted to take Linear Algebra or Abstract Algebra and Number Theory.
- **Students with a Master's degree in Statistics** will not be permitted to take Probability and Stochastic Processes or Statistical Methods.
- Students are discouraged from taking formative courses on subjects that they have studied at a similar level in their previous programmes. The final selection of formative courses should be taken in consultation with the mentor committee.

4. **Eight elective courses** from the list of elective courses.

   An eligible student (see Section 3.5 for eligibility criteria) may choose to do a **Minor Project** in lieu of one of these elective courses.

5. **Dissertation** (equivalent to three courses, one of which is typically taken in the third semester, and the remaining two in the fourth semester)

2.3 Semester-wise layout of the compulsory and formative courses

All compulsory and formative courses will be offered at least once in every academic year. The distribution of those subjects according to odd and even semesters is given below. There may be some variation in the semester-wise allocation of courses from time to time.

2.3.1 Odd semester (the first semester of an academic year)

Subject	Course Type	
	CS-stream	non-CS stream
Introduction to Programming	Remedial	Remedial
Discrete Mathematics	Compulsory	Compulsory
Data Structures and Algorithms	Compulsory	Compulsory
Digital Design and Computer Architecture	Formative	Compulsory
Database Management Systems	Formative	Formative
Abstract Algebra and Number Theory	Formative	Formative
Linear Algebra	Formative	Formative
Probability and Stochastic Processes	Formative	Formative
Statistical Methods	Formative	Formative

2.3.2 Even semester (the second semester of an academic year)

Subject	Course Type	
	CS-stream	non-CS stream
Design and Analysis of Algorithms	Compulsory	Compulsory
Computing Laboratory	Formative	Compulsory
Compiler Construction	Formative	Formative
Computer Networks	Formative	Formative
Operating Systems	Formative	Formative
Theory of Computation	Formative	Formative

2.4 Elective courses

Elective courses are classified into four tracks, namely (1) Theory, (2) Systems, (3) Cryptology and Security, (4) Artificial Intelligence and Data Science. An elective course may belong to more than one track. A student who passes a certain number of elective courses in a specific track can obtain a specialisation in that track (details regarding specialisation are noted later) which will be mentioned in the marksheet.

The following table provides a list of elective courses (along with the track(s) they belong to) that will generally be offered in the **odd semester**.

Course	Theory	AI & DS.	Crypto & Sec.	Systems
Advanced Algorithms	✓		✓	
Artificial Intelligence		✓		
Computational Geometry	✓			
Computer Vision		✓		✓
Computing Systems Security			✓	✓
Cryptology I	✓		✓	
Distributed Algorithms	✓		✓	
Generative AI and Foundational Models		✓		
Information Retrieval		✓		
Natural Language Processing		✓		
Quantum Computation and Cryptography	✓		✓	

The following table provides a list of elective courses (along with the track(s) they belong to) that will generally be offered in the **even semester**.

Course	Theory	AI & DS.	Crypto & Sec.	Systems
Coding and Information Theory	✓		✓	
Computer Graphics		✓		✓
Cryptology II	✓		✓	
Neural Networks and Deep Learning		✓		
Image Processing		✓		✓
Logic for Computer Science	✓		✓	
Machine Learning	✓	✓	✓	✓
Math Toolkits for CS	✓		✓	
Optimization Techniques	✓	✓		✓
Reinforcement Learning	✓	✓		

The following table provides a list of elective courses (along with the track(s) they belong to) that will be offered if there is enough demand and there are interested teachers.

Course	Theory	AI & DS.	Crypto & Sec.	Systems
Advanced Computer Architecture				✓
Advanced Operating Systems				✓
Advanced Logic and Automata Theory	✓			
Algorithms for Big Data	✓	✓		
Algorithms for Electronic Design Automation				✓
Computability Thoery	✓		✓	
Computational Algebra and Number Theory	✓		✓	
Computational Complexity	✓		✓	
Computational Finance	✓	✓		
Computational Game Theory	✓	✓		
Computational Learning Theory	✓	✓	✓	
Computational Molecular Biology and Bioinformatics		✓		
Computational Topology	✓	✓		
Cyber-Physical Systems				✓
Digital Signal Processing		✓		✓
Discrete and Combinatorial Geometry	✓			
Fault Tolerance and Testing				✓
Finite Model Theory	✓		✓	
Formal Verification				✓
Graph Algorithms	✓			
Introduction to Cognitive Science		✓		
Mobile Computing				✓
Principles of Programming Languages	✓		✓	✓
Quantum Information Theory	✓		✓	
Selected Topics in Image Processing		✓		
Special Topics in Complexity Theory	✓			
Statistical Computing		✓		
Parallel Algorithms	✓			
Topics in Privacy			✓	

3 Information/rules specific to the MTCS programme

The information given in this section supplements, and occasionally supersedes (but is never superseded by) the information given in the general brochure for all non-JRF degree programmes. In particular, the notions of promotion and repeating a year, and the corresponding sections of the general brochure, do not apply to the MTCS programme.

3.1 Duration of the programme

The expected time for completion of the programme is two years. A student may take upto a maximum of three years for completion. However, after completion of the second year, a student will generally not be eligible for stipends, contingency grants or hostel facilities. In special circumstances, e.g., medical emergencies, exceptions to this rule may be made by the Dean, in consultation with the Teacher's Committee/Mentor Committee, on a case by case basis.

3.2 Waiver of class attendance

For a formative course, a CS-stream student can bypass regular classes and claim credit by completing only the assignments and passing the examination(s). This may give the student the flexibility to attend an elective during that time, and thus complete the course requirements early. This option is not available for a student in the non-CS stream.

- If a student opts for waiver of class attendance, (s)he is required to seek permission from the Mentor Committee.
- The teacher of the course and the Dean's Office need to be informed before the mid-semester week.
- The usual attendance requirement for the student in such cases would be completely waived for the specific course.
- Under this option, the student has to obtain at least 60% to pass.

3.3 Registering for a course

Within 2 weeks of the start of a semester, a student will have to inform the Dean's Office through the Mentor Committee about the courses that (s)he is registering for. A student cannot opt out of a course after these 2 weeks. The scores obtained in all the courses thus registered will be recorded in the marksheet. Even after passing a course, a student may, with the approval of the Mentor Committee, opt to take the course again in subsequent semester(s). In such cases, the highest score obtained in the course across all attempts will be considered when computing the student's final result.

3.4 Dissertation

A student is required to work for a dissertation on a topic assigned/approved by the Project and Dissertation Committee under the supervision of a suitable ISI faculty member.

The work for a dissertation should be substantial and related to some important problem in an area of Computer Science and/or its applications. It should have notable theoretical or practical significance. A critical review of recent advances in an area of Computer Science and/or its applications with some contributions by the student is also acceptable as a dissertation.

Duration: The work for the dissertation should cover two semesters, will typically commence at the beginning of the third semester, and be completed along with all other courses of the fourth semester. A student who opts for finishing the degree programme in more than two years but within three years, as per guidelines in the brochure, can commence her/his dissertation at the beginning of the fifth semester and complete it by the end of the following semester. The assignment of dissertation topics and the evaluation of students' work will be coordinated by a Project and Dissertation Committee that is to be formed by the Dean's Office in consultation with the Mentor Committee before the students of that batch are promoted to their second year.

The selection of the supervisor for the dissertation of each student should be done within the first four weeks of the semester in which the student starts her/his dissertation.

Evaluation: An interim progress report of the dissertation work should be submitted at the end of the semester in which it was commenced (typically by the first week of December in the third semester). This report will be evaluated by a committee formed by the Project and Dissertation Committee. The committee will consist of relevant ISI faculty and the supervisor(s).

- If the student's interim report is regarded as satisfactory, the committee will award the student marks out of 100.
- If the report is found to be unsatisfactory, the Project and Dissertation Committee will, in consultation with the supervisor(s), allow the student extra time (not exceeding a month) to make up. If the progress of the student is deemed to be satisfactory at this second evaluation, s/he will be awarded marks out of 100, as above. Otherwise, the student will have to start the dissertation afresh in the beginning of the following semester (typically the fourth semester), possibly with a different supervisor, and continue for two semesters (typically the fourth and fifth semesters).

There will be no back paper for the dissertation.

The final dissertation should be submitted at the end of two semesters since its commencement as indicated in the academic calendar (typically by the middle of June of the standard year of completion of the batch). The dissertation will be evaluated out of 200 marks by a committee formed by the Project and Dissertation Committee. The committee will consist of relevant ISI faculty, the supervisor(s) and one or more external expert(s). The student has to defend his/her dissertation in an open seminar.

If the final dissertation is not submitted within the stipulated deadline, or if the evaluation committee, as mentioned earlier, evaluates the student to have failed in the dissertation, then the dissertation will be evaluated at the end of the next semester (typically the fifth semester). Thus, if a student's interim progress report is regarded as unsatisfactory, or if s/he fails the final dissertation evaluation, her/his dissertation will typically be re-evaluated at the end of the fifth semester.

If a student undertakes her/his dissertation during the fifth and sixth semesters, and fails in the interim or the final evaluation organized by the Project and Dissertation Committee, then the Dean and the Mentor Committee will decide on a case-by-case basis whether the student is not to be awarded the degree at all, or to be given one last chance to finish the dissertation if the student has strong reasons for the failure. In these exceptional cases, while continuing to work on the dissertation for another semester, the student will be enrolled as an ISI student (full-time or part-time) but will not be eligible for stipend and hostel facilities. The allowable exceptions are only for valid medical grounds, to be decided upon by the Dean, in consultation with the concerned mentor committee.

Joint supervision: Joint supervision of a dissertation is possible, with permission from the Mentor Committee. In such a case, the student is allowed to spend considerable time outside the institute, provided his/her course requirements are fulfilled. The primary supervisor of a jointly supervised dissertation needs to be an ISI faculty.

3.5 Minor project

A student must work on a minor project for one full semester, and may opt for it in either the third or the fourth semester. The minor project will be evaluated out of 100 marks.

A student is eligible to undertake a minor project only if (s)he satisfies the following:

- has passed at least ten courses in the first two semesters,

- the aggregate score of the best ten courses taken in the first two semesters is at least 75%.

An eligible student who does not opt for a minor project, as well as a student who is not eligible for a minor project, has to pass one additional course, as specified in Sections 2.1 and 2.2.

A student who opts for the minor project has to decide on a topic for the project within three weeks from the start of the third or fourth semester. The topic has to be approved by the Project and Dissertation Committee, and should be significantly different from that of the dissertation, as judged by the Project and Dissertation Committee. The Project and Dissertation Committee will also be responsible for the evaluation of the project, and will constitute a committee for this purpose, in accordance with the guidelines for the evaluation of the dissertations (see Section 3.4 above).

3.6 Internship/industrial training

There would be a mandatory 12 weeks gap between the first and the second year in the academic calendar. During this period, students may pursue internship/industrial training outside the institute in research institutes or public/private sector organizations in India or abroad. However, internship is not mandatory.

A student who undergoes internship/industrial training somewhere in India during the training period may receive his/her usual monthly stipend/remuneration/emoluments either from the Institute (ISI) or from the host organization at his/her own discretion with appropriate permission from the Dean. The students who are placed outside Kolkata for training will be reimbursed sleeper class to and fro train fare from Kolkata to the place of internship unless the host organization provides travel assistance.

Training may also be arranged at a research/R&D organization abroad. In case a student undergoes practical training at such a place abroad, the Institute (ISI) will not provide any financial support including the monthly stipend for that period.

3.7 Specialisation

Among the eight elective courses, if a student passes at least five elective courses from a specific track and does his/her dissertation in a topic which falls under that track, (s)he

graduates with a specialisation. The track to which a dissertation belongs is to be decided by the Project and Dissertation Committee during or before the mid-term evaluation.

A student is eligible to obtain a **double specialisation** if (s)he fulfils the following:

- Passes at least 10 elective courses with at least five in the two separate tracks in which (s)he wishes to obtain the specialisations. One elective course cannot be counted for two different specialisations.
- Obtains passing marks in a minor project.
- The minor project and the dissertation are in two different tracks for which (s)he wishes to obtain the specialisation.

3.8 Final result

Upon passing the minimum number of courses (as described in Sections 2.1 and 2.2), minor project (if applicable) and dissertation, the final percentage obtained by a student is to be computed as follows.

- For a student who opts to complete the programme with a single specialisation, or without a specialisation, the final score is computed out of 2000 marks, and is the sum of (a) the best seventeen scores obtained in the courses passed as per requirements[3], and (b) the dissertation marks.

  If the student has done a minor project, then her/his score in the minor project may be counted in lieu of one course.

  For a student who opts to complete the programme with a single specialisation, the seventeen courses selected above should include at least five elective courses from the track in which the student desires to obtain the specialisation (see Section 3.7).

- For a student who opts to complete the programme with a double specialisation, the final score is computed out of 2200 marks, and is the sum of (a) the best seventeen scores obtained in the courses passed as per requirements (see Section 3.7), (b) the score in the minor project, and (c) the dissertation marks.

[3] A student who is required to pass N_c compulsory courses, N_f formative courses and N_e elective courses, may, with permission from the Mentor Committee, repeat one or more of the compulsory courses, take more than N_f formative courses, or more than N_e elective courses. In such cases, the final score will be computed using the best score obtained in each compulsory course, the best N_f scores obtained in the formative courses, and the best N_e scores obtained in the elective courses. Please also see Section 3.3.

The seventeen courses should include at least ten elective courses with at least five from each of the two tracks in which the student opts for the specialisations.

- If a student attempts a course multiple times, the highest score obtained across all attempts is considered when computing the aggregate (see Section 3.3).

In all cases, the scores of all the courses a student has registered for, as per rules in Section 3.3, will be reflected in the final mark sheet. The division indicated in the final mark sheet will be determined as per the rules described in the general brochure.

3.9 Stipend

On admission, each student will receive the institute specified stipend, subject to the rules described in the general brochure for all non-JRF degree courses, with the following modification in respect of 'Performance in coursework' criterion. A student is eligible for a full stipend

- in the first semester, only if (s)he registers for at least four courses;
- in the second semester, only if (s)he
  - registers for a total of at least nine courses in the first two semesters, and
  - obtains at least 45% in each of at least four courses in the first semester with an average score of 60% in the best four courses;
- in the third semester, only if (s)he
  - registers for a total of at least fourteen courses in the first three semesters, and
  - obtains at least 45% in each of at least nine courses in the first two semesters with an average score of 60% in the best nine courses;
- in the fourth semester, only if (s)he obtains at least 45% in each of at least fourteen courses (or twelve courses and a minor project) in the first three semesters with an average score of 60% in the best fourteen courses. Additionally, the student must have obtained pass marks in the mid-term evaluation for the dissertation.

Further, a student who is ineligible for a full stipend in the second, third or fourth semester, may be eligible for a half stipend in that semester (i.e., second, third or fourth semester) if (s)he fulfils the registration requirements, and gets at least 45% score in each of the minimum number of individual courses as per the schedule listed above.

3.10 Mentor Committee

A group of students entering the M.Tech. (CS) programme in a particular year is treated as a batch. The Mentor Committee for each batch of students is announced at the time of their admission, and remains valid until all the students of that batch leave the programme. This Committee advises the students on their choice of courses (including opting for a course more than once even after passing it) and specialisations, and as a matter of fact any other problem. Students should approach the Mentor Committee if they face any problem.

A student's choice of courses in a semester needs to be approved by the Mentor Committee. Students are required to submit the final course choice list to the Dean's office, signed by the members of the Mentor Committee for her/his batch, within 2 weeks from the start of the semester.

The Chairperson acts as the Class Teacher for all students in a particular batch. All communications on academic matters with other authorities or statutory bodies of the institute by a student should be routed through the Class Teacher.

3.11 Teachers' Committee

The Teachers' Committee in a particular semester consists of all the teachers of the courses (including regular courses and lab courses) opted for by all students (across all years) of the MTCS programme in that semester. The Chairperson and Convener of the existing Mentor Committees will be invited members of the Teachers' Committee. The Dean of Studies is the Chair of the Teachers' Committee. Every decision related to the academic performance of a student is taken after deliberation in the relevant Teachers' Committee. A decision can be taken by any higher authority/statutory body only after the recommendations of the relevant Teachers' Committee have been taken into cognizance.

3.12 Project and Dissertation Committee

This committee, formed for a batch at the start of its second year, is responsible for the following tasks:

- Create a list of projects and dissertation topics from the relevant faculty members and circulate the list among the students of that batch. However, the students are free to choose a topic outside the list, provided a faculty member agrees to supervise such a project.

- Help a student with finding a Project/Dissertation topic and supervisor, if needed.
- For a student opting for both Minor Project and Dissertation, ascertain that the two problems are different. An approval of the committee is mandatory before a student is assigned a Minor Project.
- For a student who is opting for specialisation in a track, ascertain that the topic of the Dissertation is in the area of that track.
- For a student who is opting for double specialisation, ascertain that the topic of the Minor Project is in the area of the second track of specialisation.
- Conduct evaluation of the minor projects and dissertation at appropriate times.

3.13 Prizes

At the end of the four semesters of a particular batch, the Mentor Committee may nominate a few students for their outstanding academic performances during this period, for awards. There will be no consideration of prize for semester-specific performance.

4 Detailed Syllabi of the Courses

The courses are categorised as compulsory, formative and elective. Each course has a detailed syllabus, a few prerequisites, and names of reference books. For some courses, the suggested duration that should be spent on each topic is also provided as an additional guideline.

Prerequisites. Many courses have a few suggested prerequisite(s). The prerequisite courses have to be passed either at ISI or in the undergraduate / postgraduate degrees obtained before joining ISI. The students need to confirm with the concerned teacher if her/his prerequisite courses satisfy the demands of the particular course being taught at ISI. It is the responsibility of the students to verify from the concerned teacher that they satisfy the prerequisites.

4.1 Compulsory and Formative Courses

Abstract Algebra and Number Theory

(a) Groups: Elementary properties, subgroups and cosets, Lagrange's theorem, cyclic groups, normal subgroups and quotient groups, homomorphism and isomorphism, permutation groups, Sylow's theorem and applications. Applications to number theory. **(4 weeks)**

Rings and fields: Introduction to rings, subrings, ideals and quotient rings, homomorphism and isomorphism, integral domains, principal ideal domain(PID), Euclidean domain(ED), Euclidean algorithm for gcd, extended Euclidean algorithm, modular inverse of an integer, Chinese Remainder Theorem(CRT), Euler's ϕ-function, quadratic residues, ring of polynomials, irreducible polynomials, factorizations, fields, field of fractions, finite fields. **(7 weeks)**

Selected topics in number theory: For example, arithmetic functions, Diophantine equations, continued fractions. **(3 weeks)**

(b) **Prerequisites:** None

(c) **References:**

1. I. Niven, H. S. Zuckerman, and H. L. Montgomery, An Introduction to the Theory of Numbers. Wiley, 1991.
2. J. B. Fraleigh, First Course in Abstract Algebra. Narosa/Addison-Wesley, New Delhi/Reading, 1982/1978.
3. I. N. Herstein, Topics in Algebra. John Wiley & Sons., 1987.
4. M. Artin, Algebra. Pearson, 2010.
5. D. M. Burton, Elementary Number Theory. Allyn and Bacon, 1976.

Compiler Construction

(a) Introduction: compilers vs. interpreters, phases and passes, bootstrapping.

Lexical analysis: regular expressions and their application to lexical analysis, implementation of lexical analysers, lexical-analyser generators, use of a lexical analyser generator tool (e.g., lex, flex, or similar), symbol tables.

Parsing: formal grammars and their application to syntax analysis, ambiguity, recursive descent and predictive parsers, LR parsers, error detection and recovery.

Syntax directed translation: synthesised and inherited attributes, S-attributed and L-attributed definitions, augmented LL(1) and LR parsers, use of a parser generator tool (e.g., yacc, bison, or similar).

Type checking: represention of types, type checking.

Intermediate code generation: 3-address code, intermediate code generation for standard constructs (assignment statements, single and multi-dimensional array variables, flow control, function calls, variables declarations).

Memory management and runtime support: activation stack, stack frames, calling and return sequences, access to non-local storage.

Code generation: Register assignment and allocation problems, instruction selection, simple code-generation from intermediate code.

Code optimisation: peephole optimisation, syntax-driven and iterative data flow analysis, common sub-expression elimination, constant folding, copy propagation, dead code elimination, loop optimisation (code motion, induction variables).

Misc. topics (depending on time available): basic concepts of compiling object-oriented and functional languages; just in time compiling; interpreting byte code; garbage collection.

(b) **Prerequisite(s):** Digital Design and Computer Architecture; Theory of Computation.

(c) **References:**

1 A. V. Aho, R. Sethi and J. Ullman, Compilers: Principles, Techniques and Tools. Addison–Wesley, 1986.

2 A. V. Aho, M. S. Lam, R. Sethi and J. Ullman, Compilers: Principles, Techniques and Tools. 2nd ed., Addison–Wesley, 2007.

3 A. W. Appel, M. Ginsburg, Modern Compiler Implementation in C. Cambridge University Press, 1998.

4 A. I. Holub, Compiler Design in C. Prentice Hall, 1990. https://holub.com/compiler/

Computer Networks

(a) Introduction: Characteristics of computer networks, Network hardware, Network software, Layered network structures and Reference models.

Data communication fundamentals: Analog and digital transmissions, Different transmission media, Different transmission impairments, Different modulation techniques, Channel capacity, Basic concept of spread spectrum (DSSS and FHSS), Asynchronous and synchronous transmission, Multiplexing.

Communication networks: Introduction to LANs, MANs, and WANs; Switching techniques (Circuit-switching and Packet-switching), Network topologies, Ethernet and its performance, Repeaters, Hub, bridges, Switches and Routers.

Data link layer: Basic functions and services, Framing techniques, Error detection and correction, Flow control (Stop-and-wait and Sliding window), Performance analysis of stop-and-wait and sliding window protocols, MAC Protocols (ALOHA, CSMA, CSMA/CD, Collision free protocols, Limited contention protocol), Performance analysis of different MAC protocols, Wireless LAN (MACA, CSMA/CA).

Network Layer: Design issues, Internet protocols (IPV4, IPV6, ICMP, ARP, RARP, DHCP), IP addressing (Classful and Classless), Introduction to subnetting, NAT, Routing (Flooding, Distance vector, Link state), Congestion control (choke packets, leaky bucket, token bucket).

Transport Layer: Design issues and services, Connection establishment and release, TCP, UDP, TCP congestion control, TCP timer management, RPC, Introduction to ports, sockets and socket programming.

Application Layer: WWW, Emails, DSN.

Network Labs: Interprocess communications and socket programming: Implementation and realization of simple echo client-server over TCP and UDP, proxy web server, FTP, TELNET, Chat programs, DNS and HTTP. Implementation of client-server applications using remote procedure call. Create sockets for handling multiple connections and concurrent server. Simulating PING and TRACEROUTE commands.

(b) **Prerequisites:** None

(c) **References:**

1. L. L. Peterson and B. S. Davie, Computer Networks: A Systems Approach. Morgan Kaufmann Publishers, 2007.
2. A. S. Tanenbaum, Computer Networks. 5th ed., Prentice Hall, 2010.
3. W. Stallings, Data and Computer Communcations. 10th ed., Prentice Hall, 2017.

4. D. P. Bertsekas and R. G. Gallagher, Data Networks. 2nd ed., Prentice Hall, 1992.
5. B. A. Forouzan, Data Communications and Networking. McGraw-Hill.
6. W. R. Stevens, Unix Network Programming. PHI, 2009.
7. W. R. Stevens, TCP/IP Illustrated. Volume 1: The Protocols, Addison-Wesley Professional.

Computing Laboratory

(a) This laboratory course is meant to complement the Data Structures and Algorithms and Design and Analysis of Algorithms courses. The material, assignments and examinations are to be designed based on the coverage in those courses. The initial programming language can be `C` or can be decided by the instructor based on the background of the students. For this course, conventional mid-semester and end-semester examinations may be replaced by programming tests.

The laboratory sessions should include but are not limited to:

Problem solving techniques: recursion, divide-and-conquer, branch-and-bound, dynamic programming.

Function pointers, generic data structures, implementation of generic stacks and queues.

Trees: binary trees, recursive and non-recursive traversal of trees; heaps; binary search trees, implementation of balanced search trees (AVL / Red-Black tree).

Tries.

Disjoint set data structures (union-find).

Graphs: directed and undirected graphs, adjacency matrix and list representations, traversal algorithms and their applications; connected components, strongly connected components; shortest path algorithms; minimum spanning tree algorithms.

Introduction to other programming languages: C++ or Python.

Object oriented programming: introduction, classes and methods, polymorphism, inheritance.

Project management: multi-file programs, build systems (GNU make, CMake, Meson or similar), version management using Git.

Program debugging and testing: elementary testing methods, developing the test plan, developing a test, concept of writing automation scripts using bash / tcsh (or similar).

(b) **Prerequisites:** Introduction to Programming, Data Structures and Algorithms, Design and Analysis of Algorithms (may be taken concurrently)

(c) **References:**

1. T. A. Standish, Data Structures, Algorithms and Software Principles in C. Addison-Wesley, Reading, Mass., 1995.
2. L. Nyhoff, C++: An Introduction to Data Structures. Prentice Hall, Englewood Cliffs, 1998.
3. A. M. Tenenbaum, Y. Langsam and M. J. Augenstein, Data Structures Using C. Pearson, 1998.
4. D. E. Knuth, The Art of Computer Programming. Vol. 1, 3rd. ed. Narosa/Addison-Wesley, New Delhi/London, 1997.
5. T. A. Standish, Data Structure Techniques. Addison-Wesley, Reading, Mass., 1980.
6. E. Horowitz and S. Sahni, Fundamentals of Data Structures. Galgotia Booksource, New Delhi, 1977.
7. R. L. Kruse, Data Structures and Program Design in C. Prentice Hall of India, New Delhi, 1996.
8. A. Aho, J. Hopcroft, and J. Ullman, Data Structures and Algorithms. Addison-Wesley, Reading, Mass., 1983.
9. B. Salzberg, File Structures: An Analytical Approach. Prentice Hall, New Jersey, 1988.
10. T. Harbron, File System Structure and Algorithms. Prentice Hall, New Jersey, 1987.
11. P. E. Livadas, File Structure: Theory and Practice. Prentice Hall, New Jersey, 1990.
12. T. Cormen, C. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms. PHI Learning Pvt. Ltd., New Delhi, 2009.
13. S. Sahni, Data Structure, Algorithms and Applications in JAVA. Universities Press (India) Pvt. Ltd., New York, 2005.

14. D. Wood, Data Structure, Algorithms and Performance. Addison-Wesley, Reading, Mass., 1993.
15. M. T. Goodrich, R. Tamassia and David Mount, Data Structures and Algorithms in C++. 2nd ed., Wiley, 2011.
16. B. W. Kernighan and D. M. Ritchie, The C Programming Language. Prentice Hall of India, 1994.
17. B. Gottfried: Programming in C, Schaum Outline Series, New Delhi, 1996.
18. B. W. Kernighan and R. Pike, The Unix Programming Environment. Prentice Hall of India, 1996.
19. R. G. Dromey, How to Solve it by Computers. Pearson, 2008.

Data Structures and Algorithms

(a) Introduction: **[2 weeks]** Initial ideas of algorithms and their resource usage in terms of space and time complexity; RAM model, initial ideas of memory model and memory hierarchy; asymptotic notations; ideas of worst case, average case and amortized case analysis;

Data Structure basics: **[4 weeks]** Idea of Abstract Data Types and its concrete implementation; Basic data structures – Lists, Array implementation of lists; Linked lists, Doubly linked lists. Initial ideas of Queue and Stack and their implementation. Binary trees and their traversal.

Algorithms and Data Structures for searching and sorting: **[2 weeks]** Binary search and binary search trees; finding maximum and minimum, k largest elements in order; basic sorting and searching algorithms; heaps and heapsort, priority queue.

Basic Graph Algorithms and associated data structures: **[3 weeks]** Representation of graphs; traversal algorithms (BFS, DFS) and their applications using Queue and Stack data structure; Kruskal's and Prim's Algorithms and the use of data structure.

Advanced data structures and their applications: **[3 weeks]** Union Find Data structures, tree representation of a set, path compression techniques and analysis; Height-balanced binary search trees; Range search using K-d tree; B-tree and B+ tree.

Programming practices: Apart from theoretical analysis of data structures, programming implementations should be done as assignments. The Introduction to programming and Computing Laboratory courses acts as a supplement to this course.

(b) **Prerequisites:** None

(c) **References:**

1. T. A. Standish, Data Structures, Algorithms and Software Principles in C. Addison-Wesley, Reading, Mass., 1995.
2. L. Nyhoff, C++: An Introduction to Data Structures. Prentice Hall, Englewood Cliffs, 1998.
3. A. M. Tenenbaum, Y. Langsam and M. J. Augenstein, Data Structures Using C. Pearson, 1998.
4. D. E. Knuth, The Art of Computer Programming. Vol. 1, 3rd. ed. Narosa/Addison-Wesley, New Delhi/London, 1997.
5. T. A. Standish, Data Structure Techniques. Addison-Wesley, Reading, Mass., 1980.
6. E. Horowitz and S. Sahni, Fundamentals of Data Structures. Galgotia Booksource, New Delhi, 1977.
7. R. L. Kruse, Data Structures and Program Design in C. Prentice Hall of India, New Delhi, 1996.
8. A. Aho, J. Hopcroft, and J. Ullman, Data Structures and Algorithms. Addison-Wesley, Reading, Mass., 1983.
9. B. Salzberg, File Structures: An Analytical Approach. Prentice Hall, New Jersey, 1988.
10. T. Harbron, File System Structure and Algorithms. Prentice Hall, New Jersey, 1987.
11. P. E. Livadas, File Structure: Theory and Practice. Prentice Hall, New Jersey, 1990.
12. T. Coreman, C. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms. PHI Learning Pvt. Ltd., New Delhi, 2009.
13. S. Sahani, Data Structure, Algorithms and Applications in JAVA. Universities Press (India) Pvt. Ltd., New York, 2005.
14. D. Wood, Data Structure, Algorithms and Performance. Addison-Wesley, Reading, Mass., 1993.
15. M. T. Goodrich, R. Tamassia and David Mount, Data Structures and Algorithms in C++. 2nd ed., Wiley, 2011.
16. M. A. Weiss, Data Structures and Algorithms in C. 2nd ed., Pearson, 2020.
17. Pat Morin, Open Data Structures. Athabasca University Press, 2020.

Database Management Systems

(a) Introduction: Purpose of database systems, data abstraction and modelling, instances and schemes, database manager, database users and their interactions, data definition and manipulation language, data dictionary, overall system structure.

Relational model: Structure of a relational database, operation on relations, relational algebra, tuple and domain relational calculus, salient feature of a query language.

SQL: domain types, construction, alteration and deletion of tables, query structure and examples, natural joins and other set operations, aggregations, nested subqueries, inserting, modifying and deleting data, advanced joins, views, transactions, integrity constraints, cascading actions, autorization and roles. Hands on and practical assignments.

Entity - relationship model: Entities and entity sets, relationships and relationship sets, mapping constraints, E - R diagram, primary keys, strong and weak entities, reducing E - R diagrams to tables.

Introduction to hierarchical and network model: Data description and tree structure diagram for hierarchical model, retrieval and update facilities, limitations; Database task group (DDBTG) model, record and set constructs retrieval and update facilities, limitations.

Databases in application development: cursors, database APIs, JDBC and ODBC, JDBC drivers, Connections, Statements, ResultSets, Exceptions and Warnings. Practical case studies.

Normalization: Anomalies in RDBMS, importance of normalization, functional, multi-valued and join dependencies, closures of functional dependencies and attribute sets, 1NF, 2NF, 3NF and BCNF; (Optionally) 4NF, 5NF and 6NF; Discussion on tradeoff between performance and normalization.

Database tuning: Index selection and clustering, tuning of conceptual schema, denormalization, tuning queries and views;

Query optimization: Importance of query processing, equivalence of queries, join ordering, cost estimation, cost estimation for complex queries and joins, optimizing nested subqueries, I/O cost models, external sort.

Crash recovery: Failure classification, transactions, log maintenance, check point implementation, shadow paging, example of an actual implementation. Concurrency Control in RDBMS: Testing for serializability, lock based and time - stamp based protocols; Deadlock detection and Recovery.

NoSQL: Introduction to NoSQL databases, ACID vs BASE requirements, practical exercises with one NoSQL system (e.g., MongoDB or Neo4j).

MapReduce and Hadoop: Basics of MapReduce, Basics of Hadoop, Matrix-vector multiplication using MapReduce, relational algebra using MapReduce, matrix multiplication using MapReduce, combiners, cost of MapReduce algorithms, basics of Spark, practical exercises using Spark.

(b) **Prerequisites:** None

(c) **References:**

1. A. Silberschatz, H. F. Korth and S. Sudarshan, Database System Concepts. Tata McGraw-Hill, 6th ed., 2011. (Source: http://www.db-book.com)
2. R. Ramakrishnan and J. Gehrke, Database Management Systems, McGraw-Hil, 3rd ed., 2007. (Source: http://pages.cs.wisc.edu/~dbbook)
3. J. Leskovec, A. Rajaraman, J. Ullman, Mining of Massive Datasets. (Source: http://www.mmds.org)
4. C. J. Date, An Introduction to Database Systems, Pearson Education, Inc., 8th ed., 2006.
5. R. Elmasri and S. B. Navathe, Fundamentals of Database Systems. Pearson Education, Inc., 4th ed., 2004.
6. G. Harrison and S. Feuerstein, MySQL stored procedure programming. O'Reilly Media, Inc., 2006.
7. P. J. Sadalage and M. Fowler, NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Pearson Education, Inc., 1st ed., 2013.

Design and Analysis of Algorithms

(a) Paradigms of Algorithm Design and sorting and selection: **[2 weeks]** Initial ideas of algorithmic paradigms like divide-and-conquer, greedy, induction, dynamic programming, etc. Comparisons between various sorting algorithms, Quick sort (Average case complexity), lower bounds on comparison based sorting algorithm, non-comparative (linear time) sorting algorithms;

Recursive Algorithms: **[2 weeks]** Karatsuba Algorithm; Median Finding Algorithm; Strassen's Algorithm; Dynamic Programming Algorithms – longest common subsequence, matrix chain multiplication, etc.

Hashing: **[1 weeks]** Hash tables, hash functions, open addressing, perfect hashing.

Graph Algorithms: [2 weeks] Shortest Path Algorithms; Applications of depth first search – cycle detection in graph, topological sort, articulation points in graphs, strongly connected components; Network Flow – Max-Flow Min-Cut theorem, Max-flow algorithms like Ford-Fulkerson;

Algorithms from different domains: [2 weeks] FFT, Number Theoretic Algorithms, Geometric Algorithms – convex hull, line segment intersection.

Randomized Algorithms: [1 week] Randomized quicksort, Min-cut, Universal Hashing.

"Hard" Problems: **[3 weeks]** P vs. NP, NP-completeness, statement of Cook's theorem, some important NP-complete problems, problems beyond NP, ideas on attacking NP-complete problems, using reductions to solve hard problems.

Approximation Algorithms: [1 week] Constant multiplicative factor approximation algorithms – bin packing, vertex cover, Euclidean traveling salesman problem, etc. notions of hardness of approximation.

(b) **Prerequisites:** None

(c) **References:**

1. A. Aho, J. Hopcroft and J. Ullman, The Design and Analysis of Computer Algorithms. A. W. L, International Student Edition, Singapore, 1998.
2. S. Baase, Computer Algorithms: Introduction to Design and Analysis. 2nd ed., Addison- Wesley, California, 1988.
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms. third edition, MIT Press, 2009.
4. E. Horowitz and S. Sahni, Fundamental of Computer Algorithms. Galgotia Pub. /Pitman, New Delhi/London, 1987/1978.
5. K. Mehlhorn, Data Structures and Algorithms. Vol. 1 and Vol. 2, Springer-Verlag, Berlin, 1984.
6. A. Borodin and I. Munro, The Computational Complexity of Algebraic and Numeric Problems. American Elsevier, New York, 1975.
7. D. E. Knuth, The Art of Computer Programming. Vol. 1, 2nd ed., Narosa/Addison-Wesley, New Delhi/London, 1973; Vol. 2: 2nd ed., Addison-Wesley, London, 1981; Vol. 3: Addison-Wesley, London, 1973.

8. S. Winograd, The Arithmetic Complexity of Computation. SIAM, New York, 1980.
9. S. Dasgupta, C. Papadimitriou, U. Vazirani, Algorithms. McGraw Hill India, Chennai.
10. J. Kleinberg and E. Tardos, Algorithm Design. Pearson Education.
11. M. Mitzenmacher and E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press.

Digital Design and Computer Architecture

(a) Information representation: Number systems – binary, octal and hexadecimal numbers, signed number representation, fixed and floating point representations, and arithmetic.

Boolean algebra: Postulates and fundamental theorems, Representation of Boolean functions, canonical forms, fundamental Boolean operations, Karnaugh Maps, Minimizing Boolean functions.

Combinational Logic Design: Logic gates, Adder, Subtractor, Comparator, decoder, Multiplexer, PLA.

Sequential Logic Design: Latches and Flip-Flops, Finite state models for sequential machines, registers, shift register, ripple counters, synchronous counters; state diagrams, characteristics and excitation tables of various memory elements, state minimization for synchronous and asynchronous sequential circuits.

ALU Design: Addition of numbers – carry look-ahead and pre-carry vector approaches, carry propagation-free addition. Multiplication - using ripple carry adders, carry save adders, redundant number system arithmetic, Booth's algorithm. Division - restoring and non-restoring techniques, using repeated multiplication. Floating-point arithmetic, multiplication and addition algorithms. ALU design, instruction formats, addressing modes.

Processor Design: ISA and Microarchitecture design, hardware control unit design, Pipelining, Out of order execution.

Memory Organization: Random and serial access memories, static and dynamic RAMs, ROM. Cache Memory fundamentals.

I/O Organization: Different techniques of addressing I/O devices, data transfer techniques, programmed interrupt, DMA.

(b) **Prerequisites:** None

(c) **References:**

1. Z. Kohavi, Switching and Finite Automata Theory. 2nd ed., McGraw Hill, New York, 1978.
2. E. J. Mcclusky, Logic Design Principles. Prentice Hall International, New York, 1986.
3. John L. Hennessy and David A. Patterson, Computer Architecture: A Quantitative Approach. 2012.
4. John L. Hennessy and David A. Patterson, Computer Organization and Design: The Hardware/Software Interface. 2021.
5. J. P. Hayes, Computer Architecture and Organization. McGraw Hill, New York, 1988.
6. P. P. Choudhury, Computer Organization and Design. Prentice Hall of India, New Delhi, 1994.
7. M. M. Mano, Computer System Architecture. 3rd ed., Prentice Hall of India, New Delhi, 1993.
8. W. Stallings, Computer Organization and Architecture: Principles of Structure and Function. 2nd ed., Macmillan, New York, 1990.

Discrete Mathematics

(a) Combinatorics: **[6 weeks]** Introduction to posets, relations, asymptotic notations, proof techniques, induction on sets and graphs, multinomial theorem, principle of inclusion exclusion; pigeonhole principle; Counting using (in)distinguishable balls and (in)distinguishable bins, functions and counting, Stirling number; Recurrences and ways to solve them, solving recurrences using asymptotic notations. Generating functions. Introduction to Probabilistic methods.

Graph Theory: **[6 weeks]** Graphs and digraphs, isomorphism, connectedness and reachability, representation of graphs, Eulerian paths and circuits in graphs and digraphs, Hamiltonian paths and circuits in graphs and tournaments, trees; Minimum spanning tree, rooted trees and binary trees, planar graphs, Euler's formula, statement of Kuratowski's theorem, dual of a planar graph, independence number and clique number, chromatic number, statement of Four-color theorem, dominating sets and covering sets, matching, network flows

Basic Logic: **[2 weeks]** Introduction to propositional logic – propositions and connectives, syntax; semantics – truth assignments and truth tables, validity and satisfiability, tautology; Adequate set of connectives; Introduction to quantifers - first order logic over simple relational vocabulary.

(b) **Prerequisites:** None

(c) **References:**

1. C. L. Liu, Elements of Discrete Mathematics. McGraw Hill, 1985.
2. D. B. West, Introduction to Graph Theory. Pearson, 2000.
3. R. Diestel, Graph Theory. Springer, 2010.
4. F. Harary, Graph Theory. Narosa Publishing House, 2001.
5. E. Mendelsohn, Introduction to Mathematical Logic. Van-Nostrand, 1979.
6. J. Matousek and J. Nesetril, Invitation to Discrete Mathematics Paperback. Oxford University Press, 2009.
7. D. West, Combinatorial Mathematics. Cambridge University Press, 2021.
8. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press, 1990.

Linear Algebra

(a) Matrices and System of Equations: System of Linear Equations, Row Echelon Form, Matrix Arithmetic, Matrix Algebra, Elementary Matrices, Partitioned Matrices, Determinant and its properties.

Vector Spaces: Definition and Examples, Subspaces, Linear Independence, Basis and Dimension, Change of Basis, Row Space, Column Space, Null space

Inner product spaces: The Euclidean dot product, Orthogonal Subspaces, Least Squares Problems, Orthonormal Sets, The Gram-Schmidt Orthogonalization Process, Orthogonal Polynomials

Linear Transformations: Definition and Examples, Matrix Representations of Linear Transformations, Similarity

Eigenvalues and Eigenvectors: System of Linear Differential Equations, Diagonalization, Hermitian Matrices, Singular Value Decomposition.

Quadratic Forms: Classifiction and characterisations, Optimisation of quadratic forms.

Algorithms: Gaussian Elimination with different Pivoting Strategies; Matrix Norms and Condition Numbers; Orthogonal Transformations; The Eigenvalue Problem; Least Squares Problems.

(b) **Prerequisites:** None

(c) **References:**

1. G. Strang, Introduction to Linear Algebra. 5th ed., Wellesley - Cambridge Press, 2016.
2. S. Axler, Linear Algebra Done Right. 4th ed., Springer, 2025.
3. G. H. Golub and C. F. Van Loan, Matrix Computations. 4th Edition, The Johns Hopkins University Press, Baltimore, 2013.

Introduction to Programming

(a) Introduction to UNIX-like systems.

Basics of C programming: variables, operators, expressions, control flow (sequential, conditional, repetitive, using factorial, Fibonacci, GCD, etc. as examples), basic I/O, arrays, strings, pointers, functions, parameter passing (call by value, call by reference), structures, unions, dynamic memory allocation, multi-dimensional arrays, recursion, preprocessor directives, file I/O.

Linked lists, stacks, queues, circular queues: implementation of basic operations (insertion, deletion, reversal, etc.) and applications (prefix, infix, postfix expressions, polynomial addition and multiplication, sparse matrix multiplication and addition, etc.).

Sorting: insertion, selection, bubble sort, merge sort, quick sort; binary search.

Hashing: hash table implementation, searching, insertion and deletion.

Writing and using multi-file programs and libraries.

Efficient programming.

Memory management in programs: call stack, heap management.

Debugging tools: GDB, Valgrind.

(b) **Prerequisite(s):** None

(c) **References:**

(a) B. W. Kernighan and D. M. Ritchie, The C Programming Language. Prentice Hall, 1980.

(b) B. Gottfried, Programming in C. Schaum Outline Series, 1996.

(c) B. Stroustrup, The C++ Programming Language. 2nd ed., Addison-Wesley, 1995.

(d) C. S. Horstmann, Core Java Volume I – Fundamentals. 11th ed., Prentice Hall, 2018.

(e) J. Bloch, Effective Java. 3rd ed., Addison-Wesley, 2018.

(f) B. W. Kernighan and R. Pike, The Practice of Programming, Addison-Wesley.

(g) B. W. Kernighan and R. Pike, The Unix Programming Environment. Prentice Hall.

Operating Systems

(a) *Introduction:* Basic architectural concepts, interrupt handling, concepts of batch-processing, multiprogramming, time-sharing, real-time operations; Resource Manager view, process view and hierarchical view of an OS.
Memory management: Partitioning, paging, concepts of virtual memory, demand-paging – page replacement algorithms, working set theory, load control, segmentation, segmentation and demand-paging, Cache memory management.
Process management: CPU scheduling – short-term, medium term and long term scheduling, non-preemptive and preemptive algorithms, performance analysis of multiprogramming, multiprocessing and interactive systems; Concurrent processes, precedence graphs, critical section problem - 2-process and n-process software and hardware solutions, semaphores; Classical process co-ordination problems, Producer-consumer problem, Reader-writer problem, Dining philosophers problem, Barber's shop problem, Interprocess communication.
Concurrent Programming: Critical region, conditional critical region, monitors, concurrent languages (eq. concurrent Pascal), communicating sequential process (CSP); Deadlocks: prevention, avoidance, detection and recovery.
Device Management: Scheduling algorithms – FCFS, shortest-seek-time-first, SCAN, C-SCAN, LOOK, C-LOOK algorithms, spooling, spool management algorithm.
Information Management: File concept, file support, directory structures, symbolic file directory, basic file directory, logical file system, physical file system, access methods, file protection, file allocation strategies.
Protection: Goals, policies and mechanisms, domain of protection, access matrix and

its implementation, access lists, capability lists, Lock/Key mechanisms, passwords, dynamic protection scheme, security concepts and public and private keys, RSA encryption and decryption algorithms.
A case study: UNIX OS file system, shell, filters, shell programming, programming with the standard I/O, UNIX system calls.

(b) **Prerequisites:** None

(c) **References:**

1 A. Silberschatz and P. B. Galvin, Operating Systems Concepts. 5th ed., John Wiley and Sons, New York, 1998.

2 J. L. Peterson and A. Silberschatz, Operating Systems Concepts. Addison-Wesley, Reading, Mass., 1987.

3 P. B. Hansen, Operating System Principles, Prentice Hall, Englewood Cliffs, 1980.

4 A. S. Tannenbaum, Modern Operating Systems. Prentice Hall, Englewood Cliffs, 1992.

5 S. E. Madnick and J. J. Donovan, Operating Systems. McGraw Hill, New York, 1974.

Probability and Stochastic Processes

(a) Introduction: Sample space, probabilistic models and axioms, conditional probability, Total probability theorem and Bayes' rule, independence.

Discrete random variables: basics, probability mass functions, functions of random variables, expectation, variance, joint probability mass functions, conditioning and independence, notions of Bernoulli, Binomial, Poisson, Geometric, etc., covariance and correlation, conditional expectation and variance, some introduction to probabilistic methods.

Continuous random variables: basics, probability density functions, cumulative distribution function, normal random variable.

Moments and deviations: Markov's inequality, Chebyshev's inequality, Chernoff bounds

Limit theorems: Weak law of large numbers, central limit theorem, strong law of large numbers (proofs under some restrictive setups, if possible.)

Stochastic processes: Bernoulli and Poisson processes, branching processes. Markov chains, classification of states, ideas of stationary distributions. Introduction to Martingales and stopping times.

Applications to computer science: Balls and bins, birthday paradox, hashing, sorting, random walks, etc.

(b) **Prerequisites:** None

(c) **References:**

1. D. P. Bertsekas and J. N. Tsitsiklis, Introduction to Probability. 2nd ed., Athena Scientific, 2008.
2. W. Feller, An Introduction to Probability and its Applications. Volume I (3rd Edition), Wiley, 2008
3. W. Feller, An Introduction to Probability and its Applications. Volume II (2nd Edition), John Wiley & Sons, Inc, 1971.
4. S. Ross, A First Course in Probability, 11th ed., Academic Press, 2014.
5. N. Alon and J. Spencer, The Probabilistic Method. 4th ed., Wiley-Blackwell, 2016.
6. M. Mitzenmacher and E. Upfal, Probability and Computing. 2nd ed., Cambridge University Press, 2017.

Statistical Methods

(a) Review of Descriptive statistics: Representation of Data (collection, tabulation and diagrammatic representation of different types of univariate and bivariate data); Measures of Central Tendency (Mean, Median, Quartiles and Percentiles, Trimmed Mean, Mode); Measures of Dispersion (Range, Variance and Standard Deviation, Mean Deviation, Quartile Deviation), Measures of Shape (Skewness, Kurtosis), Contingency Table

Correlation: Pearson's product-moment correlation coefficient, Spearman's rank correlation, Kendall's tau statistic.

Regression: Simple and multiple linear regression, Method of least squares.

Estimation: Method of moments, Method of maximum likelihood estimation.

Hypothesis Testing: Concept of null and alternative hypotheses, Acceptance and critical regions, Probabilities of Type I and Type II errors, Level and Power of a test,

Most powerful tests, Tests for mean and variance of a normal distribution, Randomized test, Tests for parameters of Binomial and Poisson distributions. Concept of P-value, Z-test, paired and unpaired t-test, Likelihood ratio tests, χ^2 test for categorical data.

Interval Estimation: Interval estimation and its connection with hypothesis testing. Interval estimation for parameters of normal, binomial and Poisson distributions.

ANOVA: Fixed effect model for one-way classified data, two-way classified data with one observation per cell (if time permits).

Bayesian Estimation and Bayesian Statistical Testing

(b) **Prerequisite:** None.

(c) **References:**

1. C. E. Croxton and D. J. Cowden, Applied General Statistics, Pitman Publishing, 3rd ed., 1968.
2. M. J. Schervish, Theory of Statistics. Springer Series in Statistics, 1995.
3. D. Montgomery and G. C. Runge, Applied Statistics and Probability for Engineers, Wiley, 1994.
4. A. M. Goon, M. Gupta and B. Dasgupta, Fundamentals of Statistics. Vol I, II, III and IV, World Press.

Theory of Computation

(a) Automata and Languages: Finite automata, regular languages, regular expressions, deterministic and non-deterministic finite automata, minimization of finite automata, closure properties, Kleene's Theorem, Pumping Lemma and its application, Myhill-Nerode theorem and its uses; Context-free grammars, context-free languages, Chomsky normal form, closure properties, Pumping Lemma for context-free languages, push down automata.

Computability: Computable functions, primitive and recursive functions, universality, halting problem, recursive and recursively enumerable sets, parameter theorem, diagonalisation, reducibility, Rice's Theorem and its applications. Turing machines and variants; Equivalence of different models of computation and Church-Turing thesis.

Introduction to Complexity: Discussions on time and space complexities; P and NP, NP-completeness, Cook's Theorem, other NP-Complete problems; PSPACE; polynomial hierarchy.

(b) **Prerequisites:** Discrete Mathematics

(c) **References:**

1. N. J. Cutland, Computability: An Introduction to Recursive Function Theory. Cambridge University Press, London, 1980.
2. M. D. Davis, R. Sigal and E. J. Weyuker, Complexity, Computability and Languages. Academic Press, New York, 1994.
3. J. E. Hopcroft, J. D. Ullman and R. Motwani, Introduction to Automata Theory, Languages and Computation. Addison-Wesley, California, 2001.
4. H. R. Lewis and C. H. Papadimitriou, Elements of the Theory of Computation. Prentice Hall, Englewood Cliffs, 1981.
5. M. Sipser, Introduction to the Theory of Computation. PWS Pub. Co., New York, 1999.
6. M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to The Theory of NP- Completeness. Freeman, New York, 1979.

4.2 Elective courses

Advanced Algorithms

(a) The course has two parts – Randomized Algorithms and Approximation Algorithms. The instructor can choose from the broad list given against the two parts. The course can, if needed, start with a brief introduction to (i) NP completeness, strong NP completeness; (ii) Linear programs – strong and weak duality.

Randomized Algorithms: The syllabus consists of several tools from the theory of randomization and its application to several branches of computer science like graphs, geometry, discrepancy, metric embedding, streaming, random graphs, etc.

Topics: Applications (can be chosen from the following list):

(1) Computational Geometry – Randomized incremental construction; backward analysis; random sampling – VC dimension, epsilon-nets; convex polytopes; geometric data structures

(2) Streaming algorithms – estimating the number of distinct elements; estimating frequency moments; geometric streams and core-sets; metric stream and clustering; graph streams; proving lower bounds from communication complexity;

(3) Metric embedding and dimension reduction – Johnson-Lindenstrauss lemma, Noga's lower bound, Bourgain embedding, Bartal's result

(4) Discrepancy – Combinatorial discrepancy for set systems; VC dimension and discrepancy;

(5) Probabilistic methods – Linearity of expectation; alteration; second moment; Lovasz local lemma – existential and constructive proofs; derandomization techniques; expander graphs; random graphs

(6) Miscellaneous topics – Data structures; Hashing and its variants; Primality testing; approximate counting; graph algorithms; randomized rounding; etc.

Approximation Algorithms:

(1) Greedy algorithms and local search – k-center problem; TSP; minimum degree spanning tree;

(2) Rounding and Dynamic Programming – knapsack; bin-packing; scheduling jobs on identical parallel machines

(3) Deterministic rounding of linear programs – solving large linear programs in polynomial time via ellipsoid method; prize collecting Steiner tree; uncapacitated facility location

(4) Random Sampling and randomized rounding of linear programs – derandomization; linear and non-linear randomized rounding; integrality gap; MAX-CUT, MAX-SAT; prize collecting Steiner tree; uncapacitated facility location; integer multicommodity flows

(5) Semidefinite programming – introduction; randomized rounding in semidefinite programming; finding large cuts; approximating quadratic programs

(6) Primal Dual method – introduction; feedback vertex set; shortest s-t path; Lagrangean relaxation and k-median problem

(7) Cuts and metrics – multiway cut, multiple cut, balanced cut, probabilistic approximation of metrics by tree metrics; spreading metrics, tree metrics and linear arrangement

(b) **Prerequisites:** Design and Analysis of Algorithms, Probability and Stochastic Processes

(c) **References:**

1. M. Mitzenmacher and E. Upfal, Probability and Computing – Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, 2005.
2. R. Motwani and P. Raghavan, Randomized Algorithms. Cambridge University Press, 2004.
3. N. Alon and J. H. Spencer, The Probabilistic Method. Wiley, 2008.
4. K. Mulmuley, Computational Geometry - An Introduction through Randomized Algorithms. Prentice Hall, 1994.
5. J. Matousek, Geometric Discrepancy: An Illustrated Guide. Springer.
6. S. Muthukrishnan, Data Streams: Algorithms and Applications. Now Publishers, Foundations & Trends in Theoretical Computer Science.
7. B. Chazelle, The Discrepancy Method: Randomness and Computation. Cambridge University Press.
8. V. V. Vazirani, Approximation Algorithms. Springer, 2003.
9. D. P. Williamson and D. B. Shmoys. The Design of Approximation Algorithms. Cambridge University Press.
10. S. Har-Peled, Geometric Approximation Algoithms. American Mathematical Society.
11. L. C. Lau, R. Ravi and M. Singh, Iterative Methods in Combinatorial Optimization. Cambridge University Press, 2011.

Advanced Computer Architecture

(a) The instructor may select only some of the following topics, and include other topics of current interest.

Brief Review: Von Neumann model, Processor components, Instruction Set Architecture (ISA), Micro-architecture design fundamentals (single and multi-cycle)

Modern Processor Design: Pipelining, Out of Order Execution, Branch Prediction, Precise Exceptions, Decoupled Access Execute, Prefetching, Speculative Execution.

Dataflow, Superscalar, VLIW architectures

SIMD Architectures: GPU Architectures, Systolic Array processing, GPU programming.

Memory: Memory System Design, Memory Controllers, Emerging Memory Technologies.

Advanced Caches: Cache Design and Management, Cache Controllers.

MultiProcessors: Design, Memory Ordering, Multiprocessor programming.

Cache Coherence: Protocols and Implementation.

Interconnects and OnChip Networks:

(b) **Prerequisites:** Digital Design and Computer Architecture, Operating Systems

(c) **References:**

1. John L. Hennessy and David A. Patterson, Computer Architecture, A Quantitative Approach, 6th ed., Morgan Kaufmann, 2017.
2. Smruti R. Sarangi, Basic Computer Architecture. 1st ed., WhiteFalcon, 2021.
3. Smruti R. Sarangi, Next-Gen Computer Architecture. 1st ed., WhiteFalcon, 2023.

Advanced Computer Networks

(a) Introduction: Overview and motivation, Characteristics of communication networks, Protocol design issues, Protocol stacks and layering

Transmission fundamentals: Analog and digital transmissions, Different transmission media, Different transmission impairments, Different modulation techniques, Channel capacity, Basic concept of spread spectrum and frequency hopping, Asynchronous and synchronous transmission, Multiplexing.

Communication networks: Introduction to LANs, MANs, and WANs; Switching techniques: Circuit-switching and Packet-switching; Topological design of a network, LAN topologies, Ethernet, Performance of Ethernet, Repeaters and bridges, Asynchronous Transfer Mode.

Queuing theory: Introduction to queuing theory and systems, Elementary queuing systems, Network performance analysis using queuing systems.

Data link layer: Services and design issues, Framing techniques, Error detection and correction, Flow control: Stop-and-wait and Sliding window; Performance analysis of stop-and-wait and sliding window protocols, MAC Protocols: ALOHA, CSMA, CSMA/CD, Collision free protocols, Limited contention protocol; Wireless LAN protocols: MACA, CSMA/CA; Comparative analysis of different MAC protocols.

Internetworking and IP: Design issues, Organization of the subset, Routing: Static and dynamic routing, Shortest path routing, Flooding, Unicast and multicast routing, Distance-vector routing, Linkstate routing; Congestion control: choke packets, leaky bucket, token bucket; IP protocol, IPV4, IPV6, IP addressing, CIDR, NAT, Internet control protocols: ICMP, ARP, RARP.

Transport and Reliable Delivery: Design issues, Port and socket, Connection establishment and release, TCP, UDP, TCP congestion control, TCP timer management, RPC.

(b) **Prerequisites:** Digital Design and Computer Architecture, Computer Networks

(c) **References:**

1. L. L. Peterson and B. S. Davie, Computer Networks: A Systems Approach. Morgan Kaufmann Publishers, 2007.
2. A. S. Tanenbaum, Computer Networks. 5th ed., Prentice Hall, 2010.
3. W. Stallings, Data and Computer Communications. 10th ed., Prentice Hall, 2017.
4. D. P. Bertsekas and R. G. Gallagher, Data Networks. 2nd ed., Prentice Hall, 1992.
5. W. R. Stevens, Unix Network Programming. PHI, 2009.
6. W. R. Stevens, TCP/IP Illustrated. Volume 1: The Protocols, Addison-Wesley Professional.
7. J. F. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach. 8th ed., Pearson Education, 2022.
8. D. Gross and C. M. Harris, Fundamentals of Queueing Theory, Wiley, 2008.

Advanced Logic and Automata Theory

(a) Monadic second order logic: syntax, semantics, truth, definability, relationship between logic and languages, Büchi-Elgot-Trakhtenbrot theorem.

Automata on infinite words: Büchi automata, closure properties, Müller automata, Rabin automata, Streett automata, determinization, decision problems, Linear temporal logic and Büchi automata, Finite and infinite tree automata, closure properties, decision problems, complementation problem for automata on infinite trees, alternation, Rabin's theorem.

Modal mu-calculus: syntax, semantics, truth, finite model property, decidability, parity games, model checking problem, memoryless determinacy, algorithmic issues, bisimulation, Janin/Walukiewicz theorem.

(b) **Prerequisites:** Discrete Mathematics, Automata Theory, Languages and Computation, Logic for Computer Science

(c) **References:**

1. B. Khoussainov and A. Nerode, Automata Theory and its Applications. Springer, 2001.
2. E. Grädel, W. Thomas and T. Wilke (Eds.), Automata, Logics, and Infinite Games. LNCS 2500, Springer, 2002.
3. D. Perrin and J. -E. Pin, Infinite Words: Automata, Semigroups, Logic and Games. Elsevier, 2004.
4. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree Automata Techniques and Applications. 2008. (Open source: http://tata.gforge.inria.fr)
5. P. Blackburn, M. de Rijke and Y . V enema, Modal Logic. Cambridge University Press, 2001.
6. Y. Venema, Lectures on the Modal μ-calculus. 2012. (Source: https://staff.science.uva.nl/y.venema/teaching/ml/mu/mu20121116.pdf)

Advanced Operating Systems

(a) The instructor may select only some of the following topics, and include other topics of current interest

Operating systems structures: monolithic, microkernel, ExoKernel, multi kernel.

System calls, interrupts, exceptions.

Symmetric Multi Processor (SMP) systems: scheduling, load balancing, load sharing, process migration; synchronisation in SMP systems.

Interprocess communication: signals, message passing.

Naming in distributed systems: directory services, DNS.

Remote Procedure Calls (RPC): model, stub generation, server management, parameter passing, call semantics, communication protocols, client-server binding, exception handling, security, optimization.

Distributed shared memory: architecture, consistency model, replacement strategy, thrashing, coherence.

File systems: Fast File System (FFS), Virtual File System (VFS), log-structured file systems and journalling, RAID; Distributed File Systems (DFS), stateless and stateful DFS, Andrew File System (AFS), Network File Systems (NFS).

Virtualisation: introduction, nested virtualisation, case study.

Device drivers.

Fault tolerance.

Clusters, cloud computing.

Protection and security.

Projects and real systems implementations

(b) **Prerequisites:** Digital Design and Computer Architecture, Operating Systems

(c) **References:**

1. T. Anderson and M. Dahlin, Operating Systems Principles and Practice. 2nd ed., Recursive Books, 2014.
2. D. P. Bovet and M. Cesati, Understanding the Linux Kernel. 3rd ed., O'Reilly, 2008.
3. R. Love, Linux Kernel Development. 3rd ed., Addison-Wesley Professional, 2010.
4. J. Corbet, A. Rubini and G. Kroah-Hartman, Linux Device Drivers. 3rd ed., O'Reilly, 2005.

Algorithms for Big Data

(a) Review of Linear Algebra and Probability

Sketching and Streaming algorithms for basic statistics: Distinct elements, heavy hitters, frequency moments, p-stable sketches.

Dimension Reduction: Johnson Lindenstrauss lemma, lower bounds and impossibility results

Graph stream algorithms: connectivity, cut/spectral sparsifiers, spanners, matching, graph sketching.

Lower bounds for Sketching and Streaming.

Communication complexity: Equality, Index and Set-Disjointness.

Locality Sensitive Hashing: similarity estimation, approximate nearest neighbor search, data dependent hashing.

Fast Approximate Numerical Linear Algebra: matrix multiplication, low-rank approximation, subspace embeddings, least squares regression

(b) **Prerequisites:** Probability and Stochastic Processes, Linear Algebra.

(c) **References:**

1. A. Blum, J. Hopcroft, and R. Kannan, Foundations of Data Science, 2020. (Source: https://www.cs.cornell.edu/jeh/book.pdf)
2. D. P. Bertsekas and J. N. Tsitsiklis, Introduction to Probability, 2nd ed., Athena Scientific, 2008.
3. M. Mitzenmacher and E. Upfal, Probability and Computing, 2nd ed., Cambridge University Press, 2017.
4. N. Alon and J. Spencer, The Probabilistic Method. 4th ed., Wiley-Blackwell, 2016.

Algorithms for VLSI Design

(a) Introduction: VLSI design, Design domains, design styles and parameters, popular technologies.

High level synthesis: Introduction to Synthesis, Design representations and transformations, partitioning, scheduling, allocation algorithms for high level synthesis with C-to-logic as a case study.

Computational Boolean Algebra: Boolean algebra concepts, Minimization using k-map, Minimization using Tabular method, FSM-based design minimization, Boolean Algebra using BDD and SAT.

Logic Synthesis: Two-level logic synthesis and Multi-level logic synthesis, The Reduce-Expand-Irredundant Optimization Loop, Multilevel Logic and the Boolean Network Model, Algebraic Model for Factoring, Algebraic Division, Role of Kernels and Co-Kernels in Factoring, Mulitlevel Logic and Divisor Extraction.

Floor-planning and Placement: Floor-planning concepts, Shape Functions and Floorplan sizing, Placement, Wirelength Estimation, Types of placement problems and algorithms.

Technology Mapping: Graph covering and Technology mapping, Tree covering by Dynamic programming, Decomposition, Delay optimization.

Routing: Local routing, Global routing, Algorithms for routing.

Timing Analysis: Logic-level timing, Interconnect timing, Static Timing Analysis.

(b) **Prerequisites:** Digital Design and Computer Architecture, Design and Analysis of Algorithms

(c) **References:**

1. D. Pucknell and K. Eshraghian, Basic Principles of VLSI Design. Prentice Hall, Englewood Cliffs, 1985.
2. G. D. Hachtel and F. Somenzi, Logic Synthesis and Verification Algorithms. Springer, 1996.
3. C. Mead and L. Conway, Introduction to VLSI Systems. Addison-Wesley, Reading, Mass., 1980.
4. R. K. Brayton, G. D. Hachtel, C. T. McMullen, A. L. Sangiovanni-Vincentelli, Logic Minimization for VLSI Synthesis. Kluwer Academic Publishers, Boston, 1984.
5. D. Gajski, N. Dutt, A. C-H Wu, and S. Y-L Lin, High Level Synthesis: Introduction to Chip and System Design. Kluwer Academic, Boston, 1992.
6. N. Sherwani, Algorithms for VLSI Physical Design Automation. Kluwer Academic, Boston, 1999.
7. S. Sutherland, S. Davidmann, and P. Flake, SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling. Springer-Verlag, 2010.

Artificial Intelligence

(a) *Introduction:* A brief history, applications, and techniques of artificial intelligence.

State Space and Search: State space, state space search, defining a problem as a state space search problem; Uniform or blind search, breadth first, depth first, uniform cost search, depth first iterative deepening, constraint satisfaction problems, cryptarithmetic problems, means-ends analysis.

Informed Search: Informed search, heuristic, best first search, admissibility, monotonicity, informedness, OR graph, algorithm A*, iterative deepening A*, AND-OR graphs, AO* algorithm.

Heuristic Search: Local search and optimization, state space landscape, hill climbing approach, steepest ascent hill climbing, local and global maxima, ridges, plateau, simulated annealing, local beam search.

Game Playing: Game playing, minimax search, alpha-beta pruning.

Knowledge Representation: Propositional and predicate logic, rule based, associative or semantic networks, conceptual graphs, frame, scripts.

Learning: Basic concepts of supervised learning, unsupervised learning, reinforcement learning, semi-supervised learning, self-supervised learning, active learning, incremental learning.

Reasoning: Monotonic reasoning, non-monotonic reasoning, logical reasoning, theorem proving, soundness, completeness, validity, consequence, consistency; Problem solving, truth table, semantic tableaux, resolution principle; Reasoning with uncertainty, truth maintenance systems, probabilistic reasoning, approximate reasoning, fuzzy reasoning.

Expert Systems: Definitions, features, structure, rule based expert system, forward chaining, backward chaining.

Planning: Plan space and partial order planning, planning algorithms, plan generating systems.

AI Tools and Applications: Fuzzy logic, artificial neural network, genetic algorithm, rough sets, decision tree; Applications to natural language processing, vision, speech understanding, verification, MYCIN; Languages for AI tools.

(b) **Prerequisites:** None.

(c) **References:**

1. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Prentice Hall-Pearson, 2016.
2. N. J. Nilsson, Artificial Intelligence: A New Synthesis. Morgan Kauffman, 1998.
3. E. Rich, K. Knight, Artificial Intelligence. McGraw-Hill, 1991.
4. G. F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Pearson, 6th ed., 2008.

Coding and Information Theory

(a) Introduction: Basic definitions: codes, dimension, distance, rate, error correction, error detection.

Entropy and its application

Shannon's Theorems: Noiseless coding; Noisy Coding; Shannon Capacity;

Gilbert Varshamov bound; Singleton bound; Plotkin bound

Linear Codes: Properties of linear codes; Hamming codes; Efficient decoding of Hamming codes; Dual of a linear code

Algebraic codes: Reed-Solomon codes; Concatenated codes; BCH codes; Reed-Muller codes; Hadamard codes; Dual BCH codes.

Algorithmic issues in coding: Decoding Reed-Solomon Codes; Decoding Concatenated Codes

List Decoding: List decoding; Johnson bound; List decoding capacity; List decoding from random errors. List decoding of Reed-Solomon codes.

Advanced Topics: Graph Theoretic Codes; Locality in coding: Locally decodable codes, locally testable codes; codes and derandomization.

(b) **Prerequisites:** Probability and Stochastic Processes, Linear Algebra.

(c) **References:**

1. V. Guruswami, A. Rudra, and M. Sudan, Essential Coding Theory, 2025. (Source: https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book)
2. F. J. MacWilliams and N. J. A. Sloane, Theory of Error-Correcting Codes. North Holland Publishing Co., 1977.
3. J. H. van Lint, Introduction to Coding Theory. Springer, 1973.
4. V. S. Pleaa and W. C. Huffman (Eds.), Handbook of Coding Theory, I and II, Elsevier, 1998.

Computability Theory

(a) Recursive functions; Turing machines; Coding; Self-reference; Universal machines.

Computable and computably enumerable sets; Diagonalization; Undecidable sets; Uniformity; Many-one reducibility,; Simple sets; Fixed-point Theorem.

Arithmetical Hierarchy

Relativized computation and Turing degrees: Turing reducibility; Jump; Incomparable degrees; Priority method

Decidable and undecidable theories

(b) **Prerequisites:** Theory of Computation

(c) **References:**

1. R.I. Soare, Turing Computability. Springer, 2016.
2. B. Cooper, Computability Theory. Chapman & Hall/CRC, 2004.
3. R. I. Soare, Recursively Enumerable Sets and Degrees. Springer, 1987.
4. R. G. Downey, D. R. Hirschfeldt, Algorithmic Randomness and Complexity. Springer, 2010.
5. A. Nies, Computability and Randomness. Oxford University Press, 2009.

Computational Algebra and Number Theory

(a) Number Theoretic Algorithms: GCD and its extensions, randomized and deterministic primality testing algorithms

Polynomial Manipulations: Polynomial evaluation and interpolation, GCD algorithm, factoring polynomials over finite fields and integers

Applications in Cryptography

Matrix Computations: Asymptotically fast matrix multiplication algorithms

Solving Systems of Linear and Non-linear Equations Grobner basis and its application.

Computer Algebra Systems: Issues of data representation – sparse, dense, canonical, normal; Representations of polynomials, matrices and series;

Algebraic Complexity Theory: Uniform and non-uniform models, straight-line and branching programs; Survey of lower bound results for polynomial, and Polynomial Identity Testing (PIT)

(b) **Prerequisites:** Design and Analysis of Algorithms

(c) **References:**

1. J. V. zur Gathen and J. Gerhard, Modem Computer Algebra. Cambridge University Press, London, 1999.
2. V. Shoup, A Computational Introduction to Number Theory and Algebra. Cambridge University Press
3. D. E. Knuth, The Art of Computer Programming; Semi-Numerical Algorithms. Vol. 2, 3rd ed., Addison-Wesley Publishing Company, Reading, Mass., 1997.

Computational Complexity

(a) Introduction: Review of machine models, Turing machines and its variants, reduction between problems and completeness, time and space complexity classes

Structural Results: Time and space hierarchy theorems, polynomial hierarchy, Ladner's theorem, relativization, Savitch's theorem

Circuit Complexity: Circuits and non-uniform models of computation, parallel computation and NC, P-completeness, circuit lower bounds, AC^0 and parity not in AC^0, Hastad's Switching Lemma, introduction to natural proof barrier

Random Computation: Probabilistic computation and complexity classes and their relations with other complexity classes, BPP=P?

Interactive proofs: Introduction to Arthur-Merlin Games, IP=PSPACE, multiprover interactive proofs, introduction to PCP theorem

Complexity of counting: Complexity of optimization problems and counting classes, Toda's theorem, inapproximability, application of PCP theorem to inapproximability and introduction to unique games conjecture

Cryptography: Public-key cryptosystems, one-way functions, trapdoor-functions, application to derandomization

(b) **Prerequisites:** Discrete Mathematics, Design and Analysis of Algorithms

(c) **References:**

1. J. Balcazar, J. Diaz, and J. Gabarro, Structural Complexity – I and II. Springer-Verlag, Berlin, 1988/1990.
2. D. P. Bovet and P. Crescenzi, Introduction to the Theory of Complexity. Prentice Hall, Englewood Cliffs, 1994.
3. M. Sipser, Introduction to Theory of Computation, PWS Publishing, New York, 1999.

4. C. H. Papadimitriou, Computational Complexity. Addison-Wesley, 1994.
5. J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory. Languages and Computation, Addison-Wesley, 1979.
6. O. Goldreich, Lecture Notes on Computational Complexity. Tel Aviv University, 1999.
7. S. Arora and B. Barak, Computational Complexity: A Modern Approach. Cambridge University Press, 2009.

Computational Finance

(a) Basic Concepts: (i) Arbitrage, Principle of no arbitrage, Law of one price; Frictionless / Efficient market, Transaction cost, Contingent contracts, Concept of complete market (ii) Time value of money, discounting: deterministic and stochastic; Martingale, Risk neutral valuation, Equivalent martingale measure; (iii) Mean Variance utility / Normal distributed returns; Capital Asset pricing Model (CAPM), Extensions, test for efficiency

Contracts: Forwards, Futures, Options (Call, Put, European, American, Exotics), Combinations; Risk neutral portfolio construction

Valuation of contracts in discrete time models. Computation using Binomial tree. Link with the continuous time model: Brownian motion, Black Scholes option pricing and hedging.

High frequency trading (Machine learning, Neural networks), Algorithmic trading

(b) **Prerequisites:** None

(c) **References:**

1. S. R. Pliska, Introduction to Mathematical Finance: Discrete Time Models, Wiley, 1997.
2. J. C. Hull, Options, Futures, and Other Derivatives, 11th ed., Pearson Education, 2022.
3. E. Z. Prisman, Pricing Derivative Securities: An Interactive, Dynamic Environment with Maple V and Matlab, Academic Press Inc, 2000.
4. B. Oksendal, Stochastic Differential Equations: An Introduction with Applications, Springer, 2014.

Computational Game Theory

(a) Zero Sum Games Zero Sum Games and Pure Strategies, saddle point, Mixed Strategies, Donimant Strategies, and Equilibria

Computing in Zero Sum Games: Minimax theorem, LP duality, Statement of the strong duality theorem, Weak duality theorem and its proof, Polyhedral set, Farkas Lemma and its proof, Representation of Polyhedral set, completely mixed games, proof of Minimax theorem

Non-cooperative Games: Utility Theory, Nash equilibria, Existence of Nash equilibrium via Kakutani's fixed point theorem, Sperner's Lemma, Brouwer's fixed point theorem, and existence of Nash equilibria.

Computational Complexity of Nash Equilibrium: Linear Complementary Problem (LCP), Computing Nash Equilibrium, Complexity of Computing Nash Equilibrium

Refining Nash: Games with Turns and Subgame Perfect Equilibrium. Nash Equilibrium without Full Information: Bayesian Games

Cooperative Games Markets and Their Algorithmic Issues

Transferable Utility Games: Stable sets: exsistence, uniqueness, Core: LP formulation, Existence of Core, Bondareva-Shapley Theorem, Banzhaf index, Shapley Value, Least Core, Nucleolus, Bargaining Set.

Non-transferable Utility Games: Hedonic games and computational complexity of finding core.

Stable Matching: properties, core and Incentive Compatible Mechanism Design

Introduction to Mechanism Design: Social Choice: Impossibility results, Mechanisms with Money: VCG Mechanism

Equilibrium Computation for Two-Player Games in Strategic and Extensive Form

Learning, Regret Minimization, and Equilibria: External Regret Minimization, Generic Reduction from External to Swap Regret, Partial Information Model

Online Mechanisms

(b) **Prerequisites:** None

(c) **References:**

1. N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani (eds). Algorithmic Game Theory, Cambridge University Press, 2007.

2. K. H. Erickson, Game Theory: A Simple Introduction. Createspace Independent Pub, 2013.
3. G. Chalkiadakis, E. Elkind, and M. Wooldridge, Computational Aspects of Cooperative Game Theory. Springer, 2012.
4. Y. Narahari, Lecture Notes on Game Theory, IISc.

Computational Geometry

(a) Preliminaries: Basic Euclidean geometry

Grids and Hulls: Fixed-radius near neighbors, convex hull algorithms, dominance and applications.

Linear Programming: Half-plane intersection and randomized LP, backwards analysis, applications of low-dimensional LP.

Intersections and Triangulation: Plane-sweep line segment intersection, triangulation of monotone subdivisions, plane-sweep triangulation of simple polygons, art gallery problems.

Point Location: Trapezoidal decompositions and analysis, history DAGs.

Voronoi Diagrams: Basic definitions and properties, Fortune's algorithm.

Geometric Data Structures: kd-trees, range trees and orthogonal range searching, segment trees, ham-sandwich cut tree, simplex range searching.

Delaunay Triangulations: Point set triangulations, basic definition and properties, randomized incremental algorithm and analysis.

Arrangements and Duality: Point/line duality, incremental construction of arrangements and the zone-theorem, applications.

Geometric Approximation: Dudley's theorem and applications, well-separated pair decompositions and geometric spanners, VC dimension, epsilon-nets and epsilon-approximations, Polynomial Time Approximation Schemes (Shifting Strategy of Hochbaum and Maass)

(b) **Prerequisites:** None

(c) **References:**

1. F. P. Preparata and M. I. Shamos, Computational Geometry: An Introduction. Springer-Verlag, Berlin, 1985.

2. M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, Computational Geometry: Algorithms and Applications. 3rd ed., Springer-Verlag, 2008.
3. S. H. Peled, Geometric Approximation Algorithms. American Mathematical Society, 2010.
4. J. Matousek, Lectures on Discrete Geometry. Springer, 2002.
5. D. Mount, Lecture notes on Computational Geometry. CMSC 754, Stanford.

Computational Learning Theory

(a) Introduction to the PAC learning framework

Sample complexity: basic bounds and the Occam's Razor principle

Consistent learning algorithms

VC-dimension and growth functions, and sample complexity upper and lower bounds via VC-dimension

PAC learnability of common concept classes (intervals, halfspaces, conjunctions, etc.)

Boosting and weak learnability (AdaBoost)

Hardness of learning under cryptographic assumptions

Exact learning with membership and equivalence queries

Learning DNF, monotone DNF, and DFA in the query model

Statistical query model and its limitations

Learning real-valued functions

Rademacher complexity and uniform convergence

Linear regression and generalized linear models in the agnostic setting

Online learning and mistake-bounded frameworks

Perceptron algorithm, Winnow algorithm, and mistake bounds

Experts problem, multiplicative-weights update and applications of multiplicative weights (boosting, game theory, regret minimization)

(b) **Prerequisites:** Design and Analysis of Algorithms, Probability and Stochastic Processes

(c) **References:**

1. M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning. 2nd ed., MIT Press, 2018.
2. M. J. Kearns and U. Vazirani, An Introduction to Computational Learning Theory. MIT Press, 1994.
3. S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

Computational Molecular Biology and Bioinformatics

(a) Sequence Alignment: Global alignments (Needleman-Wunsch), Local alignments (Smith-Waterman), Semi-global alignments, Multiple sequence alignments, k-mer based methods (BLAST), Advanced alignment methods (Gibbs sampling, suffix trees).

Genome Analysis: Genetic mapping, Physical mapping, Recombinant DNA and Sequencing technologies, Whole-genome shotgun (Arachne) and clone-by-clone sequencing (Walking), Population genomics, SNP discovery, discovery of copy number variation and other structural variations, disease mapping, Gene recognition (Genscan) and cross-annotation (Rosetta).

Transcriptome Analysis: Transcriptional regulation, microarray technology, expression clustering, DNA binding sites, location analysis, regulatory motif prediction, ribozymes, non-coding RNAs, RNA secondary structure.

Evolution: RNA world, Phylogenetic analysis.

Protein Structure Analysis: Introduction to protein structure, Protein motifs - hidden Markov models for MSA, prediction (coiled-coil and beta-helix motifs), Threading.

Protein Dynamics: Molecular mechanics, Side-chain packing, Drug discovery tools, Lattice models for protein folding, Simulating virus shell assembly.

Molecular Systems Biology: Systems Biology and analysis of biochemical pathways, Kinetic modeling, Network based modeling, Flux balance analysis, Omics and Multiomics Data Analysis.

Disease Modelling

(b) **Prerequisite:** Design and Analysis of Algorithms.

(c) **References:**

1. C. Setubal and J. Meidanis: Introduction to Computational Molecular Biology, PWS Publishing Company, Boston, 1997.

2. P. A. Pevzner: Computational Molecular Biology – An Algorithmic Approach, MIT Press, 2000.
3. R. Durbin, S. R. Eddy, A. Krogh and G. Mitchison: Biological Sequence Analysis – Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.
4. D. Gusfield: Algorithms on Strings, Trees, and Sequences, Cambridge University Press, USA, 1997.
5. H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore and J. Darnell: Molecular Cell Biology, W. H. Freeman, USA, 2000.
6. C.-I. Branden, J. Tooze: Introduction to Protein Structure, Garland Publishing, 1998.
7. A. Kowald, C.hristoph Wierling, E. Klipp, and W. Liebermeister: Systems Biology, Wiley-VCH, 2016.
8. B.O. Palsson: Systems Biology – Constraint based Reconstruction and Analysis, Cambridge University Press, 2015.

Computational Topology

(a) Topics:

- Basics of planar graphs: Graph minor, subdivision, and Kuratowski's and Wagner's theorems
- Efficient algorithms for planar graphs: Testing planarity, graph drawing, MST algorithm, separator theorem and its applications, graph coloring, computing cuts and related problems
- Introduction to classification of surfaces, and graphs embedded on surfaces
- Introduction to homotopy and homology
- Algorithms for graphs embedded on a surface: Computing cut loci, homotopy test, minimum basis computations and related topics
- Introduction to computational 3-manifold theory
- Complexity issues in high dimensional computational topology
- Persistent homology and its applications
- Distance to a measure
- Discrete Morse Theory

(b) **Prerequisites:** Design and Analysis of Algorithms, Probability and Stochastic Processes

(c) **References:**

1. Ê. C. de Verdíre, Algorithms for embedded graphs. (Source: http://monge.univ-mlv.fr/~colinde/cours/all-algo-embedded-graphs.pdf)
2. H. Edelsbrunner and J. Hared, Computational Topology: An Introduction. American Mathematical Society, 2009.
3. B. Mohar and C. Thomassen, Graphs on Surfaces. Johns Hopkins University Press, 2001.
4. J. Hass, J. C. Lagarias, and N. Pippenger, The Computational Complexity of Knot and Link Problems. Journal of the ACM, 46(2), pp. 185-211, 1999.
5. F. Lazarus and A. de Mesmay, Lecture Notes on Computational Topology.
6. J.-D. Boissonnat, F. Chazal, and M. Yvinec, Geometric and Topological Inference. (Source: https://geometrica.saclay.inria.fr/team/Fred.Chazal/papers/CGLcourseNotes/main.pdf)

Computer Graphics

(a) Overview of Graphics Systems: displays, input devices, hard copy devices, GPU, graphics software, graphics programming language, e.g. openGL

Line drawing algorithms, circle and ellipse drawing algorithms, polygon filling, edge based fill algorithms, seed fill algorithms

2D and 3D camera geometry, Affine and Projective transformations, Orthographic and Perspective view transformations, object to image projection, pin-hole camera model, 3D scene reconstruction, epipolar geometry

2D and 3D clipping, subdivision line-clipping algorithms, line clipping for convex boundaries. Sutherland-Hodgman algorithm, Liang-Barsky algorithm

Hidden line and hidden surfaces algorithms, ray tracing and z-buffer algorithm, Floating horizon algorithm, list priority and backface culling algorithms

2D and 3D object representation and visualization, Bezier and B-Spline curves and surfaces, 2D and 3D surface mesh representation and drawing, sweep representations, constructive solid geometry methods, Octrees, BSP trees, Fractal geometry methods, Visualization of datasets - visual representations for scalar, vector, and tensor fields

Different colour representations, transformation between colour models, halftoning

Rendering, Illumination models, Gouraud shading, Phong shading, transparency, shadows, image and texture mapping and synthesis, Radiosity lighting model

Raster animations, key frame systems, inbetweening, morphing, motion and pose interpolation and extrapolation

Graphical user interface and interactive input methods, interactive picture construction techniques, virtual reality environments.

Projects and Assignments: At least two assignments and one class project, assignments should include implementation of graphics algorithm using a programming language

(b) **Prerequisites:** None

(c) **References:**

1. P. Shirley, M. Ashikhmin, and S. Marschner, Fundamentals of Computer Graphics. CRC Press, 4th ed., 2016.
2. J. F. Hughes, A. V. Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, and K. Akeley, Computer Graphics: Principles and Practice. Pearson Education, 3rd ed., 2014.
3. D. D. Hearn, M. P. Baker, and W. Carithers, Computer Graphics with Open GL, 4th ed., Pearson Education, 2014.

Computer Vision

(a) Image processing to computer vision, computer vision system, applications.

Camera Model: Coordinate systems in computer vision; Pinhole camera model, perspective projection; Homogeneous coordinates, PnP problem, camera calibration; Binocular vision system, epipolar geometry, normalized camera, essential matrix; Fundamental matrix, eight-point algorithm; RANSAC algorithm; Inferring 3D world points; Multi-view reconstruction, structure from motion, generation of novel view.

Models for Transformation: Euclidean, similarity, affine, and homography transformations; Learning and inference in transformation models; Transformation between images; Image mosaicing, image registration, interpolation, similarity metric.

Low-to-Mid Level Processing: Convolution, spatial filtering; Gaussian blurring; Edge preserving smoothing; Bilateral filter; Harris corner detector; HOG; Graph cut.

Scale-Space Analysis: Anisotropic diffusion; Gaussian and Laplacian pyramids; Multiscale image registration and template matching; Image blending, hybrid image generation; LoG, DoG, scale-invariant feature transform.

Object Detection and Segmentation: Image classification, different architectures of CNN; Class activation map; Attention mechanism: channel, spatial; Semantic segmentation; Encoder-decoder, U-Net, SegNet; Object detection; Region proposal; R-CNN, region proposal network, YOLO, feature pyramid network; Instance segmentation, mask R-CNN.

Attention and Self-Supervision: Image captioning, RNN, attention; Vision transformer; Autoencoder, denosing autoencoder, learning by inpainting, colorization; Deep clustering of visual features.

Video Processing and Understanding: Motion estimation, optical flow, aperture problem; Lucas-Kanade method, iterative and multi-resolution approaches; Horn-Schunck method; Learning optical flow; Background subtraction; Active contour, Viterbi algorithm; Tracking using active contour; Video classification, early fusion, late fusion, 3D CNN; Action recognition from motion; Recurrent CNN.

3D Vision: Multi-view CNN, 3D shape representations, depth map, surface normal; Mesh R-CNN; Synthesis of novel view.

(b) **Prerequisites:** Linear Algebra, Image Processing

(c) **References:**

1. S. J. D. Prince, Computer Vision: Models, Learning, and Inference. Cambridge University Press, 1st ed., 2012.
2. Richard Szeliski, Computer Vision: Algorithms and Applications. Springer, 2011.
3. D. A. Forsyth and J. Ponce, Computer Vision, A Modern Approach. Prentice Hall Pearson, 2015.
4. R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2004.
5. A. Blake and A. Zisserman, Visual Reconstruction. MIT Press, Cambridge, 2003.

Computing Systems Security

(a) Computer Security: Linux Security Basics, Set-UID Privileged Programs, Shell-

shock Attack, Buffer-Overflow Attacks, Return-to-Libc Attacks, Race Condition, Dirty COW Attack, Meltdown and Spectre Attacks, Format String Attacks

Internet Security: Network Basics, Packet Sniffing and Spoofing, MAC Layer and Attacks, Network Layer: IP, ICMP and Attacks, Transport Layer: UDP and Attacks, Tansport Layer: TCP and Attacks, DNS and Attacks, Virtual Private Network (VPN), Firewall, BGP Routing and Attacks, Heartbleed Attack

Web Security: Web Security Basics, Cross-Site Request Forgery Attack, Cross-Site Scripting Attack, SQL Injection Attack, Clickjacking Attack

(b) **Prerequisites:** Operating Systems, Computer Networks, Discrete Mathematics

(c) **References:**

1. Computer & Internet Security: A Hands-on Approach, Wenliang Du.
2. Computer Security: A Hands-on Approach, Wenliang Du.
3. Internet Security: A Hands-on Approach, Wenliang Du.
4. C. P. Pfleeger, S. L. Pfleeger, and J. Margulies, Security in Computing. 5th ed., Prentice Hall, 2015.
5. D. Wheeler, Secure Programming HOWTO. (Source: https://www.dwheeler.com/secure-programs/)
6. M. Zalewski, Browser Security Handbook. Michael Zalewski, Google. (Source: https://code.google.com/archive/p/browsersec/wikis/Main.wiki)
7. B. S. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed., John Wiley & Sons, New York, 1995.
8. A. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, Boca Raton, 1996.

Cryptology I

(a) Symmetric-Key encryption:

i. Classical ciphers (like Shift, Substitute, Affine, Vigenère, Hill) and their cryptanalysis;

ii. Information Theoretic Security, and One-Time Pad;

iii. Pseudorandom Generators (PRG) and Stream ciphers (like CSS, RC4, Trivium, Salsa/ChaCha); Cryptanalysis of Stream Ciphers;

iv. Pseudorandom Function (PRF) and Pseudorandom Permutation (PRP); Block ciphers (like DES, AES); Cryptanalysis of Block ciphers;

v. Mode of Operations;

vi. Notion of CPA and CCA security with examples;

Cryptographic hash functions: Preimage, Second-Preimage and Collision resistance, Merkle-Damgard Transform; Universal and Almost-Universal hash functions;

Symmetric-Key authentication: Message Authentication Code (MAC) (like CBC-MAC, NMAC, HMAC), and their cryptanalysis;

Modern modes of operations: Authenticated Encryption;

Key Exchange Protocol: The Diffie-Hellman key exchange and its cryptanalysis;

Introduction to Public-key encryption (like The RSA, ElGamal, Rabin, Paillier encryption schemes) and their cryptanalysis;

Digital Signatures (like RSA, ElGamal, DSA) and their cryptanalysis;

Elliptic curve based Cryptography: The group of points on an elliptic curve; Elliptic curves over finite fields; The curve P256; The Montgomery and Edwards curves: curve 25519; ECDH and ECDSA;

(b) **Prerequisites:** None

(c) **References:**

1. J. Katz and Y. Lindell, Introduction to Modern Cryptography. Chapman & Hall/CRC, 2007.
2. D. R. Stinson, Cryptography Theory and Practice. 3rd ed., Chapman & Hall/CRC, 2006.
3. D. Boneh, V. Shoup, A Graduate Course in Applied Cryptography, 2023. (Source: http://toc.cryptobook.us).
4. B. S. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed., John Wiley & Sons, New York, 1995.
5. A. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, Boca Raton, 1996.

Cryptology II

(a) Theoretical construction of pseudorandom objects: One way functions, pseudorandom generators, pseudorandom functions and pseudorandom permutations.

Public-Key Encryption and Commitments: El-Gamal and its variants, additively homomorphic encryptions; CCA-security - Fujisaki Okamoto; Cramer-Shoup; Commitment Schemes; Pedersen, Hash commitments;

Zero-Knowledge Proofs: Knowledge Extraction; Zero-knowledge Simulation; Sigma Protocols (like Schnorr Chaum-Pederson; Okamoto's); General Sigma Protocols for any algebraic relations; Existential Soundness; OR proofs; Proof of Equal Plaintext; Proof of 1-bit messages;

Secure Multiparty Computations: Idea of Simulation-based definition; Oblivious Transfer; Yao's Garbled Circuits; GMW- dishonest majority; BGW - honest majority; Beaver's protocol in the pre-processing model - maliciously secure; MPC in the head - ZK from MPC;

Bilinear pairings: Bilinear pairings; Signature schemes from bilinear pairings; Identity-based encryption; Broadcast encryption.

Lattice Based Cryptology: Integer lattices; hard problems on lattices: SIS, LWE and ring LWE problems; Trapdoor sampling from lattices; signatures and encryption schemes from lattices;

Some other relevant topics may be covered – however it will depend on the instructor;

(b) **Prerequisites:** Cryptology I

(e) **References:**

1. O. Goldreich, Foundations of Cryptography Vol 1 and 2, Cambridge University Press.
2. D. Boneh and V Shoup, A Graduate Course in Applied Cryptography. (Source: http://toc.cryptobook.us)
3. S. D. Galbraith, Mathematics of Public Key Cryptography. Cambridge University Press, 2012.
4. R. Pass and A. Shelat, A Course in Cryptography, Lecture notes. (Source: https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf)
5. D. Micciancio, S. Goldwasser, Complexity of Lattice Problems: A Cryptographic Perspective. Kluwer, 2002.

6. L. C. Washington, Elliptic Curves: Number Theory and Cryptography. 2nd ed., CRC Press 2008.
7. S. Chatterjee and P. Sarkar, Identity-Based Encryption. Springer, 2011.

Cyber-Physical Systems

(a) Cyber-Physical Systems (CPS) in the real world, Basic principles of design and validation of CPS, AUTomotive Open System Architecture (AutoSAR), Industrial Internet-of-things (IIoT) implications, Building Automation, Medical CPS

CPS - Platform components: CPS Hardware platforms - Processors, Sensors, Actuators, CPS Network, Control Area Network (CAN), Automotive Ethernet, CPS, Software stack, Real Time Operating Systems (RTOS), Scheduling Real Time control tasks

Principles of Automated Control Design (basic control theory): Dynamical Systems and Stability, Controller Design Techniques, Stability Analysis: Common Lyapunov Function (CLF), Multiple Lyapunov Function (MLF), stability under slow switching, Performance under Packet drop and Noise

CPS implementation: From features to software components, Mapping software components to Electronic Control Units (ECU), CPS Performance Analysis - effect of scheduling, bus latency, sense and actuation faults on control performance, network congestion

Safety and Security Assurance of Cyber-Physical Systems: Advanced Automata based modelling and analysis, Timed and Hybrid Automata, Definition of trajectories, Zenoness, Formal Analysis, Flow-pipe construction, reachability analysis, Analysis of CPS Software, Weakest Pre-conditions, Bounded Model checking

Secure Deployment of CPS: Attack models, Secure Task mapping and Partitioning, State estimation for attack detection, Automotive Case Study

CPS Case studies and Tutorials

- MATLAB toolboxes - Simulink, Stateflow
- Control, Bus and Network Scheduling using Truetime
- Hybrid Automata Modeling : Flowpipe construction using Flowstar, SpaceX and Phaver tools
- CPS Software Verification: Frama-C, CBMC

- Automotive and Avionics : Software controllers for Antilock Braking Systems (ABS), Adaptive Cruise Control (ACC), Lane Departure Warning, Suspension Control, Flight Control Systems
- Heathcare: Artificial Pancreas/Infusion Pump/Pacemaker
- Green Buildings : automated lighting, Air-Condition (AC) control

(b) **Prerequisite:** None.

(c) **References:**

1. R. Alur, Principles of Cyber-Physical Systems. MIT Press, 2015.
2. A. Platzer, Lecture Notes on Cyber-Physical Systems, Carnegie Mellon University. (Source: https://symbolaris.com/course/fcps14/fcps14.pdf)
3. E. Lee and S. Seshia, Introduction to Embedded Systems – A Cyber Physical Systems Approach, 2nd ed., MIT Press, 2017.

Neural Networks and Deep Learning

(a) Introduction and Foundations: What is Deep Learning? Motivation and historical context, Biological inspiration and perceptron models, Universal Approximation Theorem, Overview of applications: vision, language, speech, science and technology.

Elementary Neural Networks: Perceptrons and Multi-Layer Perceptrons (MLPs), Activation functions (sigmoid, tanh, ReLU, GELU, etc.), Forward propagation and loss functions (MSE, cross-entropy), Backpropagation and gradient computation, Gradient descent and variants (SGD, momentum, Adagrad, Adam).

Training Deep Neural Networks: Issues in deep training: vanishing/exploding gradients, Weight initialization strategies (Xavier, He, orthogonal), Normalization techniques (BatchNorm, LayerNorm, GroupNorm), Dropout and regularization in deep networks, Data augmentation and early stopping, Hyperparameter tuning and best practices.

Convolutional Neural Networks (CNNs): Convolution, pooling, and feature maps, Architectures: LeNet, AlexNet, VGG, ResNet, DenseNet, Modern design patterns in CNNs (depthwise separable convolutions, Inception, EfficientNet), Applications in computer vision (classification, object detection, localization, semantic and instance segmentation).

Sequence Models: RNNs and LSTMs: Recurrent Neural Networks (RNNs): formulation and backpropagation through time, Limitations: vanishing/exploding gradients in sequences, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), Applications: time series, text modelling, speech recognition.

Advanced Topics in Deep Learning: Residual connections and highway networks, Graph Neural Networks (GNNs): basic concepts and applications, Neural architecture search (NAS) – overview, Scaling laws in deep learning, Ethical and societal impacts of deep learning (bias, interpretability, environmental cost).

(b) **Prerequisites:** Probability and Stochastic Processes, Linear Algebra, Machine Learning (must be done beforehand or taken in parallel)

(c) **References:**

1. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
2. A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, Dive into Deep Learning. (Open source: https://arxiv.org/abs/2106.11342)
3. M. Nielsen, Neural Networks and Deep Learning. 2015. (Open source: http://neuralnetworksanddeeplearning.com)
4. A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly, 3rd ed., 2022.

Digital Signal Processing

(a) Introduction: Applications of signal processing, elements of analog signal processing.

Discrete time signals and systems: Causal and stable systems, linear time invariant systems, difference equation representations, Fourier transform of sequences, transfer function.

Random signals: Stationary signals, autocorrelation function, power spectral density.

Sampling of continuous time signals: Frequency domain interpretation of sampling, reconstruction of band limited signals from samples.

The z-transform: Region of convergence, properties of z-transform, inverse z-transform, relation with other transforms.

Transfer function: Poles and zeroes, interpretation of causality and stability, frequency response for rational transfer functions, minimum phase and all-pass systems.

Transform analysis of discrete signals: Discrete Fourier series, discrete Fourier transform, relationships with Fourier transform of sequences.

Structures for discrete time systems: Block diagrams, signal flow graphs, direct, cascade and parallel forms, transposed forms, structures for FIR filters, lattice filters.

Effects of finite precision: Coefficient quantization, round-off noise, analysis of various structural forms, limit cycles in IIR filters.

Filter design: Filter specifications, design using analog filters, impulse invariance, bilinear transformation, frequency transformation of low-pass IIR filters, computer-aided design, FIR filter design by windowing.

Computation of DFT: Direct computation, FFT and other implementations, finite precision effects.

Applications of DFT: Fourier analysis of signals using DFT, DFT analysis of sinusoidal signals, spectral estimation, analysis of non-stationary signals.

Some advanced topics.

Practical exercises using MATLAB or other software.

(b) **Prerequisites:** None

(c) **References:**

1. A. V. Oppenheim and R. W. Schafer, Discrete Time Signal Processing. Prentice Hall, Englewood Cliffs, 1989.
2. S. K. Mitra, Digital Signal Processing. McGraw Hill, New York, 1998.
3. S. K. Mitra, Digital Signal Processing Laboratory Using MATLAB. McGraw Hill, New York, 1999.
4. A. Peled and B. Liu, Digital Signal Processing. Wiley, New York, 1976.

Discrete and Combinatorial Geometry

(a) Planarity and Crossing Number

Convexity and Lattices

Extremal problems in Discrete Geometry

Arrangement of Geometric and Algebraic Objects

Range Spaces, VC dimension, and Epsilon net/sample

Cuttings and Simplicial Partitions

Incidence Geometry (Geometric Approach) and its Applications

Algebraic Approach to Combinatorial Geometry

Applications of Topological Methods

Additional Topics: Measure Concentration in High Dimensions and its Applications, Metric Embedding

(b) **Prerequisites:** Design and Analysis of Algorithms, Discrete Mathematics, Probability and Stochastic Processes

(c) **References:**

1. J. Matousek, Lectures on Discrete Geometry. Springer, 2001
2. J. Matousek, Geometric Discrepancy. Springer, 1999
3. B. Chazelle, The Discrepancy Method. Cambridge University Press, 2002
5. P. K. Agarwal and J. Pach, Combinatorial Geometry. Wiley-Interscience, 1995
6. L. Guth, Polynomial Methods in Combinatorics. American Mathematical Society,

Distributed Computing

(a) Part 0: Introduction to distributed computing models

- Message passing model (CONGEST, LOCAL)
- Shared memory model

Part 1: Distributed systems algorithms

- Physical clocks, logical clocks, clock synchronization
- Message ordering and group communication
- Distributed mutual exclusion, deadlock detection, global predicate detection, checkpointing and rollback recovery

Part 2: Distributed network algorithms

- Broadcast, Spanning tree, shortest paths
- Leader election, Minimum Spanning Tree (MST)

- Symmetry breaking (coloring, maximal independent set, dominating set)

Part 3: Security and Fault Tolerance in Distributed Networks

- Fault-tolerant computation, consensus/agreement (Crash failure, Byzantine failure)
- Application to blockchains and cryptocurrency network

Part 4: Distributed Big Data

- MapReduce algorithms
- K-machine model
- Massively Parallel Computation (MPC)

Part 5 (Optional): Distributed computation by mobile robots/agents

(b) **Prerequisites:** Design and Analysis of Algorithms

(c) **References:**

1. N. Santoro, Design and Analysis of Distributed Algorithms. Wiley 2006.
2. A. Kshemkalyani and M. Singhal, Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press 2011.
3. H. Attiya, J. Welch. Distributed Computing: Fundamentals, Simulations, and Advanced Topics. Wiley 2004.
4. D. Peleg, Distributed Computing: A Locality-Sensitive Approach. SIAM 2000.
5. G. Pandurangan, Distributed Network Algorithms.

Fault Tolerance and Testing

(a) Origin of fault-tolerant computing, reliability, maintainability, testability, dependability; Faults, errors and fault models – stuck-at, bridging, delay, physical, component level; Design techniques for fault- tolerance, triple-modular redundancies, m-out-of-n codes, check sums, cyclic codes, Berger codes, etc; Fault tolerant design of VLSI circuits and systems; Concepts of t-diagnosability, self-checking, BIST, LSSD, etc; Testing and Design for testability, fault equivalence, dominance, checkpoints, test generation, D-algorithm, PODEM, FAN, Boolean difference, testability analysis, fault sampling, random pattern testability, testability-directed test generation, scan path, syndrome and parity testing, signature analysis; CMOS and PLA testing,

delay fault testing, system-on-a chip testing, core testing; BDDs. Formal verification: Introduction, Overview of Digital Design Verification, Simulators, Test Scenarios and Coverage, Binary Decision Diagrams (BDD), State Machines and Equivalence Checking, Model Checking; case studies.

(b) **Prerequisites:** Digital Design and Computer Architecture.

(c) **References:**

1. D. K. Pradhan, Fault Tolerant Computing. Vol 1 and 2, Prentice Hall, Englewood Cliffs, 1986.
2. B. W. Johnson, Design and Analysis of Fault-Tolerant System. Addison-Wesley, 1989.
3. V. D. Agrawal and S. C. Seth, Tutorial: Test Generation for VLSI Chips. IEEE Computer Society Press, Los Alamos, California, 1988.
4. M. A. Breuer, A. D. Friedman, and M. Abramovici, Digital Systems Testing and Testable Design. IEEE Press, Los Alamos, California, 1993.

Finite Model Theory

(a) Structures: Relations, functions, models, homomorphisms.

Games: Ehrenfreucht-Fraisse games; Pebble games; Model comparison.

Logic: Constants, variables, terms; sentences and formulas; First order Logic and Finite variable logic; Game characterizations; Undefinability results.

Descriptive Complexity: Word models and finite automata; Monadic second order logic, Büchi's Theorem; Turing machines and complexity classes; Fixpoint Logics; Characterizing PTIME; Fagin's Theorem.

Zero-one laws; Spectrum.

(b) **Prerequisites:** Discrete Mathematics

(c) **References:**

1. H.-D. Ebbinhaus and J. Flum, Finite Model Theory. Springer, 1995.
2. L. Libkin, Elements of Finite Model Theory. Springer, 2004.
3. N. Immerman, Descriptive Complexity. Springer, 1999.

4. E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein, Finite Model Theory and its Applications. Texts in Theoretical Computer Science, Springer, 2007.

Formal Verification

(a) System Modelling: Modelling sequential and parallel systems.

Requirement Specification: Specification languages and paradigms, Linear Temporal Logic (LTL), Computation Tree Logic (CTL) and variants.

Symbolic Methods: Decision Diagrams, SAT Solvers, Symbolic execution.

Techniques for verification: Explicit-State Model Checking, Symbolic Model Checking, Bounded Model Checking, Equivalence checking, Partial Order Reduction, Symbolic execution, Counterexample guided abstraction refinement, k-induction, Interpolation, Property Directed Reachability.

Tools: SAT solvers, BDD tools, Model Checkers.

(b) **Prerequisites:** Discrete Mathematics

(c) **References:**

1. C. Baier and J.-P. Katoen, Principles of Model Checking. The MIT Press, 2008.
2. E. M. Clarke, Jr., O. Grumberg, Daniel Kroening, and D. A. Peled, Model Checking. MIT Press.
3. M. Huth and M. Ryan, Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press.

Generative AI and Foundational Models

(a) Generative Modelling Basics: Density estimation and likelihood-based generative models, Variational inference and the reparameterization trick, Autoencoders and Variational Autoencoders (VAE), Energy-based models (overview and limitations)

Generative Adversarial Networks (GANs): Min-max optimization and adversarial training, GAN architecture and training, Variants: DCGAN, Conditional GANs

Diffusion Models and Score-based Generators: Diffusion as probabilistic generative modeling, Forward and reverse diffusion processes, Score-based models and denoising, Denoising Diffusion Probabilistic Models (DDPM), State-of-the-art systems: Stable Diffusion, Imagen, DALL·E (overview)

Transformers as the Backbone of Foundation Models: Recap: encoder, decoder, and encoder- decoder architectures, Self-attention and scalability, Pretraining objectives: masked language modeling, causal language modeling, contrastive learning, Sequence-to-sequence models: BERT, GPT, T5, multimodal architectures (CLIP, Flamingo, Gemini)

Large Language Models (LLMs) and Scaling: From Transfer Learning to Foundation Models, Scaling laws in neural networks and LLMs, Data requirements, curation, and quality control; Distributed training at scale (ZeRO, Megatron-LM, DeepSpeed), Fine-tuning strategies: full fine-tuning, adapters, LoRA, prompt tuning, Instruction tuning and RLHF (Reinforcement Learning with Human Feedback)

Evaluation of Generative Models and LLMs: Metrics: FID, IS for images; BLEU, ROUGE, perplexity for text; Evaluation challenges in LLMs (truthfulness, hallucination, coherence); Alignment benchmarks and safety evaluation

Ethics, Safety, and Societal Impacts: Bias, fairness, hallucination, toxicity in LLMs; Explainability and interpretability in large models; Misuse risks: deepfakes, misinformation, copyright; Governance, policy, and responsible deployment of foundation models.

(b) **Prerequisites:** Neural Networks and Deep Learning, Machine Learning

(c) **References:**

1. D. Foster, Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play. O'Reilly, 2nd ed., 2022.
2. Denis Rothman, Transformers for Natural Language Processing. Packt, 2nd ed., 2022.
3. S. Raschka, Build a Large Language Model (From Scratch). Manning, 2024.

Graph Algorithms

(a) *Shortest path (SP) problems:* Single source SP problem, SP tree, Ford's labelling method, labelling and scanning method, efficient scanning orders – topological order for acyclic networks, shortest first search for non-negative networks (Dijkstra), BFS search for general networks, correctness and analysis of the algorithms; All pair SP problem – Edmond-Karp method, Floyd's algorithm and its analysis.
Flows in Networks: Basic concepts, maxflow-mincut theorem, Ford and Fulkerson augmenting path method, integral flow theorem, maximum capacity augmentation,

Edmond-Karp method, Dinic's method and its analysis, Malhotra-Kumar-Maheswari method and its analysis, Preflow-push method (Goldberg Tarjan) and its analysis; Better time bounds for simple networks.
Minimum cost flow: Minimum cost augmentation and its analysis.
Matching problems: Basic concepts, bipartite matching – Edmond's blossom shrinking algorithm and its analysis; Recent developments.
Planarity: Basic fact about planarity, polynomial time algorithm.
Graph isomorphism: Importance of the problem, backtrack algorithm and its complexity, isomorphism complete problems, polynomial time algorithm for planar graphs, group theoretic methods.
NP-hard optimization problems: Exponential algorithms for some hard problems – dynamic programming algorithm for TSP, recursive algorithm for maximum independent set problem; Review of NP-completeness of decision problems associated with TSP, bin packing, knapsack, maximum clique, maximum independent set, minimum vertex cover, scheduling with independent task, chromatic number etc; Formulation of the concept of NP-hard optimization problem, perfect graphs and polynomial time algorithms for hard problems on graphs, approximation algorithms and classification of NP-optimization problems with respect to approximability.

(b) **Prerequisites:** Design and Analysis of Algorithms

(c) **References:**

1. G. Ausiello, A. Marchetti-Spaccamela, P. Crescenzi, G. Gambosi, M. Protasi, V. Kann, Complexity and Approximation: Combinatorial Optimization Problems and Their Approximation Properties. Springer, Berlin, 1999.
2. T. H. Cormen, C. E. Leisarsen, and R. L. Rivest, Introduction to Algorithms. Prentice Hall of India, New Delhi, 1998.
3. M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W, H. Freeman, New York, 1979
4. D. S. Hochbaum (Ed.), Approximate Solution of NP-Hard Problems. PWS Publishing, New York, 1947.
5. D. Jungnickel, Graphs, Networks and Algorithms. Springer-Verlag, Berlin, 1999.
6. K. Mehlhorn, Data Structures and Algorithms. Vol 2., Springer-Verlag, Berlin 1984.
7. M. Golumbic, Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York, 1980.

8. C. M. Hoffman, Group Theoretic Algorithms and Graph Isomorphisms. Springer-Verlag, Berlin, 1982.
9. C. H. Papadimitriou and K. Stiglitz, Combinatorial Optimization: Algorithms and Complexity. Prentice Hall of India, New Delhi, 1997.
10. R. E. Tarjan, Data Structures and Network Algorithms. SIAM, Philadelphia, 1983.
11. E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms. Galgotia Publications, New Delhi, 1985.

Image Processing

(a) *Introduction:* Image definition, image processing to computer vision; Applications and key stages of image processing.

Image Formation: Digital image acquisition; Image formation model; Sampling, quantization, dynamic range, image contrast; Spatial and intensity resolution.

Intensity Transformations: Basic concept of image enhancement, intensity transformation, spatial filtering and neighbours of a pixel; Different intensity transformation functions; Contrast stretching, intensity-level slicing, bit-plane slicing; Histogram, histogram equalisation, histogram specification; Local histogram processing, local contrast enhancement, histogram statistics.

Spatial Domain Filtering: Spatial filtering; Order-statistic filter; Weighted and unweighted filter; Point spread function and its properties; Convolution, correlation, and their properties; Linear spatial filtering; Separable kernels; Smoothing: box filter, Gaussian filter; Sharpening: Prewitt, Sobel, Laplacian.

Frequency Domain Filtering: Discrete Fourier transform and its properties; Convolution and correlation in frequency domain; Filtering in frequency domain: low-pass, high-pass, band-pass.

Edge Detection: Edge, edgels; Sobel operators; Linking edgels, local and global processing; Hough transform; Canny edge detection algorithm, non-maxima suppression, hysteresis edge linking; LoG, DoG.

Segmentation: Pixel classification, grey level thresholding; Optimum thresholding: Bayes analysis; Global and local thresholding; Otsu thresholding; Region growing, region splitting and merging.

Color Image Processing: Primary and secondary colors; Radiance, luminance, brightness; Hue, saturation, intensity; Color models: RGB, CMYK, HIS, YCbCr; Multi-

spectral images, principal component analysis; Pseudo color, intensity slicing, color coding, intensity to color transformations; Histogram processing, smoothing, sharpening, edge detection, segmentation.

Feature Extraction: Textural features, rolling bins in histogram, histograms of gradient magnitude and orientation, run-length matrix, grey level co-occurrence matrix, local binary patterns; Moments; Connected component analysis; Convex hull; Distance transform, medial axis transform, thinning, shape properties.

Compression: Predictive and transform compression; Lossy and lossless compression; Error criteria; Block truncation compression; Huffman coding; Transformed coding, arithmetic and run-length coding; Vector quantization; JPEG compression.

(b) **Prerequisites:** Linear Algebra

(c) **References:**

1. R. C. Gonzalez and R. E. Woods, Digital Image Processing. Prentice Hall, 2018.
2. M. Petrou and C. Petrou, Image Processing: The Fundamentals. John Wiley & Sons, Ltd, 2010.
3. A. Rosenfeld and A. C. Kak, Digital Picture Processing. 2nd ed., Vol 1 and 2, Academic Press, New York, 1982.
4. M. Petrou and S. Kamata, Image Processing: Dealing with Texture. John Wiley & Sons, Ltd, 2021.
5. M. Sonka, V. Hlavac and R. Boyle, Image Processing, Analysis and Machine Vision. 4th Ed., Cengage Learning, 2015.

Information Retrieval

(a) Syllabus

Foundations of Information Retrieval

- Overview of IR tasks and applications: search, QA, recommendation, and retrieval for LLMs.
- Text processing: tokenization, normalization, stopword removal, lemmatization.
- Inverted index construction and basic compression.

- Retrieval models: Vector Space Model, TF–IDF, BM25, and comparison between them.
- Query expansion and feedback: Rocchio, pseudo-relevance feedback, and relevance models (RM1, RM3).
- Evaluation: precision, recall, F1, MAP, NDCG, test collections, pooling, and significance testing.

Web-Scale Retrieval

- Web crawling and indexing at scale: BFS/DFS strategies, robots.txt, freshness, and incremental updates.
- Duplicate and near-duplicate detection: shingling, Jaccard similarity, MinHash, and LSH.
- Link analysis and ranking: PageRank, HITS, TrustRank.
- Query understanding: spelling correction, auto-completion, query suggestion, intent detection.

Neural Information Retrieval

- Motivation for neural retrieval: lexical vs. semantic matching; dense vs. sparse paradigms.
- Word and sentence embeddings: Word2Vec, GloVe, BERT, and Sentence-BERT.
- Dense retrieval with dual encoders: architecture, training objectives, in-batch and hard negatives.
- DPR, ANCE, and ColBERT concepts; approximate nearest neighbor search (FAISS) and retrieval pipelines.
- Cross-encoders and re-ranking: architecture, latency–quality tradeoff, and hybrid fusion.
- Knowledge distillation: transferring knowledge from cross-encoders to dual encoders; ANCE and ColBERT-v2.
- Evaluation benchmarks: MSMARCO, BEIR, MIRACL; domain adaptation and multilingual retrieval.
- Advanced approaches: late-interaction models (ColBERT), sparse neural retrievers (SPLADE, uniCOIL).
- Transition from retrieval to RAG architectures.

Retrieval-Augmented Generation and LLM Integration

- RAG architecture: retriever–generator interface and design motivation.
- Advanced designs: Fusion-in-Decoder (FiD), FiD-KD, Atlas, and related architectures.
- Prompting vs. retrieval: use cases, complementarities, and performance trade-offs.
- Evaluation of RAG systems: factual consistency, hallucination mitigation, and grounding-based evaluation.

Explainability in IR

- Explainability: local and global explanations, interpretability in neural ranking.
- Adversarial retrieval, evaluation robustness, and responsible IR principles.

(b) **Prerequisite(s):** Probability and Stochastic Processes, Linear Algebra, Neural Networks and Deep Learning.

(c) **References:**

1. C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008. (Source: https://nlp.stanford.edu/IR-book)
2. B. Mitra and N. Craswell, An Introduction to Neural Information Retrieval, Foundations and Trends in Information Retrieval, 2018.
3. Selected research papers on DPR, ColBERT, FiD, and RAG architectures.

Introduction to Cognitive Science

(a) From AI to real brains – the history of cognitive science, Psychophysics, introduction to experiment visualization using MATLAB (or other software), Perception, Human-Computer interaction, Vision, Face and object recognition, Learning and Development, Brain imaging and brain mapping, Social Cognition, Memory, Consciousness, the Developing Brain, Metacognitive approaches, Intelligence, Ethics, Cognitive Neuroscience and Security systems, Big data cognitive experiments – the Internet.

(b) **Prerequisites:** An interest about scientifically understanding human brain functions, including the human mind. Basic understanding of matrices and MATLAB would be helpful.

(c) **References:**

1. P. Thagard, Mind: Introduction to Cognitive Science. 2nd ed., 2005.
2. J. Friedenberg and G. Silverman, Cognitive Science: An Introduction to the Study of Mind. 2nd ed., SAGE Publications Inc, 2011.

Logic for Computer Science

(a) Propositional logic: Syntax, semantics, truth and satisfiability, logical consequence, compactness, Hilbert-style deduction system, Completeness theorem, Natural deduction system.

First order logic: syntax, semantics, truth and satisfiability, logical consequence, definability, compactness, Hilbert-style deduction system, Completeness theorem.Undecidability of satisfiability and finite satisfiability, introduction to finite model theory, EF games and partial isomorphism theorem, first-order (un)definability.

Modal Logic: Correspondence theory, bisimulations, finite model property, decidability, complexity, completeness theorems.

(b) **Prerequisites:** Discrete Mathematics

(c) **References:**

1. H. B. Enderton, A Mathematical Introduction to Logic. 2nd ed., Academic Press, 2001.
2. H.-D. Ebbinghaus, J. Flum and W. Thomas, Mathematical Logic. Springer-Verlag, 1994.
3. A. Nerode and R. Shore, Logic for Applications. Springer-Verlag, 1997.
4. M. Huth and M. Ryan, Logic in Computer Science. Cambridge University Press, 2004.
5. G. E. Hughes and M. J. Cresswell, A New Introduction to Modal Logic. Routledge, 1996.
6. P . Blackburn, M. de Rijke, and Y . Venema, Modal Logic. Cambridge University Press, 2001.

Machine Learning

(a) **Introduction to Machine Learning:** Definition, scope, and types of ML: supervised, unsupervised, reinforcement; ML workflow: data preprocessing, feature engineering, model selection, evaluation; Overview of applications in NLP, Computer Vision, and so on.

Understanding and Preparing Data: Statistical perspective on data and features, Increasing complexity of modern datasets and approaches to visualization, Data preprocessing: cleaning, imputation, and transformation; Handling categorical and mixed-type data, Outlier detection and treatment.

Classification: General idea, Bayes' classification, k-Nearest Neighbors, Perceptrons, Support Vector Machines (SVMs), Gaussian Discriminant Analysis: LDA and QDA, Logistic Regression, Decision Trees, Ensemble Classifiers: Bagging (Random Forest) and Boosting techniques (AdaBoost, Gradient Boosting: XGBoost); classifier performance measures: Precision, Recall, Sensitivity, Specificity, F1-Score, AUROC.

Regression: Simple and Multiple linear regression, Evaluating a Regressor: Sum of Squared Errors (SSE), Total Sum of Squares (SST), R^2 Score.

Density Estimation: Maximum Likelihood Estimation (MLE) and Maximum A Posteriori Estimation (MAP).

Dimensionality Reduction: Feature selection: Different criterion functions, Principal Components Analysis (PCA), Fisher's Linear Discriminant Analysis.

Unsupervised Learning: k-means clustering, Hierarchical clustering, DBSCAN, Spectral clustering.

Model Evaluation and Regularization: Overfitting and underfitting, Bias-variance trade-off, Cross-validation techniques.

Explainability in ML: Why explainability matters, Global vs. local interpretability, Model-specific vs. model-agnostic, Key methods: feature importance, LIME, SHAP.

(b) **Prerequisites:** Linear Algebra, Probability and Stochastic Processes

(c) **References:**

1. C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.
2. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer, 2nd Edition, 2009.

3. K. P. Murphy, Machine Learning: A Probabilistic Perspective. MIT Press, 2012.
4. A. C. Müller and S. Guido, Introduction to Machine Learning with Python. O'Reilly, 2016.
5. T. M. Mitchell, Machine Learning. McGraw Hill, 1997.
6. D. Barber, Bayesian Reasoning and Machine Learning. Cambridge University Press, 2012.

Math Toolkits for CS

(a) Information Theory and its applications: Entropy, Mutual Information, Chain rule, Relative entropy, KL divergence, and its applications in combinatorics and computer science

Spectral Graph Theory: Graph Laplacians and their eigenvalues, connections to random walks and mixing, isoperimetric and Cheeger inequalities, expanders, and algorithmic applications

Convex Geometry: Geometric properties of high-dimensional convex bodies, Fritz John's theorem and isotropy, Brunn-Minkowski's theorem and isoperimetric inequalities, Dvoretzky's theorem, concentration of measure, and applications to include volume computation and convex programming

Fourier Analysis on the Boolean cube: Fundamental results of Fourier analysis on the Boolean cube, and its applications in learning theory, social choice theory, and hardness of approximation

Algebraic Techniques in Combinatorics: Linear algebra background, orthogonality and rank arguments, and the polynomial method

(b) **Prerequisites:** Discrete Mathematics, Probability and Stochastic Processes

(c) **References:**

1. T. M. Cover and J. A. Thomas, Elements of Information Theory. John Wiley, 1991.
2. L. Trevisan, Graph Partitioning, Expanders and Spectral Methods. Lecture Notes, UC Berkeley, 2016.
2. J. Matoušek, Lectures on Discrete Geometry. Springer, 2002.

3. R. O'Donnell, Analysis of Boolean Functions. Cambridge University Press, 2014.
4. L. Babai and P. Frankl, Linear Algebra Methods in Combinatorics. Department of Computer Science, University of Chicago, preliminary version, 1992.
5. J. Matoušek, Thirty-Three Miniatures (Mathematical and Algorithmic Applications of Linear Algebra). American Mathematical Society, 2010.

Mobile Computing

(a) Introduction: Overview of wireless and mobile systems; Basic cellular concepts and architecture; Design objectives and performance issues; Radio resource management; Radio interface; Radio propagation and pathloss models; Channel interference and frequency reuse; Cell splitting; Static and dynamic channel assignment strategies; Channel borrowing schemes; Overview of generations of cellular systems:- 1G to 5G and beyond.

Wireless communication fundamentals: Introduction to narrowband and wideband systems; Spread spectrum; Frequency hopping; Introduction to MIMO; MIMO Channel Capacity and diversity gain; Introduction to OFDM; MIMO-OFDM system; Multiple access control (FDMA, TDMA, CDMA, SDMA); Wireless local area network; Wireless personal area network (Bluetooth and zigbee).

Base station planning and optimization: Several strategies for the optimal placement of the base stations to maximize coverage, capacity, and network lifetime while minimizing costs and energy consumption.

Location and handoff management: Introduction to location management (HLR and VLR); Mobility models characterizing individual node movement (Random walk, Fluid flow, Markovian, Activity based); Mobility models characterizing the movement of groups of nodes (Reference point based group mobility model, Community based group mobility model); Static location management schemes (Always vs. Never update, Reporting Cells, Location Areas); Dynamic location management schemes (Time, Movement, Distance, Profile Based); Tracking Area and Tracking Area List based location tracking, Terminal Paging (Simultaneous paging, Sequential paging); Introduction to handoffs; Overview of handoff process; Factors affecting handoffs and performance evaluation metrics; Handoff strategies; Different types of handoffs (soft, hard, horizontal, vertical), Vertical handover decision algorithms, Conditional handoffs.

Mobile Ad hoc networks: Characteristics and applications; Coverage and connectivity problems; Routing and topology control in MANETs.

Wireless sensor networks: Concepts, basic architecture, design objectives and applications; Sensing and communication range; Coverage and connectivity; Sensor placement; Data relaying and aggregation; Energy consumption; Clustering of sensors; Energy efficient Routing (LEACH), Fault detection and recovery algorithms. Topology control, Target and event detection mechanism.

Cognitive radio networks: Fixed and dynamic spectrum access; Direct and indirect spectrum sensing; Spectrum sharing; Interoperability and co-existence issues; Applications of cognitive radio networks.

D2D communications: Introduction to D2D communications; Different architectures; Resource management, power control and mode selection problems; Relay selection problem.

Emerging topics: Introduction to millimeterwave communication, mmWave propagation characteristics, architecture and MAC access; Introduction to RIS, Active and passive Beamforming, RIS Deployment and Optimization; Fundamentals of UAVs: Classification, components, and civilian applications, including delivery, surveillance, and disaster management; UAV sensing and control, UAV trajectory planning, Mobile edge-computing (MEC) for UAVs, Location-based services (LBS) for UAVs.

Labs: Development and implementation of different network protocols using network simulators such as NS-3 and OMNET++.

(b) **Prerequisites:** Computer Networks

(c) **References:**

1. T. Rappaport, Wireless Communications: Principles and Practice. Pearson Education.
2. J. Schiller, Mobile Communications. Pearson Education.
3. A. Goldsmith, Wireless Communications. Cambridge University Press.
4. E. Biglieri, MIMO Wireless Communications. Cambridge University Press.
5. I. Stojmenovic, Handbook of Wireless Networking and Mobile Computing. Wiley.
6. J. Cowling, Dynamic Location Management in Heterogeneous Cellular Networks. MIT Thesis. (Source: http://people.csail.mit.edu/cowling/hons/jcowling-dynamic-Nov04.pdf)

7. T. Keshav, Location Management in Wireless Cellular Networks. (Source: https://www.cse.wustl.edu/~jain/cse574-06/ftp/cellular_location.pdf)
8. F. A. Batayneh, Location Management in Wireless Data Networks. (Source: https://www.cse.wustl.edu/~jain/cse574-06/ftp/wireless_location.pdf)
9. G. L. Stüber, Principles of Mobile Communication. Springer.
10. L. Song, D. Niyato, Z. Han, and E. Hossain, Wireless Device-to- Device Communications and Networks. Cambridge University Press.
11. E. Biglieri, A. J. Goldsmith, L. J. Greenstein, N. Mandayam, and H. V. Poor, Principles of Cognitive Radio. Cambridge University Press.
12. E. H. Callaway, Jr. and E. H. Callaway, Wireless Sensor Networks: Architectures and Protocols. CRC Press, 2003.
13. ns-3 Manual. (Source: https://www.nsnam.org/docs/manual/html/index.html)

Natural Language Processing

(a) Fundamentals: Introduction to NLP and language engineering, Pre-neural language processing: Components/tools of pre-neural NLP systems; Statistical Language Learning: n-gram based language modelling, Hidden Markov Model (HMM) and its use in POS tagging; EM algorithm, Basics of Statistical Machine Translation (SMT).

Neural language processing: Word embedding (Dense), RNNs, Vanishing gradient problem, and LSTM; Sequence-to-sequence learning: Encoder-decoder Framework; Attention mechanism and Transformer; Encoder-only models (e.g., BERT), encoder-decoder models (e.g., T5), and decoder-only models (e.g., GPT).

Large language models: Collecting datasets at the trillion token scale to train transformers; Inference: how to generate text from an LLM including topics such as beam-search, greedy decoding, and random sampling; Prompting and In-context Learning: Zero-shot/few-shot prompting, chain-of-thought and its variants; Adapting Transformers for specific use cases: Fine-tuning, and parameter efficient techniques such as LoRA, Q-LoRA, prompt-tuning, etc.; Optimization Techniques: Device/hardware considerations (GPU computing, CUDA programming), topics such as Grouped-Query Attention (GQA), Mixture-of-Experts (MoE), FlashAttention, and other modern approaches to Attention; LLMs and Reinforcement Learning: Human

preference incorporation (e.g., RLHF, PPO, DPO) and reasoning models (RLVR/GRPO) with reference to DeepSeek or Qwen; LLM agents.

(b) **Prerequisites:** The student should have had a formal course on Programming and Data Structures, and basic familiarity with Probability, Statistics and Neural Networks. Proficiency in `python`, and familiarity with standard approaches in deep learning (MLP, CNN, backpropagation, stochastic gradient descent, loss functions, etc.).

(c) **References:**

1. C. D. Manning and H. Schütze, Foundations of Statistical Natural Language Processing.
2. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
3. C. M. Bishop and H. Bishop, Deep Learning: Foundations and Concepts, Springer-Nature New York Inc, 2023.

Optimization Techniques

(a) Linear Programming: Theory of LP — geometric interpretation of LP; basic feasible solution; feasible region of LP, convexity and convex polyhedra; vertices of the convex polyhedron, linear independence and basic feasible solution; Algorithms for LP — a brief review of simplex and revised simplex algorithms, Bland's rule, polynomial time algorithms – ellipsoidal and interior point methods; Duality — duality of LP, weak duality and strong duality theorems; Farkas lemma; Applications of LP — some applications from graph theory, game theory, LP relaxation to be done.

Integer Programming: Integer and mixed integer programming problems, cutting planes and branch and bound algorithms, NP-completeness of integer programming and ILP, travelling salesman and other related problems.

Non-linear Programming: Quadratic programming, convex programming problems; Unconstrained and constrained optimization problems; Karush-Kuhn-Tucker-Lagrangian necessary and sufficient conditions, interior point methods, standard algorithms like gradient descent, steepest descent, Newton's method, etc., ideas of convergence of the methods;

Semidefinite Programming

(b) **Prerequisites:** None

(c) **References:**

1. R. J. Vanderbei, Linear Programming Foundations and Extensions. Kluwer Academic Publishers, Boston/London, 1997.
2. D. G. Luenberger and Y. Ye, Linear and Non-Linear Programming. Springer, 2010.
3. C. H. Papadimitriou and K. Steiglitz, Combinational Optimization, Prentice Hall, Englewood Cliffs, 1982.
4. R. Garfinkel and G. Nemhauser, Integer Programming. John Wiley, New York, 1976.
5. G. Nemhauser and L. Wolsey, Integer and Combinational Optimization. Wiley, New York, 1988.
6. D. Bertsekas, Non-Linear Programming. Athena Scientific, Belmont, Mass., 1995.
7. S. Nash and A. Sofer, Linear and Non-Linear Programming. McGraw Hill, New York, 1996.
8. F. Hillier and G. Liebermann, Introduction to Mathematical Programming. McGraw Hill, 1995.
9. K. G. Murty, Linear and Combinatorial Programming. John Wiley, New York, 1976.
10. M. Bazaraa, J. Jarvis, and H. Sherali, Linear Programming and Network Flows. Wiley, New York, 1977.
11. W. I. Zangwill, Non-Linear Programming. Prentice Hall, New Jersey, 1969.
12. R. Fletcher, Practical Methods of Constrained Optimization. John Wiley, Chichester, 1981.
13. J. Matoušek and B. Gärtner: Understanding and Using Linear Programming. Springer, 2007.
14. S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2009.
15. V. Chvátal, Linear Programming. W. H. Freeman & Co. Ltd., 1983.

(a) A. Schrijver, Theory of Linear and Integer Programming. Wiley, 1998.

16. G. M. Ziegler, Lectures on Polytopes. Springer, 2012.
17. A. Schrijver, Combinatorial Optimization. Vol A, B and C, Springer, 2003.

Parallel Algorithms

(a) Parallel Programming Models: Shared-memory model (PRAM, MIMD, SIMD), network model (line, ring, mesh, hypercube), performance measurement of parallel algorithms.

Algorithm Design Techniquess for PRAM Models: Balancing, divide and conquer, parallel prefix computation, pointer jumping, symmetry breaking, pipelining, accelerated cascading.

Algorithms for PRAM Models: List ranking, sorting and searching, tree algorithms, graph algorithms.

Parallel Complexity: Lower bounds for PRAM models, the complexity class NC, P-completeness.

(b) **Prerequisite:** Design and Analysis of Algorithms.

(c) **References:**

1. J. F. Jaja, An Introduction to Parallel Algorithms. Addison-Wesley, 1992.
2. M. J. Quinn, Parallel Computing: Theory and Practice. 2nd ed., McGraw Hill, 1994.
3. F. Gebali, Algorithms and Parallel Computing. Wiley, 2011.

Principles of Programming Languages

(a) **Introduction:** Overview of different programming paradigms e.g. imperative, object oriented, functional, logic and concurrent programming.

Syntax and semantics of programming languages: A quick overview of syntax specification and semiformal semantic specification using attribute grammar.

Imperative and OO Languages: Names, their scope, life and binding. Control-flow, control abstraction; in subprogram and exception handling. Primitive and constructed data types, data abstraction, inheritance, type checking and polymorphism.

Functional Languages: Typed-calculus, higher order functions and types, evaluation strategies, type checking, implementation.

Logic Programming: Computing with relation, first-order logic, SLD-resolution, unification, sequencing of control, negation, implementation, case study.

Concurrency: Communication and synchronization, shared memory and message passing, safety and liveness properties, multithreaded program.

Formal Semantics: Operational, denotational and axiomatic semantics, languages with higher order constructs and types, recursive type, subtype, semantics of non-determinism and concurrency.

Assignments: Using one or more of the following as much time permits: C++ / Java / OCAML / Lisp / Haskell / Prolog

(b) **Prerequisites:** None

(c) **References:**

1. G. Winskel, A Formal Semantics of Programming Languages: An Introduction. MIT Press, 1993.
2. B. C. Pierce, Types and Programming Languages. MIT Press, 2002.
3. D. P. Friedman, M. Wand and C. T. Haynes, Essentials of Programming Languages. 3rd ed., MIT Press, 2008.
4. T. W. Pratt, M. V. Zelkowit, Programming Languages: Design and Implementation. 4th ed., Pearson, 2000.
5. A. B. Tucker and R. Noonan, Programming Languages, Principles and Paradigms. McGraw-Hill Education, 2nd ed., 2006.
6. R. W. Sebesta, Concepts of Programming Languages. Pearson Education, 10th ed., 2013.

Quantum Computations and Cryptography

(a) **Introduction and Basics**

- Brief history about the emergence of quantum computation and cryptograhy.
- Brief review of fundamental concepts of the quantum theory: Axioms of Hilbert space quantum mechanics; indeterminism; interference; uncertainty; superposition; entanglement.
- Quantum bits; Reversible computation; Measurement; Schmidt decomposition
- Density operators; General quantum measurement(POVM); Evolution of quantum state, partial trace
- Quantum no-cloning theorem

Quantum Algorithms

- Bernstein–Vazirani algorithm
- Simon's algorithm
- Deutsch's algorithm
- Deutsch-Jozsa algorithm
- Quantum Fourier Transform and Quantum phase estimation
- Shor's factoring Algorithm
- Grover 's search algorithm

Quantum Cryptography

- Motivation: why quantum cryptography
- No-cloning theorem and its cryptographic implications
- Bell inequalities and CHSH game
- Entanglement in cryptographic tasks
- Security definitions in cryptography
- Classical vs. quantum adversaries
- Privacy amplification against quantum adversaries
- Quantum Key Distribution. Detail description of BB84 protocol

(b) **Prerequisites:** Data Structures and Algorithms, Design and Analysis of Algorithms, Digital Design and Computer Architecture

(c) **References:**

1. M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information. Cambridge University Press, 2010.
2. J. Watrous, The Theory of Quantum Information. Cambridge University Press, 2018.
3. T. Vidick, Introduction to Quantum Cryptography. Cambridge University Press, 2023.

Quantum Information Theory

(a) *Introduction:* A brief history about the emergence of quantum information; a brief review of fundamental concepts of the quantum theory: axioms of Hilbert space quantum mechanics; indeterminism; interference; uncertainty; superposition; entanglement.

The Noiseless and Noisy Quantum Information: (i) Noiseless Quantum Information: quantum Bits and qudits; Reversible evolution; Measurement; Schmidt decomposition; HJW theorem, Kraus representation theorem. (ii) Noisy Quantum Information: Density operators; General quantum measurement(POVM); Evolution of quantum states; Quantum channels and their examples; Purification; Isometric evolution.

Basic Quantum Protocols and Resource Inequalities: Entanglement Distribution; Super dense coding; Teleportation; Optimality of quantum protocols; Extension to qudits; Quantum nonlocality and contextuality and their applications - Bell theorem and CHSH game.

Basic Tools and Information Measures in Quantum Information: Distance Measures: Trace Distance; Fidelity; Purified Distance; Relationship between various distance measures; Gentle measurement lemma. Information Measures and their Properties: Shannon entropy; Relative entropy; Von Neumann entropy; Quantum relative entropy; Coherent information; Hypothesis testing divergence; Max relative entropy; additivity and sub-additivity property of various information measures.

Quantum Shannon Theory: Noiseless Tasks: Schumacher Compression; Distillation of entanglement; State merging; State splitting; State redistribution. Noisy Tasks: Classical capacity of quantum channels; Holevo information; Private capacity of quantum Channels; Entanglement assisted classical capacity of quantum channels; Quantum capacity of quantum channel.

Quantum Cryptography: Privacy amplification and Information reconciliation; Quantum key distribution; Privacy and Quantum information; Security of quantum key distribution.

(b) **Prerequisites:** Basic familiarity with linear algebra and probability as determined by the teacher.

(c) **References:**

1. M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information. Cambridge University Press, 2010.

2. M. Hayashi, Quantum Information. Springer, 2017.
3. John Preskil, Lecture Notes. (Source: http://theory.caltech.edu/~preskill/ph229)
4. J. Watrous, Quantum Computation Lecture Notes. (Source: https://cs.uwaterloo.ca/~watrous/QC-notes)

Reinforcement Learning

(a) Syllabus

Introduction, Bandits, and Markov Decision Processes (3 lectures)

- Agent–environment interaction, rewards, goals, and returns.
- Exploration–exploitation dilemma; reinforcement vs supervised learning.
- Multi-armed bandits: ϵ-greedy, optimistic initialization, UCB, gradient bandit algorithms.
- Finite Markov Decision Processes: states, actions, rewards, transitions, discounting.
- Policies, value functions, Bellman equations, optimality concepts.

Dynamic Programming (1 lecture)

- Policy evaluation, policy improvement, policy iteration, value iteration.
- Generalized policy iteration and convergence intuition.

Monte Carlo Methods (2 lectures)

- Monte Carlo prediction and control.
- On-policy and off-policy estimation; importance sampling.
- Comparison with dynamic programming; motivation for temporal-difference learning.

Temporal-Difference Learning and Eligibility Traces (6 lectures)

- TD prediction: TD(0) update, bias–variance tradeoff, convergence idea.
- Control: Sarsa, Q-learning, Expected Sarsa, and Double Q-learning.
- n-step TD and the λ-return; connection between MC and TD methods.

- Eligibility traces: TD(λ), Sarsa(λ), and True Online TD(λ); intuition and applications.

Planning and Search (2 lectures)

- Dyna architecture: integrating model learning, planning, and acting.
- Sample vs expected updates; model learning.
- Monte Carlo Tree Search (MCTS): selection, expansion, simulation, backpropagation.

Value Function Approximation (Linear and Neural) (3 lectures)

- Semi-gradient TD, least-squares TD.
- Feature construction: coarse coding, tile coding (demonstration).
- Instability in off-policy learning and the "deadly triad".
- Nonlinear function approximation using neural networks.
- Key mechanisms: replay buffer, target networks, and stability challenges.

Policy Gradient and Actor–Critic Methods (2 lectures)

- Policy Gradient Theorem and REINFORCE algorithm.
- Variance reduction via baseline.
- Actor–Critic architecture and advantage estimation.

Deep Reinforcement Learning (5 lectures)

- Deep Q-Network (DQN): integrating target networks and replay buffers.
- DQN variants: Double DQN, dueling networks.
- Deep actor–critic methods: A3C, PPO.
- AlphaZero connection: unifying value, policy, and planning.

Advanced Topics and wrap-up (4 lectures)

- Meta-RL motivation: learning to adapt across tasks.
- Memory-based (RL^2) and gradient-based (MAML) meta-learning—conceptual overview.
- Transfer and imitation learning; exploration, generalization, and safety.

- Case study: AlphaGo and AlphaZero—combining MCTS with neural value and policy networks.
- Other modern advances in RL.

(b) **Prerequisite(s):** Probability and Stochastic Processes, Linear Algebra, Neural Networks and Deep Learning (must be done beforehand or taken in parallel)

(c) **References:**

1. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. 2nd ed., MIT Press, 2018.
2. V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, An Introduction to Deep Reinforcement Learning. *arXiv:1811.12560*, 2018. (Source: https://arxiv.org/abs/1811.12560)
3. A. Agarwal, N. Jiang, S. M. Kakade, and W. Sun, Reinforcement Learning: Theory and Algorithms. UW Seattle, 2021. (Source: https://rltheorybook.github.io)
4. D. Silver, et al., Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529, pp. 484-489, 2016.
5. C. Finn, et al., Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. Proc. ICML, 2017.

Selected Topics of Image Processing

(a) *Edge Preserving Smoothing:* Enhancement, smoothing, low-pass filtering, Convolution and spatial filtering; Gaussian kernel and edge adaptive smoothing; Bilateral filter; Mean shift smoothing; Scale space representation, anisotropic diffusion.

2-D Transformations of Images and Frequency Domain Filtering: Frequency domain analysis, discrete Fourier transform, fast Fourier transform, convolution and correlation in frequency domain, frequency domain filtering; Walsh transform; Hadamard transform; Discrete cosine transform; Hotelling transform.

Image Restoration: Homomorphic filtering; Minimum mean square restoration, Wiener filter; Least square restoration, constrained least-squares restoration; Blind deconvolution; Superresolution.

Segmentation: Model-based - facet model, semantic (saliency based) region grouping; Interactive segmentation – Graph cut, growcut; Active contour / Snake model, Viterbi algorithm.

Image Registration: Monomodal, multimodal, modality to model registration; Feature space, search space, search strategy; Extrinsic, intrinsic, local, global registration; Nearest neighbour and bilinear interpolation; Similarity metric: cross-correlation, sum of absolute differences, mutual information - properties, computation, criteria.

Texture Analysis: Stationary texture analysis, texture features from power spectrum, magnitude and phase of Fourier transform; Non-stationary texture analysis, uncertainty principle in image processing, short time Fourier transform; Gabor transform, spatio-frequency domain analysis;

Multi-resolution Image Analysis: Gaussian and Laplacian pyramids; Continuous wavelet analysis, dyadic wavelet, fast wavelet analysis, fast inverse wavelet analysis, 1D and 2D wavelets; Wavelet packets; Wavelet based compression.

Morphological Image Processing: Erosion, dilation, opening, closing, Hit-or-Miss transformation; Grey-scale morphology, area morphology; Watershade algorithm.

Image Analysis: Pattern spectrum; Structural features, polygonal approximation; Shape matching, template matching, shape metric, image understanding.

Image Databases: Attribute list, relational attributes, indexing, storage and retrieval; Content based image retrieval.

Some Applications (from the following but not restricted to): (i) Biomedical image processing; (ii) Document image processing; (iii) Fingerprint classification; (iv) Digital water-marking; (v) Image fusion; (vi) Image dehazing; (vii) Face detection and recognition; (viii) Content aware resizing.

(b) **Prerequisites:** Image Processing

(c) **References:**

1. R. C. Gonzalez and R. E. Woods, Digital Image Processing. Prentice Hall, 2018.
2. M. Petrou and C. Petrou, Image Processing: The Fundamentals. John Wiley & Sons, Ltd, 2010.
3. M. Petrou and P. G. Sevilla, Image Processing: Dealing with Texture. John Wiley & Sons, Ltd, 2006.
4. M. Sonka, V. Hlavac, and R. Boyle, Image Processing: Analysis and Machine Vision. PWS Pub. Co., London, 1998.
5. K. R. Castleman, Digital Image Processing. Prentice Hall, Englewood Cliffs, 1996.

6. A. R. Rao, Taxonomy for Texture Description. Springer-Verlag, Berlin, 1990.
7. R. M. Haralick and L. G. Shapiro; Computer and Robot Vision. Vol 1 and 2, Addison-Wesley, 1992.
8. A. Rosenfeld and A. C. Kak, Digital Picture Processing. 2nd ed., (Vol 1 and 2), Academic Press, New York, 1982.
9. W. K. Pratt, Digital Image Processing. 2nd ed., John Wiley, New York, 1992.
10. H. C. Andrews and B. R. Hunt, Digital Image Restoration. Prentice Hall, Englewood Cliffs, 1977.
11. M. Vetteerli and J. Kovacevic, Wavelet and Sub-Band Coding. Prentice Hall, 1995.
12. R. M. Rangayyan, Biomedical Image Analysis. CRC Press, 2005.
13. S. Khoshafian and A. B. Baker, Multimedia and Imaging Databases. Morgan Kaufmann, San Mateo, 1996.
14. P. Soille, Morphological Image Analaysis. Springer, New York, 2003.

Special Topics in Complexity Theory

(a) Algebraic Complexity Theory: Various models of computation, like, circuit, algebraic branching program (ABP), formula; the classes VP and VNP; algebraic circuit lower bound; polynomial identity testing (PIT).

Pseudorandomness: Pseudorandom generator; Expander; Extractor; One-way function; Hardness vs Randomness; Derandomization of restricted computational models; High dimensional expander.

Hardness Amplification

Hardness of Approximation, PCP and Unique Game Conjecture

Introduction to Proof Complexity

Communication Complexity: Various communication models, like, two-party communication model (deterministic, randomized, public and private coins), multiparty communication model; lower bound techniques; communication complexity of various well known problems; applications in theoretical computer science.

Fine-Grained Complexity Theory: Basic setup; Lower Bounds from P!=NP, ETH (exponential time hypothesis), and SETH (strong exponential time hypothesis); Various central problems like orthogonal vector problem, all pair shortest path problem,

3-SUM problem, and deriving hardness results from the suitable hardness assumptions of these problems.

(b) **Prerequisite:** Design and Analysis of Algorithms; Computational Complexity.

(c) **References:**

1. Sanjeev Arora and Boaz Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 1st ed., 2009.
2. S. Jukna, Boolean Function Complexity: Advances and Frontiers, Springer, 2012.
3. E. Kushilevitz and N. Nisan, Communication Complexity, Cambridge University Press, 2009.
4. S. P. Vadhan, Pseudorandomness, Foundations and Trends® in Theoretical Computer Science, 2012.
5. R. Saptharishi, A survey of lower bounds in arithmetic circuit complexity. (Source: https://github.com/dasarpmar/lowerbounds-survey/releases/tag/v9.0.3)
6. K. Bringmann, Fine-Grained Complexity Theory. (Source: https://cms.sic.saarland/finegrained)

Statistical Computing

(a) Random number generation & randomness tests.

Nonparametric density estimation: Histogram, Kernel, Nearest Neighbors Density estimates

EM & MM Algorithms

Nonparametric regression: Kernel, Splines, Nearest Neighbors, Trees

Classification: Nearest neighbor classifiers, KDA, Tree, random forests

Markov Chains and Monte Carlo Methods

(b) **Prerequisite:** Statistical Methods, Probability and Stochastic Processes.

(c) **References:**

1. S. M. Ross, Simulation. Second edition.

2. L. Devroye, Non-uniform Random Variate Generation.
3. R. A. Thisted, Elements of Statistical Computing.
4. L. Breiman et al., Classification and Regression Trees.
5. M. P. Wand and M. C. Jones, Kernel smoothing.
6. D. W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization.
7. R. O. Duda P. E. Hart D. G. Stork, Pattern Classification.
8. D. Kundu and A. Basu, Statistical Computing.
9. G. McLachlan and T. Krishnan, The EM Algorithm and Extensions.
10. K. Lange, MM Optimization Algorithms.
11. C. de Boor, A Practical Guide to Splines
12. T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction.
13. C. Robert and G. Casella, Monte Carlo Statistical Methods.
14. W. R. Gilks, S. Richardson and D. J. Spiegelhater, Markov Chain Monte Carlo in Practice.

Topics in Privacy

(a) Cryptographic foundations: Homomorphic Encryption, group signatures, blind signatures, anonymous credential management, commitment schemes, zero-knowledge proofs, proof of knowledge, SNARK, oblivious transfer, secure multiparty computation, Oblivious RAM, private set intersections, private information retrieval.

Perturbation, K-anonymity, L-diversity

Differential privacy

De-anonymization techniques

Privacy-preserving analytics

Applications: Mixnets, Onion Routing (TOR), e-cash, e-voting, location privacy, profiling, Web Privacy (Online tracking and advertising), Bitcoin, Zerocash etc.

Privacy for outsourced data

Privacy risk analysis

Ethical Aspects of privacy: Privacy compliance, GDPR, HIPAA etc.

(b) **Prerequisites:** None

(c) **References:**

1. C. Dwork and A. Roth, The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science.
2. J. Katz and Y. Lindell, Introduction to Modern Crypography. 2nd ed., CRC Press.
3. R. Pass and A. Shelat, A Course in Cryptography. Lecture notes. (Source: https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf)