computational complexity theory

computational complexity theory is a fundamental area of theoretical computer science that studies the inherent difficulty of computational problems and classifies them based on the resources needed to solve them. This field analyzes algorithms to determine the time and space they require, helping to understand the practical limits of computation. By examining complexity classes, reductions, and completeness, computational complexity theory provides a framework to distinguish between problems that are efficiently solvable and those that are intractable. It plays a crucial role in cryptography, optimization, and algorithm design by identifying which problems can be feasibly tackled by computers. This article explores the key concepts, complexity classes, important theorems, and applications related to computational complexity theory. It also discusses the ongoing challenges and open questions that continue to drive research in this vital domain.

Fundamentals of Computational Complexity Theory

Computational complexity theory investigates the resources required for solving computational problems, primarily focusing on time and space. This theoretical framework helps differentiate between problems that can be solved efficiently and those that demand excessive computational resources. The theory assumes an abstract computational model, such as Turing machines, to measure complexity independently of hardware specifics. Central to this study is the concept of algorithms, which provide step-by-step procedures for problem-solving.

Computational Models

The study of computational complexity relies on abstract models like deterministic and nondeterministic Turing machines. These models enable formal definitions of complexity measures such as time complexity and space complexity. Time complexity refers to the number of computational steps an algorithm takes, whereas space complexity measures the amount of memory used. These metrics allow the classification of problems based on the growth rate of resources relative to input size.

Measuring Complexity

Time and space complexity are commonly expressed using Big O notation, which characterizes the upper bound of resource usage. Polynomial time algorithms, denoted as O(n^k) for some constant k, are considered efficient or feasible. Conversely, exponential time algorithms grow too quickly to be practical for large inputs. This distinction is fundamental in computational complexity theory because it guides the feasibility assessment of problem-solving methods.

Complexity Classes and Problem Classification

Complexity classes categorize computational problems according to the resources required for their solution. These classes serve as the foundation for understanding the relative difficulty of problems and their relationships. The most studied classes include P, NP, PSPACE, and EXP, each defined by different constraints on time or space.

Class P (Polynomial Time)

The class P consists of decision problems solvable by a deterministic Turing machine within polynomial time. Problems in P are considered tractable because efficient algorithms exist for their solution. Examples include sorting, searching, and graph connectivity problems. Computational complexity theory regards P as the benchmark for efficient computability.

Class NP (Nondeterministic Polynomial Time)

NP encompasses decision problems for which a given solution can be verified in polynomial time by a deterministic Turing machine. While it is unknown whether all NP problems can be solved efficiently, many important problems, such as the Boolean satisfiability problem (SAT), belong to NP. The relationship between P and NP remains one of the most significant open questions in the field.

Other Complexity Classes

Additional complexity classes include:

- **PSPACE:** Problems solvable using polynomial space regardless of time constraints.
- **EXP:** Problems solvable in exponential time.
- **co-NP:** Complements of problems in NP.
- **NP-Complete:** The hardest problems in NP to which every NP problem can be reduced.

Reductions and Completeness

Reductions are a central concept in computational complexity theory used to relate the difficulty of different problems. By transforming one problem into another, reductions help establish problem hardness and completeness within complexity classes.

Polynomial-Time Reductions

Polynomial-time reductions transform instances of one problem into another in polynomial time. If a problem A can be reduced to problem B, solving B efficiently implies an efficient solution for A. This concept is instrumental in classifying problems, especially in identifying NP-complete problems.

Completeness and Hardness

A problem is complete for a complexity class if it is both in the class and as hard as any other problem in that class. NP-complete problems, for example, are those to which every problem in NP can be reduced in polynomial time. These problems serve as benchmarks for the class's difficulty and are critical in understanding the boundaries of efficient computation.

Important Theorems in Computational Complexity

Several foundational theorems underpin computational complexity theory, providing structure and insight into the behavior of complexity classes and computational problems.

The Cook-Levin Theorem

The Cook-Levin theorem established that the Boolean satisfiability problem (SAT) is NP-complete. This groundbreaking result provided the first known NP-complete problem and initiated the study of NP-completeness, leading to the identification of numerous other NP-complete problems.

Time Hierarchy Theorem

The time hierarchy theorem demonstrates that given more time, a Turing machine can solve strictly more problems. This theorem proves that complexity classes defined by different time bounds are strictly contained within one another, establishing a hierarchy of complexity classes based on time constraints.

Space Hierarchy Theorem

Analogous to the time hierarchy theorem, the space hierarchy theorem shows that increasing the available memory allows for the solution of strictly more problems. This theorem confirms the existence of a hierarchy among complexity classes defined by space usage.

Applications of Computational Complexity Theory

Computational complexity theory has far-reaching applications in various domains of computer science, impacting algorithm design, cryptography, and optimization.

Algorithm Design and Analysis

Understanding complexity classes guides the design of efficient algorithms and helps identify when heuristic or approximation methods are necessary. By classifying problems, computational complexity theory informs whether an exact solution is feasible or if alternative approaches should be pursued.

Cryptography

Cryptography heavily relies on computational complexity theory to ensure security. The difficulty of problems such as integer factorization and discrete logarithms underpins the strength of cryptographic protocols. Complexity theory helps identify problems believed to be computationally hard, providing a foundation for secure encryption schemes.

Optimization Problems

Many optimization problems are NP-hard, meaning no known polynomial-time algorithms exist. Computational complexity theory aids in understanding these problems' intractability, motivating the development of approximation algorithms and specialized heuristics.

Current Challenges and Open Problems

Despite significant progress, computational complexity theory continues to face unresolved questions that drive ongoing research efforts.

The P vs NP Problem

The question of whether P equals NP is the most famous open problem in computational complexity theory. Resolving this question would have profound implications for computer science, mathematics, and cryptography. It remains unsolved despite decades of research.

Separations Between Complexity Classes

Determining strict separations between various complexity classes, such as NP and PSPACE or P and EXP, remains a challenging area. Proving these separations would deepen the understanding of computational limitations and capabilities.

Quantum Complexity

With the advent of quantum computing, complexity theory is expanding to study quantum complexity classes such as BQP. Understanding the power and limits of quantum algorithms relative to classical complexity classes presents new theoretical challenges and opportunities.

Frequently Asked Questions

What is computational complexity theory?

Computational complexity theory is a branch of theoretical computer science that studies the resources required to solve computational problems, such as time and space, and classifies problems based on their inherent difficulty.

What are the main complexity classes in computational complexity theory?

The main complexity classes include P (problems solvable in polynomial time), NP (nondeterministic polynomial time), co-NP, PSPACE (problems solvable with polynomial space), and EXP (exponential time), among others.

What is the P vs NP problem?

The P vs NP problem asks whether every problem whose solution can be verified quickly (in polynomial time) can also be solved quickly. It is one of the most important open problems in computer science.

What does NP-complete mean?

A problem is NP-complete if it is in NP and as hard as any problem in NP, meaning that if an efficient (polynomial time) algorithm is found for one NP-complete problem, all NP problems can be efficiently solved.

How does computational complexity theory impact real-world computing?

It helps in understanding which problems can be efficiently solved and which are likely intractable, guiding algorithm design, cryptography, optimization, and resource allocation in computing systems.

What is the significance of the class PSPACE?

PSPACE consists of decision problems solvable by a Turing machine using a polynomial amount of memory, regardless of the time taken, capturing problems that may require a lot of computation but limited memory.

What role do reductions play in computational complexity theory?

Reductions transform one problem into another in polynomial time and are used to show problem hardness and completeness, helping classify problems by their relative complexity.

What is a randomized complexity class?

Randomized complexity classes, such as BPP (Bounded-error Probabilistic Polynomial time), include problems solvable efficiently with algorithms that use randomization and have a bounded probability of error.

How does space complexity differ from time complexity?

Time complexity measures the number of steps to solve a problem, while space complexity measures the amount of memory used. Both are critical resources in computational complexity theory.

What are some recent trends in computational complexity research?

Recent trends include fine-grained complexity aiming to understand exact time bounds, quantum complexity exploring quantum algorithms, and advances in understanding hardness of approximation and circuit complexity.

Additional Resources

1. *Computational Complexity: A Modern Approach*
This comprehensive textbook by Sanjeev Arora and Boaz Barak provides a detailed introduction to the theory of computational complexity. It covers a wide range of topics including NP-completeness, probabilistic computation, interactive proofs, and more. The book is well-suited for advanced undergraduates and graduate students, offering both rigorous mathematical treatment and intuitive explanations.

2. *Introduction to the Theory of Computation*
Written by Michael Sipser, this classic text serves as a foundational introduction to computational theory, including complexity theory. It covers automata theory, computability, and complexity classes with clarity and precision. Its accessible style and well-structured content make it a favorite among students and instructors alike.

3. *Complexity Theory: Exploring the Limits of Efficient Algorithms*
By Ingo Wegener, this book focuses on the boundaries of efficient computation and the underlying complexity classes. It delves into circuit complexity, Boolean functions, and lower bound techniques. The text is valuable for those interested in theoretical computer science research and algorithmic limitations.

4. *Computational Complexity Theory*
Authored by Oded Goldreich, this book offers an in-depth exploration of complexity theory with a focus on the foundations and recent developments. It emphasizes rigorous proofs and formal

definitions, covering topics such as cryptography, randomness, and interactive proofs. The book is ideal for graduate students and researchers.

5. *Complexity and Cryptography: An Introduction*
By John Talbot and Dominic Welsh, this text bridges the gap between complexity theory and cryptography. It explains how computational hardness assumptions underpin cryptographic protocols and security. The book provides a clear introduction to complexity classes relevant to cryptography and is accessible to readers with a background in algorithms.

6. *Computational Complexity: A Conceptual Perspective*
This book by Oded Goldreich presents complexity theory from a conceptual viewpoint, focusing on the intuition behind the definitions and theorems. It aims to deepen understanding rather than cover the full breadth of the field. It is especially useful for readers seeking to grasp the core ideas driving complexity theory research.

7. *Introduction to Computational Complexity*
Found in the series by Ding-Zhu Du and Ker-I Ko, this book offers a concise yet thorough introduction to computational complexity. It covers classical topics such as NP-completeness, space complexity, and hierarchy theorems. The text is well-suited for advanced undergraduate students beginning their study of complexity theory.

8. *The Nature of Computation*
By Cristopher Moore and Stephan Mertens, this book combines complexity theory with computational problems in physics and mathematics. It provides a unique perspective on NP-completeness and algorithmic complexity through practical examples and problem-solving. The book is engaging for readers interested in the interplay between computation and other scientific fields.

9. *Computational Complexity: A Quantitative Perspective*
This text by Luca Trevisan emphasizes the quantitative aspects of complexity theory, such as resource bounds and algorithmic efficiency. It covers topics including circuit complexity, randomness, and hardness amplification. The book serves as a useful resource for those looking to understand complexity with a focus on quantitative analysis.

Computational Complexity Theory

computational complexity theory: Theory of Computational Complexity Ding-Zhu Du, Ker-I Ko, 2014-06-30 Praise for the First Edition ... complete, up-to-date coverage of computational complexity theory...the book promises to become the standard reference on computational complexity. —Zentralblatt MATH A thorough revision based on advances in the field of computational complexity and readers’ feedback, the Second Edition of Theory of Computational Complexity presents updates to the principles and applications essential to understanding modern computational complexity theory. The new edition continues to serve as a comprehensive resource on the use of software and computational approaches for solving algorithmic problems and the related difficulties that can be encountered. Maintaining extensive and detailed coverage, Theory of Computational Complexity, Second Edition, examines the theory and methods behind complexity theory, such as computational models, decision tree complexity, circuit complexity, and probabilistic complexity. The Second Edition also features recent developments on areas such as NP-completeness theory, as well as: A new combinatorial proof of the PCP theorem based on the notion of expander graphs, a research area in the field of computer science Additional exercises at varying levels of difficulty to further test comprehension of the presented material End-of-chapter literature reviews that summarize each topic and offer additional sources for further study Theory of Computational Complexity, Second Edition, is an excellent textbook for courses on computational theory and complexity at the graduate level. The book is also a useful reference for practitioners in the fields of computer science, engineering, and mathematics who utilize state-of-the-art software and computational methods to conduct research.

computational complexity theory: Computational Complexity Theory Juris Hartmanis, 1989 Computational complexity theory is the study of the quantitative laws that govern computing. This book contains the proceedings of the AMS Short Course on Computational Complexity Theory, held at the Joint Mathematics Meetings in Atlanta in January 1988.

computational complexity theory: Theory of Computation Dexter C. Kozen, 2006-05-08 This textbook is uniquely written with dual purpose. It cover cores material in the foundations of computing for graduate students in computer science and also provides an introduction to some more advanced topics for those intending further study in the area. This innovative text focuses primarily on computational complexity theory: the classification of computational problems in terms of their inherent complexity. The book contains an invaluable collection of lectures for first-year graduates on the theory of computation. Topics and features include more than 40 lectures for first year graduate students, and a dozen homework sets and exercises.

computational complexity theory: *Computational Complexity Theory* Steven Rudich, Avi Wigderson,

computational complexity theory: Computational Complexity Theory Fundamentals - HandBook Fabio Felgueiras, 2023-05-09 This book is an introduction to theoretical computer science, covering topics such as formal languages, automata theory, computability theory, and complexity theory. It provides a comprehensive overview of the foundational concepts, including regular languages and finite automata, context-free languages and pushdown automata, Turing machines and computability, and time and space complexity classes. The book also covers important theorems and results, such as the Pumping Lemma, the Church-Turing thesis, Godel's Incompleteness Theorem, and NP-completeness. It is written in a clear and concise manner, making it accessible to students and researchers with a basic understanding of discrete mathematics and programming. This book serves as an essential guide for anyone interested in the fundamental concepts of theoretical computer science.

computational complexity theory: *Computational Complexity* Sanjeev Arora, Boaz Barak, 2009-04-20 This beginning graduate textbook describes both recent achievements and classical results of computational complexity theory. Requiring essentially no background apart from mathematical maturity, the book can be used as a reference for self-study for anyone interested in complexity, including physicists, mathematicians, and other scientists, as well as a textbook for a variety of courses and seminars. More than 300 exercises are included with a selected hint set. The book starts with a broad introduction to the field and progresses to advanced results. Contents include: definition of Turing machines and basic time and space complexity classes, probabilistic algorithms, interactive proofs, cryptography, quantum computation, lower bounds for concrete computational models (decision trees, communication complexity, constant depth, algebraic and monotone circuits, proof complexity), average-case complexity and hardness amplification, derandomization and pseudorandom constructions, and the PCP theorem.

computational complexity theory: Computational Complexity Christos H. Papadimitriou, 1994 The first unified introduction and reference for the field of computational complexity. Virtually non-existent only 25 years ago, computational complexity has expanded tremendously and now comprises a major part of the researh activity in theoretical science.

computational complexity theory: Computability and Complexity Theory Steven Homer, Alan L. Selman, 2001 This volume introduces materials that are the core knowledge in the theory of computation. The book is self-contained, with a preliminary chapter describing key mathematical concepts and notations and subsequent chapters moving from the qualitative aspects of classical computability theory to the quantitative aspects of complexity theory. Dedicated chapters on undecidability, NP-completeness, and relative computability round off the work, which focuses on the limitations of computability and the distinctions between feasible and intractable.Topics and features:*Concise, focused materials cover the most fundamental concepts and results in the field of modern complexity theory, including the theory of NP-completeness, NP-hardness, the polynomial hierarchy, and complete problems for other complexity classes*Contains information that otherwise exists only in research literature and presents it in a unified, simplified manner; for example, about complements of complexity classes, search problems, and intermediate problems in NP*Provides key mathematical background information, including sections on logic and number theory and algebra*Supported by numerous exercises and supplementary problems for reinforcement and self-study purposes With its accessibility and well-devised organization, this text/reference is an excellent resource and guide for those looking to develop a solid grounding in the theory of computing. Beginning graduates, advanced undergraduates, and professionals involved in theoretical computer science, complexity theory, and computability will find the book an essential and practical learning tool.

computational complexity theory: Complexity Theory of Real Functions K. Ko, 2012-12-06 Starting with Cook's pioneering work on NP-completeness in 1970, polynomial complexity theory, the study of polynomial-time com putability, has quickly emerged as the new foundation of algorithms. On the one hand, it bridges the gap between the abstract approach of recursive function theory and the concrete approach of analysis of algorithms. It extends the notions and tools of the theory of computability to provide a solid theoretical foundation for the study of computational complexity of practical problems. In addition, the theoretical studies of the notion of polynomial-time tractability some times also yield interesting new practical algorithms. A typical exam ple is the application of the ellipsoid algorithm to combinatorial op timization problems (see, for example, Lovasz [1986]). On the other hand, it has a strong influence on many different branches of mathe matics, including combinatorial optimization, graph theory, number theory and cryptography. As a consequence, many researchers have begun to re-examine various branches of classical mathematics from the complexity point of view. For a given nonconstructive existence theorem in classical mathematics, one would like to find a construc tive proof which admits a polynomial-time algorithm for the solution. One of the examples is the recent work on algorithmic theory of per mutation groups. In the area of numerical computation, there are also two tradi tionally independent approaches: recursive analysis and numerical analysis.

computational complexity theory: Advances in Computational Complexity Theory Jin-yi Cai, 1993-01-01 * Recent papers on computational complexity theory * Contributions by some of the leading experts in the field This book will prove to be of lasting value in this fast-moving field as it provides expositions not found elsewhere. The book touches on some of the major topics in complexity theory and thus sheds light on this burgeoning area of research.

computational complexity theory: *Theory of Computational Complexity* Ding-Zhu Du, Ker-I Ko, 2011-10-24 A complete treatment of fundamentals and recent advances in complexity theory Complexity theory studies the inherent difficulties of solving algorithmic problems by digital computers. This comprehensive work discusses the major topics in complexity theory, including fundamental topics as well as recent breakthroughs not previously available in book form. Theory of Computational Complexity offers a thorough presentation of the fundamentals of complexity theory, including NP-completeness theory, the polynomial-time hierarchy, relativization, and the application to cryptography. It also examines the theory of nonuniform computational complexity, including the computational models of decision trees and Boolean circuits, and the notion of polynomial-time isomorphism. The theory of probabilistic complexity, which studies complexity issues related to randomized computation as well as interactive proof systems and probabilistically checkable proofs, is also covered. Extraordinary in both its breadth and depth, this volume: * Provides complete proofs of recent breakthroughs in complexity theory * Presents results in well-defined form with complete proofs and numerous exercises * Includes scores of graphs and figures to clarify difficult material An invaluable resource for researchers as well as an important guide for graduate and advanced undergraduate students, Theory of Computational Complexity is destined to become the standard reference in the field.

computational complexity theory: *Computational Complexity: A Quantitative Perspective* Marius Zimand, 2004-07-07 There has been a common perception that computational complexity is a theory of bad news because its most typical results assert that various real-world and innocent-looking tasks are infeasible. In fact, bad news is a relative term, and, indeed, in some situations (e.g., in cryptography), we want an adversary to not be able to perform a certain task. However, a bad news result does not automatically become useful in such a scenario. For this to happen, its hardness features have to be quantitatively evaluated and shown to manifest extensively.The book undertakes a quantitative analysis of some of the major results in complexity that regard either classes of problems or individual concrete problems. The size of some important classes are studied using resource-bounded topological and measure-theoretical tools. In the case of individual problems, the book studies relevant quantitative attributes such as approximation properties or the number of hard inputs at each length.One chapter is dedicated to abstract complexity theory, an older field which, however, deserves attention because it lays out the foundations of complexity. The other chapters, on the other hand, focus on recent and important developments in complexity. The book presents in a fairly detailed manner concepts that have been at the centre of the main research lines in complexity in the last decade or so, such as: average-complexity, quantum computation, hardness amplification, resource-bounded measure, the relation between one-way functions and pseudo-random generators, the relation between hard predicates and pseudo-random generators, extractors, derandomization of bounded-error probabilistic algorithms, probabilistically checkable proofs, non-approximability of optimization problems, and others.The book should appeal to graduate computer science students, and to researchers who have an interest in computer science theory and need a good understanding of computational complexity, e.g., researchers in algorithms, AI, logic, and other disciplines.·Emphasis is on relevant quantitative attributes of important results in complexity.·Coverage is self-contained and accessible to a wide audience.·Large range of important topics including: derandomization techniques, non-approximability of optimization problems, average-case complexity, quantum computation, one-way functions and pseudo-random generators, resource-bounded measure and topology.

computational complexity theory: Kolmogorov Complexity and Computational Complexity Osamu Watanabe, 2012-12-06 The mathematical theory of computation has given rise to two important ap proaches to the informal notion of complexity: Kolmogorov complexity, usu ally a complexity measure for a single object such as a string, a sequence etc., measures the amount of information necessary to describe the object. Compu tational complexity, usually a complexity measure for a set of objects, measures the compuational resources necessary to recognize or produce elements of the set. The relation between these two complexity measures has been considered for more than two decades, and may interesting and deep observations have been obtained. In March 1990, the Symposium on Theory and Application of Minimal Length Encoding was held at Stanford University as a part of the AAAI 1990 Spring Symposium Series. Some sessions of the symposium were dedicated to Kolmogorov complexity and its relations to the computational complexity the ory, and excellent expository talks were given there. Feeling that, due to the importance of the material, some way should be found to share these talks with researchers in the computer science community, I asked the speakers of those sessions to write survey papers based on their talks in the symposium. In response, five speakers from the sessions contributed the papers which appear in this book.

computational complexity theory: *Theories of Computational Complexity* C. Calude, 2011-08-18 This volume presents four machine-independent theories of computational complexity, which have been chosen for their intrinsic importance and practical relevance. The book includes a wealth of results - classical, recent, and others which have not been published before.In developing the mathematics underlying the size, dynamic and structural complexity measures, various connections with mathematical logic, constructive topology, probability and programming theories are established. The facts are presented in detail. Extensive examples are provided, to help clarify notions and constructions. The lists of exercises and problems include routine exercises, interesting results, as well as some open problems.

computational complexity theory: Introduction to the Theory of Complexity Daniel Pierre Bovet, Pierluigi Crescenzi, 1994 Using a balanced approach that is partly algorithmic and partly structuralist, this book systematically reviews the most significant results obtained in the study of computational complexity theory. Features over 120 worked examples, over 200 problems, and 400 figures.

computational complexity theory: The Complexity Theory Companion Lane Hemaspaandra, Mitsunori Ogihara, 2001-12-01 Here is an accessible, algorithmically oriented guide to some of the most interesting techniques of complexity theory. The book shows that simple algorithms are at the heart of complexity theory. The book is organized by technique rather than by topic. Each chapter focuses on one technique: what it is, and what results and applications it yields.

computational complexity theory: Computational Complexity Theory American Mathematical Society, 2014-05-10 Computational complexity theory is the study of the quantitative laws that govern computing. This book contains the proceedings of the AMS Short Course on Computational Complexity Theory, held at the Joint Mathematics Meetings in Atlanta in January 1988.

computational complexity theory: Computational Complexity Theory , 2004 Computational Complexity Theory is the study of how much of a given resource is required to perform the computations that interest us the most. Four decades of fruitful research have produced a rich and subtle theory of the relationship between different resource measures and problems. At the core of the theory are some of the most alluring open problems in mathematics. This book presents three weeks of lectures from the IAS/Park City Mathematics Institute Summer School on computational complexity. The first week gives a general introduction to the field, including descriptions of the basic mo.

computational complexity theory: *Mathematics and Computation* Avi Wigderson, 2019-10-29 From the winner of the Turing Award and the Abel Prize, an introduction to computational complexity theory, its connections and interactions with mathematics, and its central role in the natural and social sciences, technology, and philosophy Mathematics and Computation provides a broad, conceptual overview of computational complexity theory—the mathematical study of efficient computation. With important practical applications to computer science and industry, computational complexity theory has evolved into a highly interdisciplinary field, with strong links to most mathematical areas and to a growing number of scientific endeavors. Avi Wigderson takes a sweeping survey of complexity theory, emphasizing the field’s insights and challenges. He explains the ideas and motivations leading to key models, notions, and results. In particular, he looks at algorithms and complexity, computations and proofs, randomness and interaction, quantum and arithmetic computation, and cryptography and learning, all as parts of a cohesive whole with numerous cross-influences. Wigderson illustrates the immense breadth of the field, its beauty and richness, and its diverse and growing interactions with other areas of mathematics. He ends with a comprehensive look at the theory of computation, its methodology and aspirations, and the unique and fundamental ways in which it has shaped and will further shape science, technology, and society. For further reading, an extensive bibliography is provided for all topics covered. Mathematics and Computation is useful for undergraduate and graduate students in mathematics, computer science, and related fields, as well as researchers and teachers in these fields. Many parts require little background, and serve as an invitation to newcomers seeking an introduction to the theory of computation. Comprehensive coverage of computational complexity theory, and beyond High-level, intuitive exposition, which brings conceptual clarity to this central and dynamic scientific discipline Historical accounts of the evolution and motivations of central concepts and models A broad view of the theory of computation's influence on science, technology, and society Extensive bibliography

computational complexity theory: Complexity and Real Computation Lenore Blum, 1998 The classical theory of computation has been a successful framework for theoretical computer science. The thesis of this book, however, is that it provides an inadequate foundation for modern scientific computation where most of the algorithms are real number algorithms.

Related to computational complexity theory

COMPUTATIONAL Definition & Meaning - Merriam-Webster The meaning of COMPUTATION is the act or action of computing : calculation. How to use computation in a sentence

COMPUTATIONAL | English meaning - Cambridge Dictionary COMPUTATIONAL definition: 1. involving the calculation of answers, amounts, results, etc.: 2. using computers to study. Learn more

Computational science - Wikipedia Computational science, also known as scientific computing, technical computing or scientific computation (SC), is a division of science, and more specifically the computer sciences, which

COMPUTATIONAL definition | Cambridge English Dictionary COMPUTATIONAL meaning: 1. involving the calculation of answers, amounts, results, etc.: 2. using computers to study. Learn more

COMPUTATIONAL□□ (□□)□□□□□□□ In a concise introduction to the volume, the editors list areas in which computational modelling can contribute to the field of language acquisition. The objective of psychology, in the prevalent

Computation - Wikipedia Mechanical or electronic devices (or, historically, people) that perform computations are known as computers. Computer science is an academic field that involves the study of computation

Introduction to Computational Thinking and Data Science Introduction to Computational Thinking and Data Science Course Description 6.0002 is the continuation of 6.0001 Introduction to Computer Science and Programming in Python and is

MSCF - Master of Science in Computational Finance - Carnegie Discover the unique advantages of Carnegie Mellon's top-ranked MSCF program and learn about quantitative finance career opportunities

INTRODUCTION TO COMPUTATIONAL MATHEMATICS Introduction to Computational Mathematics The goal of computational mathematics, put simply, is to find or develop algo-rithms that solve mathematical problems computationally (ie. using

Computational thinking - Wikipedia Computational thinking (CT) refers to the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. [1]

COMPUTATIONAL Definition & Meaning - Merriam-Webster The meaning of COMPUTATION is the act or action of computing : calculation. How to use computation in a sentence

COMPUTATIONAL | English meaning - Cambridge Dictionary COMPUTATIONAL definition: 1. involving the calculation of answers, amounts, results, etc.: 2. using computers to study. Learn more

Computational science - Wikipedia Computational science, also known as scientific computing, technical computing or scientific computation (SC), is a division of science, and more specifically the computer sciences, which

COMPUTATIONAL definition | Cambridge English Dictionary COMPUTATIONAL meaning: 1. involving the calculation of answers, amounts, results, etc.: 2. using computers to study. Learn more

COMPUTATIONAL□□ (□□)□□□□□□□ In a concise introduction to the volume, the editors list areas in which computational modelling can contribute to the field of language acquisition. The objective of psychology, in the prevalent

Computation - Wikipedia Mechanical or electronic devices (or, historically, people) that perform computations are known as computers. Computer science is an academic field that involves the study of computation

Introduction to Computational Thinking and Data Science Introduction to Computational Thinking and Data Science Course Description 6.0002 is the continuation of 6.0001 Introduction to Computer Science and Programming in Python and is

MSCF - Master of Science in Computational Finance - Carnegie Discover the unique advantages of Carnegie Mellon's top-ranked MSCF program and learn about quantitative finance career opportunities

INTRODUCTION TO COMPUTATIONAL MATHEMATICS Introduction to Computational Mathematics The goal of computational mathematics, put simply, is to find or develop algo-rithms that solve mathematical problems computationally (ie. using

Computational thinking - Wikipedia Computational thinking (CT) refers to the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. [1]

COMPUTATIONAL Definition & Meaning - Merriam-Webster The meaning of COMPUTATION is the act or action of computing : calculation. How to use computation in a sentence

COMPUTATIONAL | English meaning - Cambridge Dictionary COMPUTATIONAL definition: 1. involving the calculation of answers, amounts, results, etc.: 2. using computers to study. Learn more

Computational science - Wikipedia Computational science, also known as scientific computing, technical computing or scientific computation (SC), is a division of science, and more specifically the computer sciences, which

COMPUTATIONAL definition | Cambridge English Dictionary COMPUTATIONAL meaning: 1. involving the calculation of answers, amounts, results, etc.: 2. using computers to study. Learn more

COMPUTATIONAL□□ (□□)□□□□□□□ In a concise introduction to the volume, the editors list areas in which computational modelling can contribute to the field of language acquisition. The objective of psychology, in the prevalent

Computation - Wikipedia Mechanical or electronic devices (or, historically, people) that perform computations are known as computers. Computer science is an academic field that involves the study of computation

Introduction to Computational Thinking and Data Science Introduction to Computational Thinking and Data Science Course Description 6.0002 is the continuation of 6.0001 Introduction to Computer Science and Programming in Python and is

MSCF - Master of Science in Computational Finance - Carnegie Discover the unique advantages of Carnegie Mellon's top-ranked MSCF program and learn about quantitative finance career opportunities

INTRODUCTION TO COMPUTATIONAL MATHEMATICS Introduction to Computational Mathematics The goal of computational mathematics, put simply, is to find or develop algo-rithms that solve mathematical problems computationally (ie. using

Computational thinking - Wikipedia Computational thinking (CT) refers to the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. [1]