Global Insights in Artificial Intelligence and Computing

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

Foundations and Frontiers in Theoretical Computer Science: A Review of Key Concepts and Open Problems

Ankur singh[1*]

[1]University of North America

[1]Singhan@live.uona.edu

Corresponding Author

Ankur singh

Singhan@live.uona.edu

Article History:
Submitted: 23-06-2025
Accepted: 24-07-2025
Published: 01-08-2025

Keywords:
Computability, complexity, algorithms, cryptography, quantum computing, learning theory, formal verification, interactive proofs.

ABSTRACT

The survey looks into the foundations and the frontiers of Theoretical Computer Science (TCS) which is a central bit of computing to understand capabilities, limitations, and models of computation. Its concentration areas are the automata theory, computability, complexity theory, logic, and algorithms, along with the multidisciplinary overlap with quantum computing, cryptography and geometry, and machine learning. The new advances--in interactive proofs and zero-knowledge protocols and in quantum complexity--all indicate that the field still has its youthful vitality. Such important open issues as P vs NP and circuit lower bounds still exist to make the study. First-order effects include allowing TCS to provide leverage to secure communication, efficient algorithms, formal verification and theory of artificial intelligence. Looking to the future, it can be said that there are several possible issues on the horizon in the sphere in the sense of both quantum computing and ethical AI and biological computing, respectively. The design, analysis and development of computer science is still centered on TCS, and it is important to consider theory and practice through the integration of the two.

INTRODUCTION

Theoretical Computer Science (TCS) is the very foundation of any computer science as a learning subject. It creates the formal foundations and abstract models of what it is possible to compute, with how much efficiency it can be computed and what exactly a computation is. In contrast to applied or experimental subjects, TCS is based on mathematical logic, deduction, and the generation of general

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

frameworks that can be applied to a variety of computational problems. What is important about theoretical computer science is that it is a foundation [1]. All programming methods, computing languages, or systems are underpinned, directly or indirectly, by theory. These principles are used to design the algorithms, evaluate their efficiency, establish the rightfulness of software, and even certify that there is no sense to solve some problems algorithmically. In one word, TCS addresses the questions like why and how we use the tools and techniques of the modern computing [2].

The origins of TCS can be dated to the beginning of the 20th century when such luminaries as Alan Turing, Alonzo Church and Kurt Goedel made their contributions. The concept of a Turing machine formalized the concept of algorithm and established the foundations of the concept of computability of a function. The work by Church and Gödel on lambda calculus and formal logic continued this and eventually led to what is now known as the ChurchTuring thesis - a fundamental conjecture in the field of TCS, and one which equates algorithmic computation to what can be computed using a Turing machine [3].

Formal approaches to several subfields of computer science began to develop in the second half of the 20th century, including theory of automata, theory of computation, complexity theory, and algorithmic logic. All these fields play an individual role in developing knowledge about computational phenomena. An example is the automata theory, which supplies abstract machines which compute something (e.g. finite automata and pushdown automata), and complexity theory which categorizes problems arising in terms of the resources required to solve them, leading to such profound questions as the so-called P vs NP problem [4].

Up till recent years, TCS has been extended considerably. The new theoretical issues have emerged with introduction of quantum computing, probabilistic models, and artificial intelligence. Examples include comprehension of the capabilities and weaknesses of quantum algorithms, learning models in computational learning theory and a revision of the concept of complexity in terms of interactive computing or probabilistic computing [5]. Although very abstract, TCS has had practical impacts on the development of practical technologies. TCS is relevant to cryptographic protocols, optimizing compilers, database query language, and network protocols. Theory and practice do not interact just once but increasingly as computation makes itself felt more and more across the society [6].

The goal of this review is to give a thorough discussion of theoretical computer science with emphasis on its concepts, sub-disciplines, and open problems that still stimulate research. The article aims at informing the future of computing by exploring the historical and current tremendous changes to illustrate that the theory will remain relevant. In what follows we will explore the fundamental areas of TCS, examine their interactions with other fields, point out some of the principal new

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

developments, and share some of the outstanding open questions that delimit the current state of the field. With such investigation, we emphasize the importance of theory as a fundamental aspect of computer science both in the past, current and future [7].

CORE AREAS OF THEORETICAL COMPUTER SCIENCE

Theoretical Computer Science (TCS) has several foundational areas and each of these areas has its unique contribution to this subject. These fundamental areas are fundamental to computer science education and research and give fundamental tools to the study and understanding of the power and constraints of algorithms and machines. It explains five big branches: Automata theory and formal languages, computability and decidability, complexity theory, logic in computer science, and algorithm design and analysis [8]. The automata theory revolves around the study of abstract machines and what problems they are able to solve. Everywhere in the center is finite automata, pushdown automata and Turing machines, each of which define a different class of formal languages (regular, context-free, and recursively enumerable languages, respectively). Automata offer simplification of computation models which are fundamental in the development of interpreters, compilers and pattern recognition software [9].

Formal languages that are formed by string structures over finite alphabets can assist in the description of programming languages and can define the well-formed inputs to algorithms and systems. Regular expressions, to give an example, are directly connected with this theory and are commonly utilised in search engines, word parsing, and text assimilation. This is also the branch of study that is concerned with grammars, including Chomsky whose hierarchy is divided on the basis of the generative power of a language [10]. Knowledge of these classifications then enables us to reason about what kinds of problems can be effectively parsed in time and what problems will require a greater portion of the computation. This field deals with the basic question, What can be computed? Computability theory deals with questions that have efficient solutions to them, whereas the theory of decidability looks at whether a problem has an algorithm that accurately answers every possible question that might be posed [11].

The most well-known result of this field is the Halting Problem, which says that no algorithm to decide whether all program-input combinations on a program halt exists. These implications are profound in the fact that there are certain issues that are not within the computational dimensions, despite future improvement in hardware or software development [12]. This discipline takes inspiration heavily on the automata models that have been designed, particularly Turing machines, to determine the extents of algorithmic reasoning. Whereas computability addresses the question of solvability, the complexity theory addresses the question of the efficiency of solution of a problem.

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

This also involves grouping problems by the amount of time or space (resources) they consume. The most prevalent part of this region are classes of complexity like P, NP, EXP and PSPACE [13].

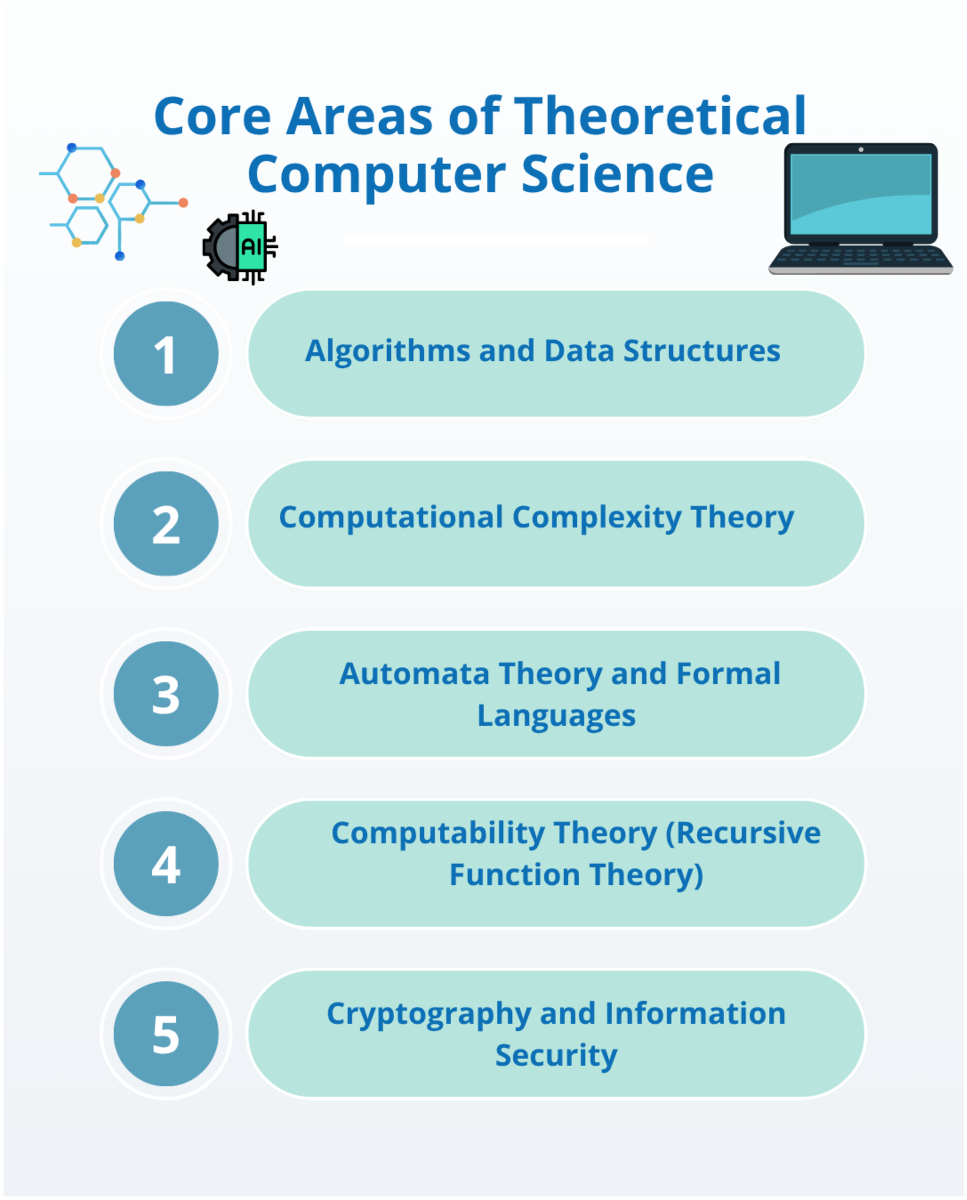
Core Areas of Theoretical Computer Science
AI
1
Algorithms and Data Structures
2
Computational Complexity Theory
3
Automata Theory and Formal Languages
4
Computability Theory (Recursive Function Theory)
5
Cryptography and Information Security

Figure: 1 showing core areas of theoretical computer science

The most well-known and not yet solved problem in this area is the P vs NP problem--whether every decision problem which can be "verified" in polynomial time can also be "solved" in polynomial time. This fundamental question of computer science has implications that extend to cryptography, optimization and artificial intelligence. Complexity theory investigates tradeoffs between deterministic and non-deterministic calculation and the place of randomness, as well as the boundaries of parallel and quantum computation [14]. Logic is the formal system in terms of which computations are specified and formal arguments made about them. All of these languages, such as propositional and first-order logic, temporal logic, and modal logic, are used to model and verify the

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

behavior of software and hardware systems [15].

Logical systems can be formally verified, which means that the correctness of an algorithm or system can be proved by means of mathematics. This would be critical especially where the problem is of a safety-critical system, e.g. aerospace or medical devices. Database query languages (e.g., SQL being based on predicate logic) and programming paradigms such as functional and logic programming are also based on logic [16]. Algorithms are the sequences of steps to be followed in deriving the solution of computational problems and any analysis pertains to the most practical part of TCS. Divide and conquer, greedy algorithms, dynamic programming, and network flows are some of the fundamental techniques used in both theory and practice of computing [17].

Another important aspect of TCS is that not only methods of designing efficient algorithms are developed but their correctness is also proved and their performance is analyzed in terms of time and space complexity. It also examines lower bounds, which are solutions that demonstrate what the smallest amount of resources that are needed to solve certain issues is. Also, this subject is concerned with approximation algorithms (to problems that are not easily solvable in a deterministic manner), online algorithms (where the input is received over time), and randomized algorithms (using probabilistic methods to find faster or simpler solutions) [18]. All of these five fundamental areas of theoretical computer science give a rigorous and broad view of computation. They provide means of analysis that determine what problems are solvable and how efficiently solvable as well as how to model and reason about systems that carry out computations. These subfields are well-theorized and there are many applications built around each, and are the bedrock that computer science is built upon [19].

CROSS-DISCIPLINARY INTERSECTIONS

TCS exists in the science of computer science, which is an absolute science. It has since become a much more interdisciplinary field involving mathematics, physics, economics, and biology, among others. These have been intersected which have enhanced TCS and made it open up to new forms of computation and new algorithms as well as new insights in understanding complex systems [20]. Some of the most visible areas overlapping across disciplines are Computational Geometry, Quantum Computation, Cryptography and Information Theory, and Computational Learning Theory. Computational geometry is concerned with algorithms related to geometric problems. It is predominant in computer graphics, robotics, computer-aided design (CAD), geographic information systems (GIS), and computer vision [20].

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

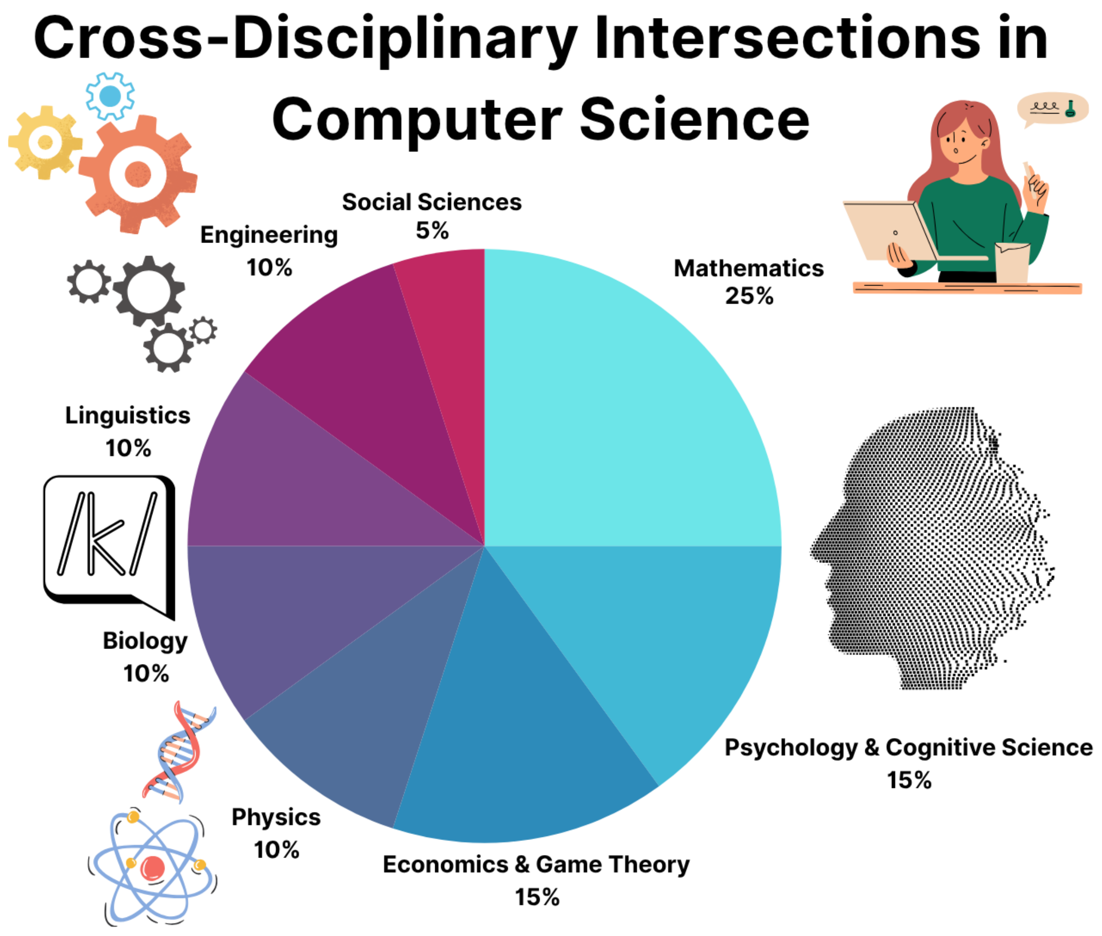

Figure: 2 showing disciplinary intersections in computer science

Theoretical study also plays a significant role in this area, to tune algorithms to large amounts of data, and to high dimensions. Algorithms in graph theory, combinatory and numerical techniques all intersect to give excellent, provably correct, methods of attacking geometric problems. Research in computational geometry has produced solutions to many difficult problems in geometry, even though it is a computational science, many problems in higher dimensional or complex constraints remain open [21]. Quantum computing is one of the most promising areas of theoretical computer science. It combines quantum physics with computation through offering models of computation where quantum mechanical effects like superposition, entanglement, and interference can be used [22].

Quantum Turing machine and quantum circuits are mathematical abstractions used to study quantum algorithms. Famous results in this field are Shor algorithm (factoring integer in polynomial time) and Grover algorithm (quadratic improvement of the speed of searching unsorted databases). Quantum complexity theory has also added novel complexity classes (BQP being one of them), and brought into question just how powerful quantum computation can be compared with classical models [23]. The theoretical and practical implications of the potential exponential speed-up of (some of) the problems that quantum machines can solve compared to classical machines are deep and profound in

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

such fields as cryptography, optimization and simulations of physical systems [24].

Cryptography is an applied and theoretical science that deals with protecting information. The more recent cryptographic techniques are very involved in number theory, algebra, and complexity theory. The belief that these problems are very difficult has been used to justify the security of popular measures like RSA and ECC. The all investigated theoretical aspects of one-way functions and zero-knowledge proofs as well as secure multi-party computation are core topics of modern cryptography. Such constructs enable secure computations of secrecy-sensitive data sharing and computations, as well as authentication [25].

About Claude Shannon, information theory has some overlap with TCS (e.g. in coding theory and data compression). It deals with the maximum capacity of storage and transmission of information, error-correcting codes and optimal codes. The merger of information theory with computational complexity brings forth the most fundamental questions about the efficiency of encoding and decoding data roots within imposed resource limitations [27]. The computational learning theory offers a theory that helps to understand the algorithm of machine learning. It aims to address questions like: Some of the key terms are PAC learning (Probably Approximately Correct learning), VC dimension (VapnikChervonenkis dimension), online learning. These concepts assist in formalizing the procedure of learning and in providing assurances on the behavior of learning processes [28].

RECENT ADVANCES IN THEORETICAL COMPUTER SCIENCE

The area of Theoretical Computer Science (TCS) has been developing very actively not only due to internal difficulties but also due to external changes in the sphere of technology. TCS has had a flood of new advances in recent decades due to the introduction of new models of computation, progress in understanding complexity, interdisciplinary influences with artificial intelligence, cryptography, biology, and quantum mechanics [29]. This chapter will point to some of the most significant recent developments and will address the ways in which such developments are transforming our conception of computation.

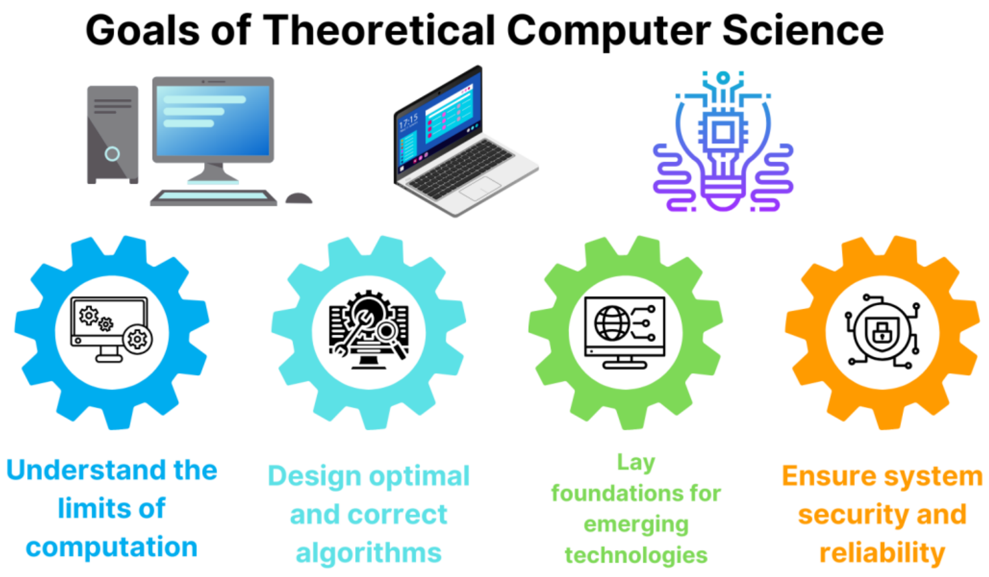
Goals of Theoretical Computer Science
Understand the limits of computation
Design optimal and correct algorithms
Lay foundations for emerging technologies
Ensure system security and reliability

Figure: 3 showing goals of theoretical computer science

Developments of the last few years one of the most important developments of recent years is the development of complexity theory, especially the fine-grained complexity of particular problems. Researchers have started to divide problems into classes not only between that which can be solved efficiently (polynomial time) and that which cannot (NP-hard), but also using time classes with finer distinctions [30]. It has given rise to the theory of fine-grained complexity which seeks to understand questions like whether, or not, Psolor? In other words, how much theoretical improvement can be made using certain hardness assumptions (like the Strong Exponential Time Hypothesis)? There also have been thrilling outcomes in circuit complexity, communication complexity and proof complexity, topics that consider the native hardness of computing functions and proving mathematical statements. Despite the continued failure to solve the central P vs. NP problem, other related questions have been developed by the community, including separating subclasses of P and lower best inquiry on constrained models of computation [31].

The interactive proofs and the zero-knowledge proposed protocols have transformed our knowledge in terms of verification and security. An interactive proof is a Turing machine written in terms of an interaction between a powerful prover and a verifier used to compute whether a particular statement is true. This suggestion has given rise to a major theoretical development the class IP (Interactive Polynomial time) has been proven equivalent to PSPACE, an unexpected equivalence that has broadened the frontiers of verifiability (with the help of interaction) [32]. Possibly with the most immense effect in approximation algorithms and hardness of approximation, probabilistic proof systems, particularly the Probabilistically Checkable Proofs (PCPs) have appeared. The PCP theorem

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

has been pivotal in establishing that many optimization problems have no approximations in approx. by use of its statement that any mathematical proof can be rewritten in such a way that, by simply checking few bits, one could be able to certify its correctness with high probability [33].

The advancements can also be found on the basis of zero-knowledge proofs where one party can provide a proof of the truth of the statement without revealing the reasoning behind it. This idea has been left theoretical and nowadays it has been put into actual practice in block chain technologies, privacy-preserving protocols, and digital identity systems. The theoretical advancements have also been aroused with the sudden interest in artificial intelligence and machine learning [34]. Learning theory new models of learning have been added to the computational learning theory, and include both agnostic learning and online learning, as well as differential privacy. Deep learning is a complex field, and theory, especially of deep learning, and simpler forms of neural networks, has just begun to be subjected to analysis using mathematical tools [35].

A pillar of TCS, optimization theory, has gotten ever more complex in order to satisfy the needs of contemporary AI applications. The theoretical contributions to convex optimization theory, stochastic gradient descent, and non-convex machine learning problems have been a vital ingredient in the process of understanding, and enhancing, machine learning algorithms. Quantum computing remains one of the liveliest areas of theoretical development [36]. The most significant advances on quantum algorithms, quantum walks, and Hamiltonian complexity, have enhanced our knowledge in relation to quantum advantage. In addition, the possible error correction in quantum states, quantum supremacy, and the foundations of quantum cryptography have received a lot of attention and have implications on safe interaction as well as future computing capabilities [37].

It is a very open question to what extent the quantum complexity classes such as BQP, QMA and others relate to classical ones. New findings have started to carve up this territory in a more precise way, which has started to clarify what quantum computers can and cannot do. Theoretical computer science hardly stands still-it is a living breathing field that will keep expanding as effected by the interaction of pure theory and practical problems [38]. Recent developments in complexity theory, interaction proofs, learning theory, and quantum computation not only have answered some fundamental theoretical questions, but also have had some impact on practice, systems and technologies. Such advancements indicate the continued vibrancy of TCS and its prominent position in determining the course of computation in the future [39].

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

UNSOLVED PROBLEMS AND UNSOLVED QUESTIONS

Although decades of intensive research have been done, Theoretical Computer Science (TCS) remains characterized by a few deepest and most difficult open problems. These unanswered questions do not merely demarcate the limitations of our existing body of knowledge, but they also act as maps that indicate our new research efforts. These issues are both mathematical in nature they relate to the boundaries of algorithmic power and more applied where their implications are broad based in cryptography, complexity theory, machine learning and quantum computing. This part shows some of the outstanding questions in TCS as it exists today [40].

Certainly, the most prominent and most mendacious open problem in theoretical computer science is the P vs NP question. It questions whether all the problems whose solutions may be checked fast (within a number of steps that grows as a power of a number) may also be solved fast. In other words, does this is such a crucial question in various aspects. A large number of practical problems are in NP but we do not know according to whether they can be efficiently solved or not is known. An argument either direction would be transformative in computer science: a proof of NP would give the argument that a huge category of computational problems is intrinsically hard [41]. In spite of great effort, however, no consensus has been achieved, and the issue encompassing it currently remains classified as one of the seven Millennium Prize Problems established by the Clay Mathematics Institute, and awarding a 1 million dollar prize to a solution [42].

The second is the desire to find good lower bounds in circuit complexity, which has a long history. The field inquires as to how massive or detailed a logical circuit should be prior to calculating a particular purpose. We can readily say that there are classes of problems that demand large circuits but finding and showing lower bounds of this type in a general and non-trivial manner has been a hard task. As an example, finding an NP-complete problem whose non-polynomial size Boolean circuit would immediately solve the P vs [43]. NP question could be done by showing that any Boolean circuit that solves an NP-complete problem must have a super-polynomial size. Likewise, we have a lack of lower bounds even (on circuit classes) which are restricted in some way (such as AC0 or NC1). These limitations are fundamental in defining the in-built complexity of compute problems [44].

As quantum computing grows up, another very important set of open problems is the connections between quantum and classical complexity classes. As an example: Settlement of these questions would shed the light on the question of quantum computers: whether they present a real computational advantage and to what degree. Also, quantum PCP conjecture is one of the most profound open problems in the field of quantum complexity, and has repercussions in quantum error correction, and

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

condensed matter physics [45]. Randomness has proven to be an effective way of enhancing the running time of many efficient algorithms, yet a central theoretical concern is whether the randomness is actually essential. Is there any general way to change any randomized algorithm into a deterministic one with only a small penalty in speed?

The DE randomization problem is the question of whether or not BPP = P (that is whether, given a problem with a solution in probabilistic polynomial time, one can efficiently find a deterministic solution). A positive solution would redefine the design of algorithms and strengthen the influence of deterministic computation. Even more related is the study of pseudorandom generators and how they can be based on hard problems [46]. Such objects are critical to cryptography, randomized machine theory, and theoretical computer science. Although machine learning has been performing exemplarily in practice, numerous theoretical grounds have not been settled. Questions like:

Theoretical computer science thrives on unsolved problems which drive its development and serve as the fuel to drive the invention of new techniques. These open problems, ranging over such topics as the foundational question of whether P vs NP, quantum computing, or machine learning, can be answered, become the boundaries of the frontier of knowledge, as well as what we dream of being able to know. When they finally are resolved, they will not only resolve profound theoretical arguments but could also be transformative as far as technology and the wider society are concerned [47].

PRACTICAL LIFE AND USAGE OF COMPUTER SCIENCE

Theoretical Computer Science (TCS) is deemed as an abstract and mathematically rigorous field, but its application in practical computing is far-reaching and extensive. Whether it is a secure form of intellectual communication, or even an efficient search algorithm, a great deal of modern technology has foundations rooted in decades-old theoretical contributions [48]. The section discusses the practice behind the most important theoretical notions and elucidates the role of TCS in intellectual advancement in software, hardware, cryptography, artificial intelligence, etc. Among the most pervasive and direct applications of TCS is the construction of efficient data structures and algorithms. All the cutting edge computer programs, including operating systems and web browsers, are based on algorithms that solve time and space problems [49].

Modes of attack including divide-and-conquer, dynamic programming and greedy algorithms developed in the practice and later in theory also became established in practice. Quick searching, sorting, storing, and retrieving of data is possible by using data structures, such as heaps, hash tables, trees, and graphs which have been designed and analyzed by rigorous theoretical classes of data structures. These data structure algorithm bases are very necessary in search therapy in real-time

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

systems, embedded application and massive scale data processing. One of the most influential contributions of TCS is the use of cryptography [50]. Online transaction security, data integrity, privacy preservation protocols are based on what is theoretically known in number theory, algebra and complexity theory. Theoretical systems like zero-knowledge proofs, homomorphic encryption and secure multi-party computation have moved out of the academic theory landscape into practical systems, affecting fields such as block chain technology, privacy-preserving machine learning and confidential data processing [51].

The issue of ensuring that software behaves is relevant at greater level, especially in security-sensitive systems, where so much of that happens is in medical devices, autonomous vehicles, and aerospace control systems. In this setting the formal methods, which are logical and automata theories, enable the engineers to demonstrate how a system satisfies its specification. Model checkers, theorem proves, formal specification languages (e.g., TLA +, Coq, Z3), and languages and tools in other areas (e.g., Petri nets, process calculi, and type state (e.g., language constructs, tools, and libraries are derived using principles of TCS [52]. They are applied to prove the correctness of hardware circuits (e.g., processor validation, operated by Intel), as well as of complex software protocols (e.g., routing protocols in networks, or smart contracts).

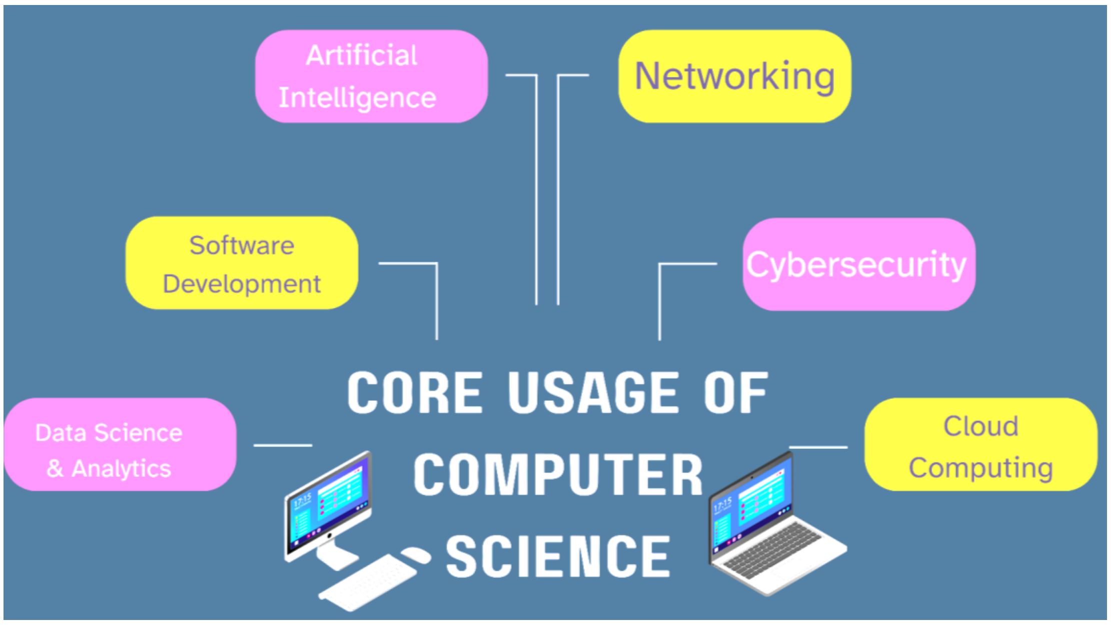

Figure: 4 showing core usage of computer science

Enterprise systems, e-commerce websites are just some of the many things based on relational database systems, where much goes back to TCS. Relational algebra, relational query languages such as SQL, and relational optimization owes its development to formal logic and to automata theory. Theoretical analysis is used to improve query plans, by performing inferences about equivalences and performance guarantees, allowing database engines to optimize functions of equivalences, which are

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

used to derive query plans [53]. Theoretical mathematical models also stimulate transaction management, consistency models, concurrency control, through the serial inability theory and the distributed consensus algorithm. Although AI has been known to be linked to heuristics based on data, theoretically, it is associated with TCS. The computational learning theory stipulates what may be learned based on the data, the number of samples required and the quality of generalization of a model [54].

Efficient and reliable learning algorithms can be developed by using theoretical models, such as, PAC learning, VC-dimension, and the online learning frameworks. Gradient descent optimization algorithms turn out to be heavily utilized in neural networks, and can be analyzed using convex optimization as well as complexity theory. Theoretical based tools are also being adopted to address fairness, explaining ability and robustness as AI systems become more integrated into the society [55]. Even the internet itself is constructed on the basis of the theoretical understandings. Examples are routing algorithms and data compression, error correcting codes, and verification of protocols which all come out of the fundamentals of TCS.

Distributed systems Design Distributed systems have several research topics that can be characterized as solving consensus and fault tolerance and synchronization challenges with theoretical models, including CAP theorem, Byzantine agreement, and distributed automata. Such notions become essential to such trends as cloud computing, block chain technologies, and real-time communication platforms. Quite the contrary, Theoretical Computer Science is not a narrow academic interest; the incredible range of dependents and products of Theoretical Computer Science, made possible by its use of algorithms, is what mostly drives the modern digital world forward [56]. It does not only offer the tools, it offers the guarantees, on which the reliability, security and efficiency of computing systems are based. The necessity of sound and theoretically rendered approaches keeps only increasing along with the development of technology, which is another argument that theory and practice cannot be turned against each other, rather they are the allies in innovation [57].

FUTURE DIRECTIONS AND FUTURE CHALLENGES

The existence and development of TCS is viable because it is expanding as a living and active science using and relying upon the principles. At the same time, the increasing level of complex computing problems being faced by the technological, science and society has some emerging questions that demand an emergence of new theoretical models and tools. The future of TCS does not just lie in resolution of such traditional problems as P vs. NP, but also in research of new and emerging directions in fields of intersection with quantum mechanics, biology, economics, ethics and artificial intelligence [58]. The next part will provide some of the promising avenues of the future theorizing

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

areas that is likely to take place within the theoretical research direction in the coming decades.

Due to the fast development of quantum hardware, exploring quantum algorithms and quantum complexity classes, as well as quantum error mitigation, has become increasingly important than ever before to do through the theory. In the scientific literature, the main direction of future work on quantum TCS may be focused on the following: Establishing some demarcation between the classical and quantum complexity classes (e.g. BQP vs. NP) [59]. Creating quantum algorithms beyond the Shor and Grover that break more solution problems faster with real applications. Creation of post-quantum cryptography capable of resisting quantum computer attack including lattice-based cryptography, hash-based cryptography and code-based cryptography. Quantum PCP conjecture and quantum interactive proofs are other open problems that are highly active research fields, and whose applications may turn out to be impactful in the context of computational theory as well as condensed matter physics [60].

The urgency of such theoretical understating is also explained by the fact that AI systems are getting more potent. Future of TCS in this region involves: The description of behaviours of deep learning models, especially in the generalization, expressivity and convergence. Development of provably robust and interpretable models which has the potential of making fair models, interpretable and robust to adversary attacks. Causal inference development theory, reinforcement learning and self-supervised learning theory [61]. Enhancements in learning theory, e.g. making statements about what cannot and/or cannot be learned with respect to high-dimensional problems, or when data is limited. Another problem is to find the compromise between the real-world deep learning and the computational learning theory that is also one of the possible future theoretical works [62].

Algorithms have been implicated in among others making crucial decisions and hence, the increasing need that TCS deals with the ethical sides of doing computations. Future studies are comprised of: Creation of algorithms such that they satisfy substantive fairness standards within the demographic groups. Continuing to test out the additional concepts regarding differential privacy and accomplishing the endeavor of achieving private learning [63]. A survey of the nature of trade-offs between accuracy, fairness and privacy in an algorithmic complexity-theoretic view. Ensuring that bias detection and mitigation were codified in the computational structures. This form of work includes blending designs of algorithm with social science and law and legal models-which is both a fascinating and socially useful possibility of TCS [64].

Broadly, natural-computing systems are any biological systems whose nature is that of natural computing; and this covers the brain or cellular networks of a human being. On this basis, Computer computation is being re-invented using biological forms of computation: DNA computing and

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

molecular programming respectively theorize that simple chemical reactions should be used as the computer [65]. New models of membrane computing and reaction networks provide an opportunity to make new abstractions of algorithmic processes in nature. Such growth of understanding how organic beings can process information could lead to entirely new paradigm of computation. This next line of research can be applied in the origin of life, synthetic biology and nanotechnology [66].
The analysis of traditional algorithms is wildly in regards to the worst-case performance, yet the performance in the reality can be estimated to be higher. Besides worst-case analysis, a trend is to come up with more realistic models, including how to model a realistic example: smoothed analysis, parameterized complexity, and instance-optimal algorithms. Fine-grained complexity theory the fine-grained complexity theory strives to determine sharp complexity thresholds and to define fine-grained reductions. This interdependency of such complexities can be useful to be aware of to offer hopes of bottlenecks and optimization in practice in as far as algorithms are concerned [67].
Computing is also a social and interactive practice even in the future. The cross-system work is increasingly designed so as to be able to interact with the human and other systems under varying conditions: The interactions between the human and the computer become an essential component of the adaptive processes. The study of mechanism design and computational economics, and game theory will help in the construction of systems that are sensitive to humanistic motive and strategic behavior [68]. The theoretical analysis of interactive proof systems, dialogue systems and algorithmic game theory will become relevant to the theoretical analysis and construction of robust and adaptive computation systems.
Theoretical Computer Science is not a finished science in the least. In the response to the need of the emerging technologies, moral needs and the fields of knowledge, it is dynamically expanding to fill the new grounds. Future TCS should be able to adapt opposites between strict mathematical reasoning on one side and demands of the society, technological shortcomings and innovative abstractness [69]. The virtual space has never been so full of action and sharing so we need to understand that the more we develop our theoretical systems today the more secure, trusting and fair our systems that will govern the world tomorrow will be.

CONCLUSION

Theoretical Computer Science (TCS) is the theoretical part of computing; its role is to give a rigorous structure of such kind that it can be known what the basic limits and possibilities of computation are. Though it may be viewed as abstract, TCS finds application across all subfields of computer science such as those pertaining to the design of algorithms, cryptography, artificial intelligence, and quantum computing. This review has traced the vital assumptions of TCS, displayed Trans and multi-

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

disciplinary overlapping zones, existing breakthroughs, current challenges, and given the directions which signal to its destiny. It has its origins in its fundamental areas, among which are automata theory, computability theory, complexity theory, logic and algorithm analysis, whose language and techniques are the basic ones with which to reason computationally.

Such disciplines touch upon such issues like: what problems can be solved? By what means are they well-solvable? What are the models that could express the essence of computation? The cleanliness and precision guided by such disciplines does not just question the theoretical knowledge but also the framework of effective and reliable systems in the real world. TCS is not just being shone upon in the academic one. Its and its combination with other areas of knowledge like in geometry, quantum mechanics, information theory and machine learning have led to whole new areas of study. The fields of computational geometry can further advance robotics and graphics and quantum computation is undertaking radical revamps as to the standard pre-existing ideas on what can be computed and even learning theory can further assist in emotional company to the artificial intelligence. The intersections are the witnesses to the fact that the field can be evolved and shape the technology which is not within its scope.

In the past couple of years, TCS has experienced a surge of innovations that make it more pertinent and important than ever. Studies of complexity of fine-grained complexity, probabilistic proof systems, zero-knowledge protocols, and in the theory of machine learning have led the way in terms of both what can be achieved in theory, and in practice. Meanwhile, emerging trends of thought in quantum computing have revealed new paradigms of computing destined to alter how secured communication along with abilities to resolve problems are conducted in near future. What is also characteristic of TCS is the character of its open problems where there is a mystery of it. The P vs NP problem, lower complexity barriers in circuit complexity, and quantum versus classical computing should be among the classic questions in computer science that have long been in the domain of experts. The fact that these problems remain open is not only the reason why the research in this direction remains in progress but also the reason why one should learn more about boundaries of algorithmic reasoning that are closer to the inside of it. Their final solution will be associated with a lot of implication in various disciplines, including cryptography, optimization, and data privacy.

As can be observed, TCS has never stopped having an impact on actual work in computing. In common technologies there are theoretical contributions, whether they are about algorithm and data structures or cryptographic protocols and formal verification tools. The areas of TCS contained the underpinning concepts concerning software accuracy, safe interaction, and effective database frameworks and expandable The domains of TCS comprised the underpinning beliefs of software

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

precision, safe correspondence and effective database framework and saleable networks. The phenomenon shows that even most abstractive theories could be employed in implementation and that could result in radical transformation. Going into the future, TCS will be able to fix some of the most relevant problems of our age. As AI starts exercising even greater power, theoretical tools development is going to play a pivotal role in ensuring such systems are fair and comprehensible and safe. TCS will be on the forefront providing us with new models of computation with quantum technologies maturing.

As individuals are growing even more worried with respect to their privacy, morals, and algorithmic-bias, theory will offer the means of developing transparent and accountable systems. In addition, due to the proliferation of computational studies to biological, chemical systems, new models will be created, inspired by nature, which will demand new theoretical methods. Fitting in between bouts of speculation and utility, thinking and computing, reasoning and education, is the next generation of Theoretical Computer Science. Its ability to contend, develop and innovate is what makes it a component of computer science that will keep remaining a very essential and live part of the computer science field. It comes not as an academicism to students, but is of immense relevance to devising ways and means that would define our digital age to the researcher and practitioner. Theoretical Computer Science does not only focus on what the computers can do but also on what, why and how the computer does it. The compass is the ruler in the computing journey, in what is done and what can be done.

REFERENCES

[1]. Da Cruz Alves, N., Von Wangenheim, C., Hauck, J., Borgatto, A., and de Andrade, D. (2020). An item response theory analysis of the sequencing of algorithms & programming concepts. CoolThink@ JC 9, 1–11. doi: 10.5753/educomp.2021.14466

[2]. C. Johnson, R. Moorhead, T. Munzner, H. Pfister, P. Rheingans, and T. S. Yoo, NIH/NSF Visualization Research Challenges Report, IEEE Computing Society, Los Alamitos, CA, 2006, http://tab.computer.org/vgtc/vrc/NIH-NSF-VRC-Report-Final.pdf. (Cited on p. 733)

[3]. D. Keyes, V. Taylor, et al., National Science Foundation Advisory Committee on CyberInfrastructure, Task Force on Software for Science and Engineering, final report, 2011, http://www.nsf.gov/cise/aci/taskforces/TaskForceReportSoftware.pdf. (Cited on pp. 734, 737)

[4]. S. Zhang, Q. Liu, W. Chen, Q. Wang, Z. Huang, Interactive networks and social knowledge construction behavioral patterns in primary school teachers' online collaborative learning activities, Computers and Education 104 (2017) 1–17. doi:10.1016/j.compedu.2016.10.011.

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

[5]. S. O. Semerikov, T. A. Vakaliuk, I. S. Mintii, V. A. Hamaniuk, O. V. Bondarenko, P. P. Nechypurenko, S. V. Shokaliuk, N. V. Moiseienko, Designing an immersive cloud-based educational environment for universities: a comprehensive approach, CEUR Workshop Proceedings 3844 (2024) 107–116

[6]. F. Garzotto, E. Beccaluva, M. Gianotti, F. Riccardi, Interactive Multisensory Environments for Primary School Children, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI ’20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1–12. doi:10.1145/3313831.3376343.

[7]. I. Nicolaidou, E. Stavrou, G. Leonidou, Building primary-school children’s resilience through a web-based interactive learning environment: Quasi-experimental pre-post study, JMIR Pediatrics and Parenting 4 (2021) e27958. Doi: 10.2196/27958.

[8]. S. S. Korniienko, P. V. Zahorodko, A. M. Striuk, A. I. Kupin, S. O. Semerikov, A systematic review of gamification in software engineering education, CEUR Workshop Proceedings 3844 (2024) 83–95

[9]. A. V. Riabko, T. A. Vakaliuk, O. V. Zaika, R. P. Kukharchuk, I. V. Novitska, Gamification method using Minecraft for training future teachers of computer science, CEUR Workshop Proceedings 3771 (2024) 22–35.

[10]. Chaczko, Z., and Braun, R. (2017) Learning data engineering: Creating IoT apps using the node-RED and the RPI technologies. 2017 16th Int. Conf. Inf. Technol. Based High. Educ. Training, ITHET 2017.

[11]. Klinger, T., and Madritsch, C. (2015) Collaborative learning using pocket labs. Proc. 2015 Int. Conf. Interact. Mob. Commun. Technol. Learn. IMCL 2015, (November), 185–189.

[12]. Saari, M., bin Baharudin, A.M., and Hyrynsalmi, S. (2017) Survey of Prototyping Solutions Utilizing Raspberry Pi. 2017 40TH Int. Conv. Inf. Commun. Technol. Electron. Microelectron., 991–994

[13]. Mahmood, S., Palaniappan, S., Hasan, R., Sarker, K.U., Abass, A., and Rajegowda, P.M. (2019) Raspberry PI and role of IoT in Education. 2019 4th MEC Int. Conf. Big Data Smart City, 1–6.

[14]. Agatolio, F., Pivetti, M., Menegatti, E., and Moro, M. (2016) a training course in educational robotics for learning support teachers. Adv. Intell. Syst. Comput., 560 (Llc), 43–57.

[15]. Ramirez-Garibay, F., Olivarria, C.M., Eufracio Aguilera, A.F., and Huegel, J.C. (2014) MyVox—Device for the communication between people: blind, deaf, deaf-blind and unimpaired. IEEE Glob. Humanit. Technol. Conf. (GHTC 2014), 506–509.

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

[16]. Wagh, P., Prajapati, U., Shinde, M., Salunke, P., Chaskar, V., Telavane, S., and Yadav, V. (2016) E-Braille-a self-learning Braille device. 2016 Twenty Second Natl. Conf. Commun., 1–6.

[17]. Srivastava, A., and Dawle, S. (2015) Mudra: a multimodal interface for braille teaching. Proc. 6th Augment. Hum. Int. Conf., 169–170.

[18]. Gustavsson, I., Nilsson, K., Zackrisson, J., Garcia-Zubia, J., Hernandez-Jayo, U., Nafalski, A., Nedic, Z., Göl, Ö., MacHotka, J., Pettersson, M.I., Lagö, T., and Håkansson, L. (2009) On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories. IEEE Trans. Learn. Technol., 2 (4), 263–274.

[19]. Lin, K. Y., Wu, Y. T., Hsu, Y. T., & Williams, P. J. (2021). Efects of infusing the engineering design process into STEM project-based learning to develop preservice technology teachers' engineering design thinking. International Journal of STEM Education, 8, 1–15.

[20]. Liu, X. (2021). Research on the Design and Application of Project-Based Teaching in High School Information Technology Curriculum [D]. Nanning Normal University. https://doi.org/10.27037/d.cnki.Ggxsc.2021.000279

[21]. Lockwood, J., & Mooney, A. (2017). Computational thinking in education: Where does it Fit? A systematic literary review. arXiv preprint arXiv:1703.07659.

[22]. Lu, J. (2021). Research on the Design of Project-Based Learning Activities for First-Year Programming Courses Aimed at Cultivating Computational Thinking [D]. Nanning Normal University. https://doi.org/10.27037/d.cnki.ggxsc.2021.000202

[23]. Luo, Z., Yu, Y., & L, D. (2005). Continuously improving teaching quality by adhering to the scientifc development concept: Collection of papers on undergraduate teaching reform at University of Electronic Science and Technology. University of Electronic Science and Technology Press.

[24]. Ma, J., Zhang, Y., Bin, H., Wang, K., Liu, J., & Gao, H. (2022). The Development of Students' Computational Thinking Practices in AI Course Using the Game-Based Learning: A Case Study. In 2022 International Symposium on Educational Technology (ISET) (pp. 273–277). IEEE.

[25]. Manches, A., & Plowman, L. (2017). Computing education in children's early years: A call for debate. British Journal of Educational Technology, 48(1), 191–201.

[26]. E. Polat, Gamification implementation for educational purposes: a scoping review (2013-2018), Educational Technology Quarterly 2023 (2023) 367–400. doi:10.55056/etq.589.

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

[27]. M.-H. M. Chen, S.-T. Tsai, C.-C. Chang, Effects of educational role-playing and simulation games: Designing interactive carbon footprint curriculum for primary school students, Journal of Research in Education Sciences 61 (2016) 1–32. doi:10.6209/JORIES.2016.61 (4).01.

[28]. Borna, K., & Rad, H. M. (2018). Serious games in computer science learning goals. In 2018 2nd National and 1st International Digital Games Research Conference: Trends, Technologies, and Applications (DGRC) (pp. 161–166). IEEE. https://doi.org/10.1109/DGRC.2018.8712030

[29]. Boyer, S. L., Edmondson, D. R., Artis, A. B., & Fleming, D. (2014). Self-directed learning: A tool for lifelong learning. Journal of Marketing Education, 36(1), 20–32. https://doi.org/10.1177/0273475313494010

[30]. Boyle, E. A., Hainey, T., Connolly, T. M., Gray, G., Earp, J., Ott, M., Lim, T., Ninaus, M., Ribeiro, C., & Pereira, J. (2016). An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. Computers & Education, 94, 178–192. https://doi.org/10.1016/j.compedu.2015.11.003

[31]. Calvo-Morata, A., Alonso-Fernández, C., Freire, M., Martínez-Ortiz, I., & Fernández-Manjón, B. (2020). Serious games to prevent and detect bullying and cyberbullying: A systematic serious games and literature review. Computers & Education, 157, 103958. https://doi.org/10.1016/j.compedu.2020.103958

[32]. Campbell, L. (2020). Teaching in an inspiring learning space: An investigation of the extent to which one school's innovative learning environment has impacted on teachers' pedagogy and practice. Research Papers in Education, 35(2), 185–204. https://doi.org/10.1080/02671522.2019.1568526

[33]. Cardinot, A., & Fairfeld, J. A. (2019). Game-based learning to engage students with physics and astronomy using a board game. International Journal of Game-Based Learning, 9(1), 42–57. https://doi.org/10.4018/IJGBL. 2019010104

[34]. Chandel, P., Dutta, D., Tekta, P., Dutta, K., & Gupta, V. (2015). Digital game-based learning in computer science education. CPUH-Research Journal, 1(2), 33–37.

[35]. Chang, C. S., Chung, C. H., & Chang, J. A. (2020). Infuence of problem-based learning games on efective computer programming learning in higher education. Educational Technology Research and Development, 68, 2615–2634. https://doi.org/10.1007/s11423-020-09784-3

[36]. Chang, C. K., & Tsai, Y. T. (2018). Pair-programming curriculum development of motion-based game for enhancing computational thinking skills. In 2018 7th International Congress

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

on Advanced Applied Informatics (IIAIAAI) (pp. 284–287). IEEE. https://doi.org/10.1109/IIAI-AAI.2018.00061

[37]. Chen, P. Y., Hwang, G. J., Yeh, S. Y., Chen, Y. T., Chen, T. W., & Chien, C. H. (2021). Three decades of game-based learning in science and mathematics education: An integrated bibliometric analysis and systematic review. Journal of Computers in Education, 9(3), 455–476. https://doi.org/10.1007/s40692-021-00210-y

[38]. Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J., & Wang, X. (2023). Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. International Journal of STEM Education, 10(1), 1–25. https://doi.org/10.1186/s4059402300434-7

[39]. Cheng, Y. P., Lai, C. F., Chen, Y. T., Wang, W. S., Huang, Y. M., & Wu, T. T. (2023). Enhancing student's computational thinking skills with student-generated questions strategy in a game-based learning platform. Computers & Education, 200, 104794. https://doi.org/10.1016/j.compedu.2023. 104794

[40]. Clark, D. B., Tanner-Smith, E. E., & Killingsworth, S. S. (2016). Digital games, design, and learning: A systematic review and meta-analysis. Review of Educational Research, 86, 79–122. https://doi.org/10.3102/0034654315 582065

[41]. Clegg, B. S., Rojas, J. M., & Fraser, G. (2017). Teaching software testing concepts using a mutation testing game. In 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET) (pp. 33–36). IEEE. https://doi.org/10.1109/ICSE-SEET.2017.1

[42]. Corda, F., Onnis, M., Pes, M., Spano, L. D., & Scateni, R. (2019). BashDungeon: Learning UNIX with a video-game. Multimedia Tools and Applications, 78(10), 13731–13746. https://doi.org/10.1007/s11042-019-7230-3

[43]. Daungcharone, K., Panjaburee, P., & Thongkoo, K. (2017). Using digital game as compiler to motivate C programming language learning in higher education. In 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 533–538). IEEE. https://doi.org/10.1109/IIAIAAI.2017.77

[44]. De Carvalho, C. V., Cerar, Š, Rugelj, J., Tsalapatas, H., & Heidmann, O. (2020). Addressing the gender gap in computer programming through the design and development of serious games. IEEE Revista Iberoamericana De Tecnologias Del Aprendizaje, 15(3), 242–251. https://doi.org/10.1109/RITA.2020.3008127

[45]. De Kereki, I. F., & Adorjan, A. (2018). Serious games: Using abstract strategy games in computer science 2: An experience report and lessons learned. In 2018 IEEE Global Engineering Education Conference (EDUCON) (pp. 169–174). IEEE. https://doi.org/10.1109/EDUCON.2018.8363224

[46]. De Troyer, O., Lindberg, R., Maushagen, J., & Sajjadi, P. (2019). Development and evaluation of an educational game to practice the truth tables of logic. In 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT) (pp. 92–96). IEEE. https://doi.org/10.1109/ICALT.

[47]. 2019.00032 Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? Computers & Education, 58(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

[48]. Díaz, J., López, J. A., Sepúlveda, S., Ramírez Villegas, G. M., Ahumada, D., & Moreira, F. (2021). Evaluating aspects of usability in video game-based programming learning platforms. Procedia Computer Science, 181, 247–254. https://doi.org/10.1016/j.procs.2021.01.141

[49]. Dočkalová Burská, K., Rusňák, V., & Ošlejšek, R. (2022). Data-driven insight into the puzzle-based cybersecurity training. Computers & Graphics, 102, 441–451. https://doi.org/10.1016/j.cag.2021.09.011

[50]. Dos Santos, A. L., Souza, M. R. D. A., Dayrell, M., & Figueiredo, E. (2019). A systematic mapping study on game elements and serious games for learning programming. In B. M. McLaren, R. Reilly, S. Zvacek, & J. Uhomoibhi (Eds.), Computer supported education (Vol. 1022, pp. 328–356). Springer. https://doi.org/10.1007/978-3-030-21151-617

[51]. Duch, P., & Jaworski, T. (2018). Enriching computer science programming classes with Arduino game development. In 2018 11th International Conference on Human System Interaction (HIS) (pp. 148–154). IEEE. https://doi.org/10.1109/HSI.2018.8430994

[52]. Eleftheriadis, S., & Xinogalos, S. (2020). Ofce Madness: Design and pilot evaluation of a serious game for learning the C++ programming language. In I. Marfsi-Schottman, F. Bellotti, L. Hamon, & R. Klemke (Eds.), Games and learning alliance (pp. 389–394). Springer. https://doi.org/10.1007/978-3-030-63464-3_36

[53]. Elmunsyah, H., Kusumo, G. R., Pujianto, U., & Prasetya, D. D. (2018). Development of mobile based educational game as a learning media for basic programming in VHS. In 2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 416–420). IEEE. https://doi.org/10.1109/EECSI.2018.8752658

[54]. X. Guo, M. Wu, An empirical study of rural primary school students' English knowledge construction based on Interactive Mobile Learning application, in: Proceedings of the 2018 International Conference on Distance Education and Learning, ICDEL '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 6–10. doi:10.1145/3231848.3231852.

[55]. K. K. Gupta, S. Agarwal, A. Srivastava, R. Sah, A Sustainable Approach Toward Tangible Interactive Setup for Improving the Learning Experience of Primary School's Children in Rural India, in: B. B. V. L. Deepak, M. V. A. R. Bahubalendruni, D. R. K. Parhi, B. B. Biswal (Eds.), Recent Trends in Product Design and Intelligent Manufacturing Systems, Lecture Notes in Mechanical Engineering, Springer Nature Singapore, Singapore, 2023, pp. 205–215. doi:10.1007/978-981-19-4606-6_21

[56]. Baer, J., & Kaufman, J. C. (2005). Bridging generality and specificity: The amusement park theoretical (APT) model of creativity. Roeper Review, 27(3), 158–163. doi:10.1080/02783190509554310

[57]. Batey, M., & Furnham, A. (2006). Creativity, intelligence, and personality: A critical review of the scattered literature. Genetic, Social, and General Psychology Monographs, 132 (4), 355–429. doi:10.3200/MONO.132.4.355-430

[58]. Berman, A., & James, V. (2018). Learning as performance: Autoencoding and generating dance movements in real time. In A. Liapis, J. J. R. Cardalda, & A. Ekárt (Eds.), International Conference on Computational Intelligence in Music, Sound, Art and Design (pp. 256–266). Springer. doi:10.1007/978-3-319-77583-8.

[59]. Biggs, S. (2009). New media: The "first word in art? In H. Smith & E. T. Dean (Eds.), Practice-led research, research-led practice in the creative arts (pp. 66–83). Edinburgh University Press.

[60]. Boden, M. A. (2004). The creative mind: myths and mechanisms (2nd ed.). New York, NY: Routledge. Boden, M. A. (2009). Computer models of creativity. AI Magazine, 30(3), 23–34. doi:10.1609/aimag.v30i3.2254

[61]. Boden, M. A. (2016). AI: Its nature and future. Oxford University Press. Botella, M., Glaveanu, V., Zenasni, F., Storme, M., Myszkowski, N., Wolff, M., & Lubart, T. (2013). How artists create: Creative process and multivariate factors. Learning and Individual Differences, 26, 161–170. doi:10.1016/j.lindif.2013.02.008

[62]. Bray, L., & Bown, O. (2016). Applying core interaction design principles to computational creativity. In F. Pachet, A. Cardoso, V. Corruble, & F. Ghedini (Eds.), Proceedings of the

ISSN : 3079-6830
Volume 1, Number 3, 2025
https://doi.org/10.70445/giaic.1.2.2025.20-43

Seventh International Conference on Computational Creativity (ICCC 206) (pp. 93–97). Paris, France: Sony CSL.

[63]. S. Nusir, I. Alsmadi, M. Al-Kabi, F. Shardqah, Designing an interactive multimedia learning system for the children of primary schools in Jordan, in: 2011 IEEE Global Engineering Education Conference, EDUCON 2011, 2011, pp. 45–51. doi:10.1109/EDUCON.2011.5773111.

[64]. N. Abdul Rawi, A. R. Mamat, M. S. Mat Deris, M. Mat Amin, N. Rahim, A novel multimedia interactive application to support road safety education among primary school children in Malaysia, Jurnal Teknologi 77 (2015) 75–81. doi:10.11113/jt.v77.6516.

[65]. D. E. Keyes, P. Colella, T. H. Dunning, Jr., and W. D. Gropp, eds., A Science-Based Case for Large-Scale Simulation, Office of Science, U.S. Department of Energy, 2003 http://www.pnl.gov/scales/docs/volume1300dpi.pdf. (Cited on pp. 711, 726)

[66]. Daungcharone, K., Panjaburee, P., and Thongkoo, K. (2019). A mobile game-based C programming language learning: results of university students’ achievement and motivations. Int. J. Mob. Learn. Organ. 13, 171–192. doi: 10.1504/IJMLO.2019.0 98184

[67]. De Paula, B., Burn, A., Noss, R., and Valente, J. (2018). Playing beowulf: bridging computational thinking, arts and literature through gamemaking. Int. J. Child Comp. Interact. 16, 39–46. doi: 10.1016/j.ijcci.2017. 11.003

[68]. Fotaris, P., Mastoras, T., Leinfellner, R., and Rosunally, Y. (2016). Climbing up the leaderboard: an empirical study of applying gamification techniques to a computer programming class. Electro. J. e-learn. 14, 94–110.

[69]. García, F. (2018). Editorial computational thinking. IEEE Rev. Iberoam. Tecnol. Aprendizaje Salamanca: University Press. 13, 17–19. doi: 10.1109/RITA.2018.2809939