Mirzakhmet B. Syzdykov[1], Yannick Leon Kardeis[2]
[1]Satbayev University, Almaty, Kazakhstan
[2]Rheinland-Pfalzische Technische Universitat Kaiserslautern-Landau, Pfalz, Germany
*e-mail: mirzakhmets@icloud.com, Yannickkardeis@gmail.com

On Experimental Proof of "P versus NP" Theorem

We propose a simple and intuitive algorithm for solving md-DFA problem using algorithm concepts within extended operators, our approach shows quadratic polynomial time and hence proves the equivalence between polynomial and non-polynomial classes, we have also shown that minimal non-emptiness of automata problem can be solved in polynomial time with help of modified subset construction, rather that building a product automaton, which lead to factorial size of the memory and time, in this work we also have used many non-tractable existing examples and computed them in polynomial time, which guarantees that our algorithm solves NP-complete problem in almost linear polynomial time, we have also avoided the problem of product automata by an algorithmic approach, we are also giving the starting ground for the proof of back-reference problem which was discussed before, notion to the globally local increment is also given as the main argument towards the resolution of "P versus NP" theorem, which coincides with the finitarity term in general mathematics.
Keywords: P versus NP, complexity, theorem, experimental, proof.

1 Introduction

The NP-hardness was first defined in [1], also there's a defined lower linear bound for deciding arbitrary non-deterministic finite automata on regular languages or even other arbitrary [2]. The problem was first seen on the partial case of non-deterministic automata [3, 4]. The problem of md-DFA is to find a minimal finite automaton which is a subset of any given automata and isn't included in others [5].

The relationship between P vs. NP is one of the greatest open problems in computer science. The central challenge is whether problems whose solutions can be efficiently verified (NP) can also be efficiently computed (P). Here, we propose a new perspective: Is there a deeper mathematical structure that enables a more efficient computation of NP problems? We investigate whether parallels exist between quantum mechanics, the fractal structure of the Riemann zeta function, and the superposition of NP problems.

NP-Completeness in DFA Problems An important class of NP-complete problems arises from automata-based computation. Two key problems are: - Minimal Distinguishing DFA Problem: Determining the smallest deterministic finite automaton (DFA) that distinguishes between two regular languages. - DFA Non-Emptiness Problem: Deciding whether the intersection of multiple DFAs is non-empty. Both problems are NP-complete [5, 7]. - The product construction for DFAs has a complexity of $O(|A_1| * ... * |A_n|)$, which grows exponentially with the number of automata. - A modified subset construction can help reduce the complexity, but a fundamental lower bound remains. Question: Is there a hidden structure that allows for more efficient computation?

Superposition of NP Problems In classical computation, NP problems are solved sequentially: all possible solutions must be explicitly checked one by

one. In quantum mechanics, states exist in superposition: - A quantum system can exist in multiple states simultaneously until a measurement collapses it to a single definite state. - Quantum computers could solve NP problems more efficiently by evaluating all solutions simultaneously and amplifying the optimal one (e.g., using Grover's algorithm). Hypothesis: NP problems are not randomly distributed but follow a hidden mathematical structure that enables a more efficient computation.

1.1 Connection to the Zeta Function Fractal Structure & Superposition

According to Kardeis [14], the analytic continuation of the Riemann zeta function exhibits remarkable symmetry: - The function has poles at s = 1 and possibly at s = 0, as supported by the functional equation. - The critical line $R(s) = 0.5$, $R(s) = 0.5$ contains infinitely many nontrivial zeros, reflecting the structure of prime numbers. - The self-similarity of the zeta function suggests a fractal order in its structure. A key point in Kardeis' work is the hypothesis that the structure of the zeta function resembles a superposition of states: - The statement "0 = 1 simultaneously like a superposition suggests that the zeros of the zeta function represent a simultaneous existence of multiple solutions. - This directly corresponds to the idea that NP problems do not need to be solved sequentially but can be structured within a higher-order fractal framework.

Hypothesis: The zeta function may reflect a deeper order in NP problems, enabling a more efficient computation.

Connecting DFA, Quantum Mechanics, and the Zeta Function DFA & NP problems are exponentially complex. - Classical algorithms require sequential computation. - Superposition in quantum mechanics allows for parallel states.

The zeta function exhibits a fractal order: - The self-similarity of its zeros and their reflection symmetry could serve as a mathematical analogue to quantum superposition. - This suggests that NP problems are not randomly distributed but follow a fractal structure.

Implication for P vs. NP: - If a deeper structure in NP problems can be identified, this could break the exponential complexity barrier. - The fractal organization could provide an alternative ordering principle for search algorithms, similar to how quantum algorithms already offer advantages today.

The fractal structure of the zeta function could provide a new perspective on NP problems. Superposition in quantum mechanics could serve as a natural mathematical analogy for the distribution of zeta function zeros. The existence of a fractal order in NP problems could open new pathways for efficient computation.

No strict mathematical proof yet: - A formal demonstration is needed to show that NP problems can indeed be described by a fractal structure. - Specific quantum algorithms must be developed to leverage this structure for efficient computations.

Future question: Could the mathematical structure of the zeta function contribute to a new theory of P vs. NP?

2 Re-writing algorithm

We give the subtraction re-writing method to solve this problem in linear-logarithmic time as per our previous research. The technique known as re-writing is summarized, for the &-operator we use the overridden state with logical consumers as well as for the subtraction operator which is defined within the same terms, however, differing only in logical statement, the complement operator in this manner is also re-written using the alphabet star and subtraction from the operand. Thus, the mix of re-writing and logical state composition gives the way to the modified subset construction, which rather than visiting every possible composition, produces exact answer on each iteration, thus giving the polynomial running time, rather than exponential or even factorial. The DFA constructed from the subtraction of DFA1 and DFA2 was constructed experimentally on the example in [5]. The first DFA corresponds to the regular expression $((aaaa)*a|(aaaa)*aa|(aaaa)*aaa)$, as the second one to $(a|aa|aaa|aa(aaa)*|aaa(aaa)*)$, the figure above shows the result produced by Regex+ software package. Thus, the decision problems based upon the extended operators can be solved in full and more efficiently if we will choose the strategy of computing locally optimal solution which gives the optimal step to the global one - this technique we will call as the globally local increment (GLI). This decision problems which we encountered are only two: minimal distinguishing DFA (md-DFA) and non-emptiness DFA of the given automata: both of which has the state space of factorial size, which, in turn, means that we have to choose the better strategy and solution in order to get to the certificate of acceptance in non-polynomial problem within the visible time limit, in our experiments, it didn't exceed more than minute. In the next section we will give the compound benchmarks for the derived examples.

Re-writing works as it was outlined, thus, we can state that the DFA corresponding to the expression $R_1 - R_2$ is a subset of DFA(R_1) and isn't contained in DFA(R_2).

3 Proof by Product Automata

As it was presented in [6], the maximal complexity of product construction is the product of its operands, which can be factorial due to the number of automata, however, we can use same methodology as we have presented before using re-writing and event call, thus, giving polynomial solution to various number of operands with variable cardinality. As it was present in [7–9] the minimal intersection of arbitrary automata cannot be approximated using product construction as it gives the factorial number of solution to be searched, otherwise, the better strategy is to use modified subset construction approach. The term

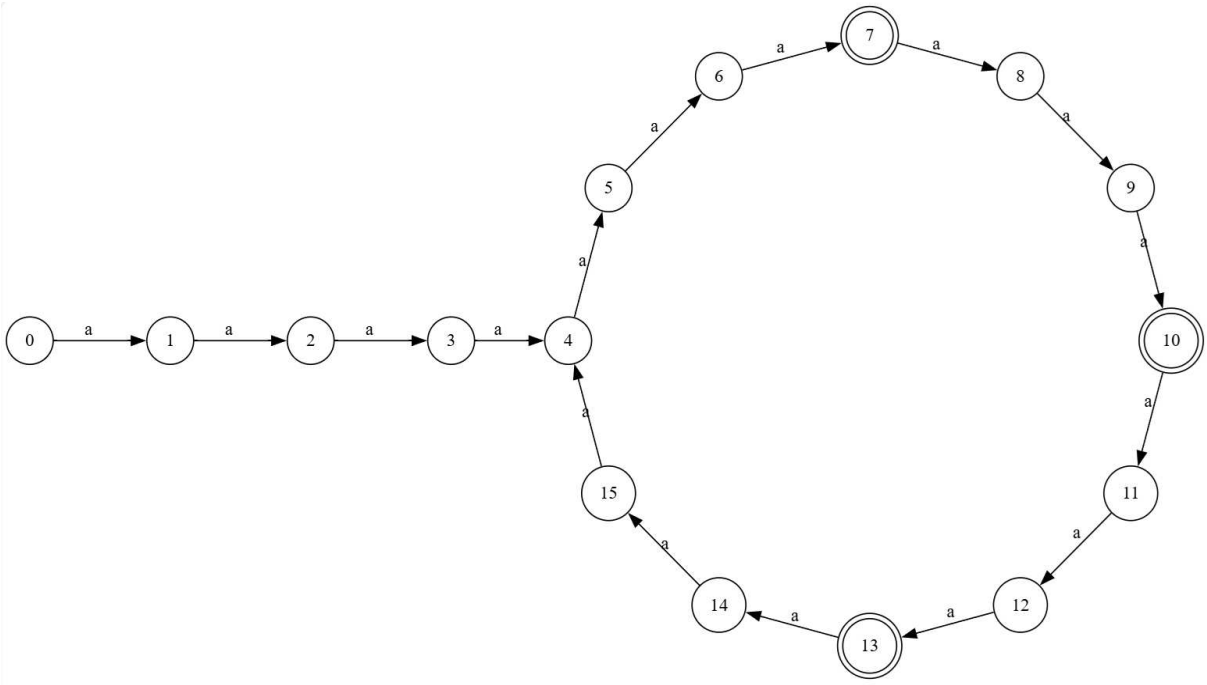

Figure 1: The distinguishing DFA for example in [5]

"minimal distinguishing" can be also viewed from the approach we invented before as we can compute the minimal possible automaton by simply computing the shortest path between starting and ending points using Dijkstra algorithm. The non-emptiness problem which is known to be NP-hard will be also proved to be solvable by a polynomial algorithm in the next section. Another counter-example is from [10], where the expression in the form $(ab) * \&...$ was studied, we have shown in the next section that it can be computed in time of several seconds for the string of length 8000 containing 1000 intersection expressions - this gives the contrary towards the minimal DFA recognizing this language which has an exponential complexity. During the review of the present results we haven't met other counter examples which could get the running experimental program to work with errors or in not observable time frame. Another case we get from [11] for intersection operator, which gives the exponential estimation of the time and space complexity, our results give the reasonable amount of time not greater than 12 seconds for one thousand intersections. The proof of correctness of re-writing algorithm and modified subset construction can be done by viewing the cut, as it was presented much more earlier. We also point the regression of complexity to almost linear and quadratic with respect to

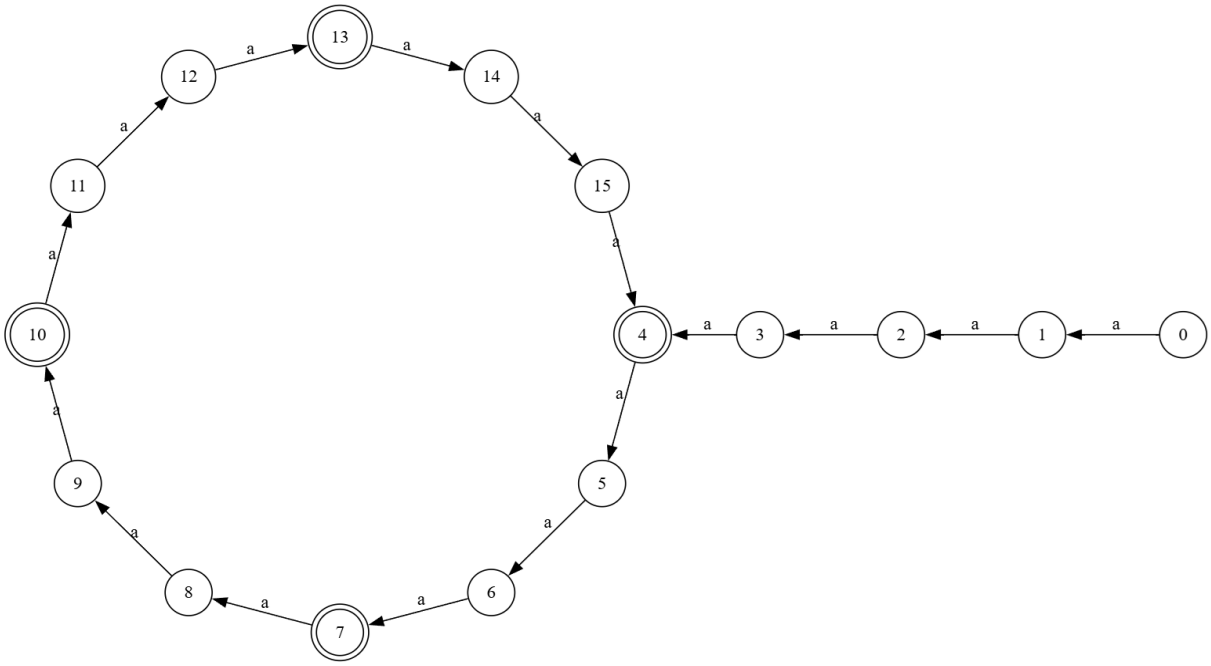

Figure 2: Example md-DFA for the expression $((aaaa) * a1, 4)4000 - (a1, 3(aaa)*)4000$

the back-referencing problem in extended regular expressions, one can see that number of decompositions decreases as we proceed further with the given search and, thus, certificate of acceptance is achieved almost "on the fly". The co-NP complete problem [12] can be by analogy solved in reasonable time as of our experimentation procedure, which states that co-NP classes lie within polynomial P classes.

4 Benchmarks and Experimentation

In table 1 there is a summary of the tests on the expression in the form of $((a^k) * a^{1..k})^k - (a^{1..k}(a^k)*)^k$.

On the next figure there is a visualization of the data in Table 1, as it can be seen results converge to quadratic polynomial function.

With respect to the term fixed-parameter tractability (FPT) as our alphabet before in tests consisted of only letter, we have run tests for arbitrary alphabet "abcd" for cases in form $((a|b|c|d)*)^k(a|b|c|d)^k - ((a|b|c|d)^k((a|b|c|d)*)^k$:

In the other test we will use the regular expression in the form $a^k * \&.., k = 1..n$, the results are as follows.

k	String length	DFA Number of States	Time (sec)
0	63	16	0.155
1	93	25	0.018
2	129	36	0.032
3	171	49	0.042
4	219	64	0.053
5	273	81	0.071
6	333	100	0.085
7	399	121	0.093
8	471	144	0.185
9	549	169	0.261
10	633	196	0.817
11	723	225	0.403
12	819	256	0.389
13	921	289	0.509
20	1803	576	2.808
30	3573	1156	13.06
40	5943	1936	41.488
50	8913	2916	188.674
60	12483	4096	276.88
70	16653	5476	585.398
80	21423	7056	1222.708

Figure 3: The results of expression tests using re-writing algorithm

5 Discussions

5.1 On Effective Algorithms and Cook's Conjecture

This section is a review of the advancement methods in modern combinatorial optimization within some major results in usage of dynamic programming on trees as well as main conjectures in graph theory and theory of computational

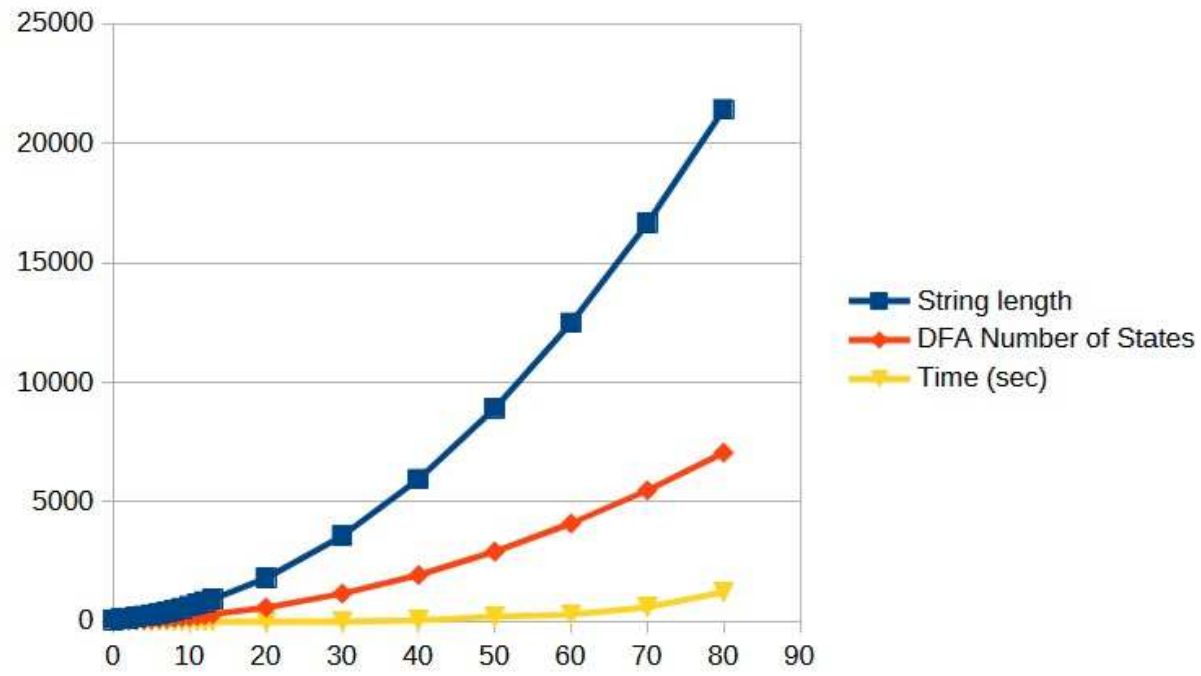

Figure 4: Visualization of the benchmarks in first table

complexity, which in recent time is studied more as we get in time within modern trends like social networks and publicly available hubs, most of which rely on artificial intelligence, however, this work won't deal with AI, better we will propose several fundamental approaches, conjectures and questions as per which we can give a clear and positive answer that this problem isn't an ending case and, thus, can be probed on the particular basis which include the deep review of the newest papers on graph theory and other conforming topics, which are, in turn, become popular during the past decade of the research within tractability, application and generalization progress, we also give the important relation to chromatic numbers in graphs.

In this preamble paper we give the definition of some effective algorithms like subset construction and variable maximum flow problem using potentials which was better studied before as the analogy by Malhotra-Kumar and Maheshwari; we will also go further and show that the Stephen Cook's conjecture of the NP-complete problem implies the uncertain complexity classes which were classified by us before as to be impractical while the certificate of correctness remains of polynomial complexity.

k	String length	DFA Number of States	Time (sec)
0	104	14	0.284
1	158	12	0.08
2	224	13	0.059
3	302	14	0.1
4	392	15	0.132
5	494	16	0.131
6	608	17	0.209
7	734	18	0.307
8	872	19	0.335
9	1022	20	0.338
10	1184	21	0.437
11	1358	22	0.394
12	1544	23	0.423
13	1742	24	0.529
20	3464	31	1.78
30	6944	41	7.442
40	11624	51	17.878

Figure 5: Tests for the quadratic alphabet

As it was proposed and defined before in a seminal paper the NP-complete problem is a problem whose verifying certificate is linear, however, it was incomplete to define the number of possible solutions to form the multiplicative space over the operator (*).

Dana and Scott also remained many unspecified in their decision of the subset construction algorithm which was actually superseded by Berry-Sethi algorithm which produces the linear number of states in deterministic automaton with respect to the preliminary construction algorithm.

Since the definition of the networks and optimal flows on them , number of many algorithms was proposed – one of them is due to Malhotra, Kumar and

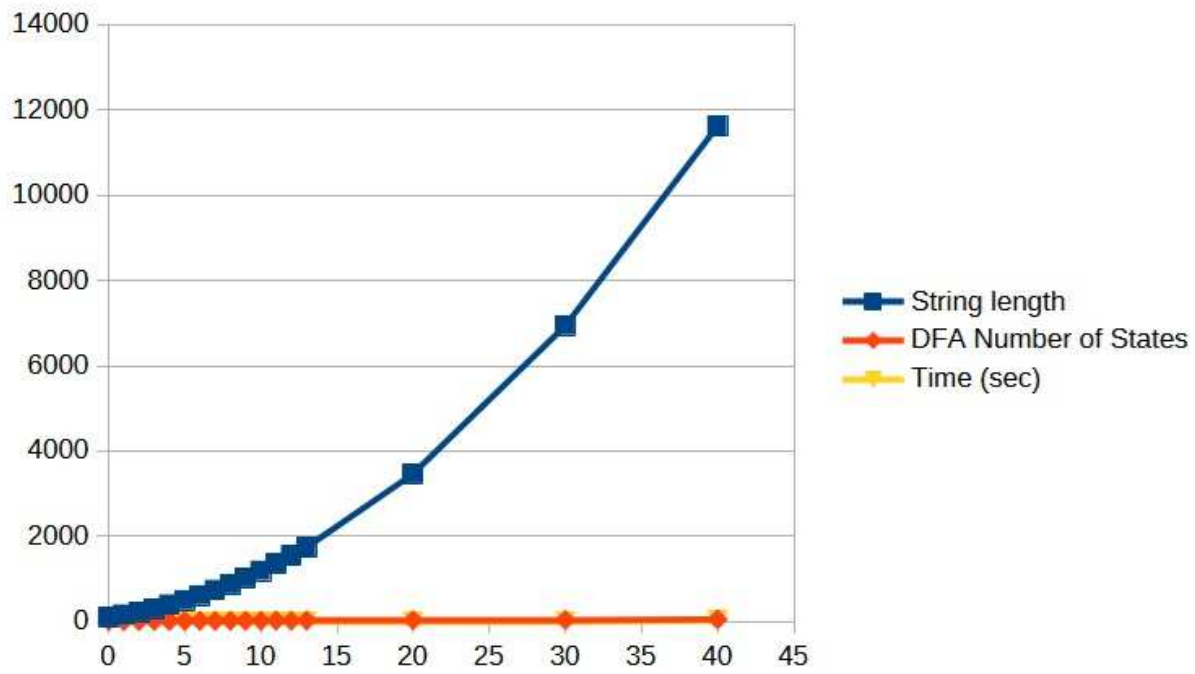

Figure 6: Visualization for example arbitrary alphabet

Maheshwari which has a polynomial cubic complexity. We also give the notion to the super string problem as it's EXPTIME- and EXPSPACE-complete, thus gives the way of defining it as NP-hard.

As the efficient algorithm which includes both the optimization process as well as the conversion of non-deterministic automaton to deterministic one can be viewed as the splice between the initial and accepting states, thus giving the notion to the cases where the exponential blow of number of states occur which we have well studied before and gave the unary O(1)-time complexity check. Thus, this tendency gives the proof of the linear nature of the subset construction algorithm whose minimal upper bound is $O(n * log(n))$, however the minimal one is $O(n)$. In this method we make the choice by the divide-and-conquer strategy from beginning to the end of the state graph describing the automaton – this gives the possibility of avoiding variety of optimization techniques which doesn't pass the dead point of the exponential growth of spaces, however, rather our algorithm makes it possible "on-the-fly" which predominant viewing on the combined techniques of construction and optimization applied together. On the figure below the basic idea is depicted which shows how the

algorithm works on non-deterministic finite automaton while making it deterministic. The super string problem can be viewed from the minimal bound of exponential complexity as there is the minimal string of variable length $n = 2^{f(s)}$ – for each s in the set of all string S as the any minimal string containing all the strings in S as sub-string can be viewed on the other hand as a minimal string along the "trie" for which the dynamic optimization is applied and, thus, the correct composition is sought on the every node of the string forrest. Also Malhotra-Kumar-Maheshwari algorithm is a good sense of creating the variable algorithm with potentials defined for each of the element of the flow network, where the optimization is applied according to the hierarchical order. At least, this is true, for variety of networks and lead to the exponential blow-up for the networks of finding maximal pair whose algorithm uses maximal flow algorithm along the augmentation paths.

We have defined the optimal cases for the subset construction algorithm which was proved to be linear in complexity, also we have shown that the definition of NP-complete problem originally has to be expanded to the class of the uncountable spaces which cannot be realized in time of the arbitrary polynomial function. From the above it follows that DFA has same power as NFA and can be used practically in the testing or membership problem. Also, we have revisited the maximum-flow problem with the definition of arbitrary potentials of each of the vertex which is defined as its minimum of the incoming and outgoing flow. Also the superstring problem is actually is NP-hard as we have shown shortly in this paper due to the variable complexity of the string to encompass any other defined set of strings to be checked against the correct answer.

5.2 Argument towards Cook-Rabin-Scott Conjecture in Complexity Theory

We give the full proof of the equivalence of complexity classes like polynomial (P) and non-polynomial (NP) according to Cook-Rabin-Scott conjecture and our prior results of the subset construction which were first proposed by Berry-Sethi. The "P versus NP" has a long-lasting history of its interpretation and first appearance and definition . As it follows from the original paper the problem can be classified as NP-complete if there's a defined subset of certified words in language DL(M), where M is a Turing tape automaton or non-deterministic finite automaton (NFA) as they are isomorphic due to our prior objective finding. As it was well presented and discussed by Scoot and Rabin in , the NFA can be well converted to deterministic finite automaton (DFA) encoding arbitrary set of accepting words over the language L(M), also: DL(M) is a subset of L(M). Berry-Sethi gave the definition of the linear size automaton and the undefined complexity of the pre-computation stage on the abstract syntax trees of the input regular expression . Also it was shown before that complexity classes have the barriers of their weight along the computational space . It was shown before that linear programming (LP) can be used to solve NP-hard problems with the given customization of constraints .

Cook-Rabin-Scott conjecture can be obtained as a theorem proving the equivalence of P and NP-classes along the full proof of the linear pre-processing and main algorithm complexities when converting the sub-automaton DL(M) to deterministic.

As we have shown before the complexity of converting NFA to DFA is linear in time and space, also any DL(M) can be represented by the regular language and, thus, we have that P equals NP for the subset of the certificate language.

5.3 Homomorphism of Regular Languages is NP-complete

We consider the problem of homomorphism on regular languages by defining the mapping over the set of alphabets on two difference language, we will also show that this problem is actually NP-complete and can be solved by polynomial algorithm.

The Cook's statement of "P versus NP" still remains actual to the present day as many researchers tend to get the argumentative response on the practical meaning of the open problems which can give the open to the new applications of regular languages theory . The partial proof of the existence of the semiring homomorphism on the account of alphabet substitution problem was given in – this problem is to find the mapping between two regular languages for their alphabets, so that they are homomorphic. We solve this NP-complete problem by using maximal bipartite matching algorithm , which can be even parallel .

Obviously the certificate of acceptance of the Homomorphism Problem on Regular Languages (HPRL) stands against the undefined, infinite or arbitrary set of input, moreover the possible measure of matchings is of factorial complexity, thus we proof that HRPL, or HOM, is NP-complete.

We give the set of measures on the bipartite matching graph, where the left side is of one alphabet and right side is of another alphabet: at each time of iteration the matching weight is increased according to the relation difference function between two symbols in both set of symbols. As we know the algorithm complexity in this case is at most cubic for the relatively small set of letters.

5.4 Differentiating between Complexity Notation within Upper and Lower Classes

We present the final outcome on the account of upper and lower bounds for complexity classes like polynomial and non-polynomial, including exponential and factorial growth as per subset sum problem or classical Travelling Salesman Problem, the further distinguished relation can be used further in particular domains of application of complexity theory like Applied Mathematics and pure Mathematical way of expressing relations between Cook's main 3SAT-theorem and its partial cases like functional divergence and other related theorems and foundations.

When the "P versus NP" was first introduced , it was still unclear if there is at most a connection between complexity classes, their big-O notation and asymptotical complexity in mathematics , which states that there exist a limit

between the complexity and its first definition by Cormen et al. Recent research also showed that even linear programming within additional constraints gives the profound solution to TSP and 3-SAT problems. We will show further the full relation between O-notation and asymptotics in terms of pure mathematics for its further application.

Thus, we get the following statement for P and NP complexity classes and their classification function like O(f(n)), where f is a function: $N^+ \to N^+$:

$O(f(x)) = \{f(x), f(x) \in NP; x^k, f(x) \in P \wedge f(x) < x^k\}$

While we have already classified the class of NP-complete functions to be starting from exponentiation and factorial, including Ackermann's function , which is a super potentials over others. Polynomial functions are actually solvable functions and can be computed in the observable amount of time.

As it was presented in the partial function $O(ln(ln(x))$ is polynomial of the order 1, while $o(ln(ln(x)) = 0$, the middle and actual theta-function will be as follows – $theta(ln(ln(x)) = \frac{(O(f(x))+o(x))}{2} = \frac{1}{2}$, thus it's obvious that series S converge to Rhiemann's complex number z.

5.5 Algorithm Deciding Automata Ambiguity Problem

We give the proof of NP-completeness of arbitrary automata ambiguity problem which shows that according to our functional hypothesis, there's a function spawning the polynomial algorithm to solve it, we will also show that it's of affordable complexity.

We start from well-known "P versus NP" theorem , which proves the full invocation impractical content in order to decide any NP-hard problem, we will use further the final version of subset construction algorithm . Also it's known that first definition of derivatives and extended operators was well-studied by Berry-Sethi . Orna Kupferman et al., later gave a cubic algorithm for the decision of extended operators like intersection, subtraction and complement . The problem itself is stated in .

The problem is to decide if the given set of initially non-deterministic finite automata (NFA) are equivalent, as well as their subsets like deterministic ones (DFA). Obviously, the problem lies in recurrent relation which leads to the undefined behavior of acceptance and search of the accepting states defined as certificates – this gives the full proof of the NP-completeness of this problem. In order to solve it we use the Modified Subset Construction (MSC) by using the subtraction operators in extended finite automata (EFA), the algorithm complexity, as it was shown, before is linearly logarithmic, $O(n * log(n))$ – to be exact.

5.6 On Account of Regular Automata Separability Problem

The recent research showed the new problems coinciding with our algorithm for extended operators including intersection, we will use it in order to solve the separability problem as it was stated before.

As we already know there's a set decidable and non-solvable problems . The subset construction algorithm gives the determinisation of non-deterministic finite automata (NFA to DFA), also intersection operator was well studied in along the regular sets or, in other words, sets produced by any regular language. The separability problem was first presented in , the problem is coNP-complete which is by definition the both sides of non-decidable or non-polynomial problems, we will give the polynomial method of computation which gives the answer of the verification against separability of arbitrary number of sets on the automata for extended regular expressions.

As we can conclude the only relation of intersection of the sets K and L in spawns arbitrary language which can be non-regular as regular languages are actually subsets of any language as a set of words by definition, thus we can simply test the intersection operator using Modified Subset Construction (MSC) with state activators in the way it was presented in our seminal paper. The complexity of algorithm lying in P-class is linear and constitutes the number $O(|K|+|L|)$ - this is by the way the lowest bound for any coNP-hard separability problem on regular sets or other non Parikh automata.

5.7 Disproof of Unsatisfiability of Boolean Circuits

We give the full disproof and shade the light towards generalized MAX-SAT problem, also classically known as 3-SAT, which cannot be solved on any Turing automata in observable amount of time even if there is a tie between polynomial and non-polynomial complexities.

In this preamble we follow the certain source of the foundations of Computational Complexity Theory by Stephen Cook , who also showed that 3-SAT problem and its general case MAX-SAT on boolean circuits cannot be handled by Turing tape automata or their isomorphisms like non-deterministic finite automata and deterministic one. For the past time the SAT problem was well applied and studied in-depp , however the main ridiculous challenge is about to build the universal SAT automata – the author of this work shows that there could be boolean function which can make the automata producing positive answer on some set of inputs and their co-variants.

Yes, the problem is still open and innovative as any boolean function on the mirror circuit can produce either positive or negative answer, however, this problem is a case of the generalized MAX-SAT problem which is known to be NP-complete and, thus, unable to be solved in reasonable amount of time using computational materials like present day hardware.

5.8 Full Proof of Universality of Regular Languages

We give the fool proof of the universality of regular languages, which states that any language is regular and every regular language can be arbitrary.

The proof problem of regular languages is actually NP-complete as there is no state automata which could be deterministic and descriptive at the same time – this fact is well-stated for any regular language and for any arbitrary

language . We will show further that these languages are actually equivalent using Aho-Corasick algorithm .

Obviously, Aho-Corasick tries are linearly deterministic and can form a logarithmic regular language – this fact shows that both regular and arbitrary languages are equivalent.

5.9 Star Packing Problem Algorithm in Linear Time

We give the full exact algorithm to star packing problem.

The problem is considered to be NP-hard and obviously NP-complete due to the reduction to classical Vertex Cover Problem which has a parameterized invariant . Star packing graphs were studied before in and the actual statement of the problem is given in .

We build the tree from the graph and optimize it linearly using the leaf traversal strategy which gives full and exact answer.

5.10 On Consensus of Cardinality of Complexity Classes

We give the notion towards Louiz's partial conjecture about inequivalence in classical "P versus NP" theorem and other related research.

We are all well-known about undecidability of 3-SAT problem . The term 'cardinality' in computational complexity and its bordering applied sciences was recently raised upon the necessary level . Dr. Akram Louiz gave all the necessary partial solution towards axiomatization of non-conforming complexity classes like polynomials and non-polynomials . The critics behind the scene is completely wrong and further we will give the full shed of light on the mystery in science and its application.

Yes, indeed the exponential, factorial or even Ackermann's complexities cannot be considered as countable – and this is what gives the strong border of the non-existent classical solutions, and as we know until the present time none of the NP-complete problem was solved.

The author of the 'critical work' is completely wrong as he sees Louiz's conjecture as a first argument towards resolution of "P versus NP" concept and we show that these classes cannot be even comparable – this shows that Jamell Ivor Samuels is completely wrong in his critical work.

5.11 Polynomial Solution for Detour Problem

We give the full polynomial solution for finding the detour in graphs in its general case.

The detour problem is NP-complete , detours were used in many aspects of science on graphs . The detour problem was first initiated in . We give the full solution using dynamic programming . Our solution depends on the length of the detour among any pair vertices as we are using dynamic programming approach with memoization which gives the recurrence relation along the two connected vertices and the visited length. We have given the full general solution to graph

detour problem using dynamic programming which runs in time $O(n*k)$, where n is a number of vertices in graph and k is the maximal length of detour.

5.12 Optimization Techniques on Automata and Graphs

We give the profound notion along algorithmic optimization techniques to manipulate with graph and automata structures.

The automata theory was well defined for the past time , the homomorphism of graphs and these automata and the application of this finding was also presented in the prior works. At the present time publication shows the connection between languages described by automata and their mapping to graph automata.

We have a novel technique based upon the strongly-connected components on arbitrary graphs like oriented and non-oriented in general – this technique shows a strong method of describing any graph by its underlying regular language and its finite non-deterministic or deterministic automaton.

5.13 Order on Trees and Hierarchical Logical Problem

We introduce the optimized algorithm solving optimization problems in linear polynomial time, also we give the notion towards the solution of hierarchical logical problem.

The graph theory remains still actual due to its wide range of applications , the problem of finding strongly connected components gives the solution towards the existence of the logical order relation between these components and hierarchy which can be seen on the depth or breadth first search. We are using Fenwick trees for range min query in order to give the full relation between the hierarchical logical system consisting of operands and comparison operators like less, greater or equal.

The optimization towards trees is computed using the directed edge and its subtree by the preserving dynamical programming and the ordered computation of not more than two values for the general case with star vertexes.

The hierarchy according to which our algebra can give the answer to the query in the form of relation between two any operands in a directed graph can be done using the range min query as of Bender-Farach-Colton method of finding least common ancestor in a graph before vertex labeling operation – we are using the same approach which gives the linear-logarithmic complexity of the solution.

5.14 Isomorphism Problem on Graphs within Regular Language Notation

We give the notion towards the algorithm of indentifying isomorphism on graphs using the finite automata and their regular languages.

The graph isomorphism problem is NP-complete . It has a long-lasting history and application. Regular languages were introduced previously in . We

will give the notion towards the automata describing graphs upon their internal structure with respect to some of the values like degree of vertexes and their adjacency property.

5.15 Permutation Pattern Waves and Polynomial Solution

We give fully polynomial solution to the permutation waves problem.

Since Cook's first statement the problem is considered to be NP-complete due to the existence of certificate of criteria as per integer sets with permutation pattern waves. Permutations were well studied before , the first appearance of the permutation pattern wave problem is given in .

As we can see we can use any notation building the corresponding relation on an oriented graph and performing the valid labeling.

5.16 Configuration Swap Problem on Describing Trees

We give the definition of the exact and sub-optimal algorithm to the configuration swap problem on graphs.

The graph theory is a theory which has a long-lasting history and its application . The swapping problem was introduced in and is of very practical meaning.

We simply build the tree from the graph after which we apply the oriented edge optimization as per the given circumstance where all other sets are settled – this gives exact and optimal solution to the minimal swapping problem.

5.17 On Boolean Circuits and Optimal Prefix Codes

We give the full notion on the boolean circuits and their relation to finite automata as well as the definition of the optimal prefix codes for the binary encoding of the words in text.

The boolean circuits were defined in , with its prior statement on the solution of the defined function – as it can be seen they can be converted to the deterministic finite automata defining the language on which the boolean function will be satisfied or, in other words, equal to "true". Efficient encoding and prefix codes is a far more historical problem .

The boolean circuits can form a typical non-deterministic model which can be determined, thus giving the observation towards the solution on a random function and random configuration.

Optimal encoding prefix codes are to be formed from the assumption of the division of the sums of occurrences of the symbol in source text, thus together forming the combinatorial optimization problem, where the division strategy is due to pivot selection and obeys the certain subset sum problem.

5.18 Non-polynomial Complexity of Permutation Automata

We give the full coverage of the notion of the permutation automata deciding complexity to be NP-complete.

The definition of non-solvable problem was first introduced in , permutation automata in general were presented in , the problem of permutation automata acceptance without weighted function was proposed in .

As we have already shown that there is a local bound for permutation automata which can be re-presented in regular languages with extensions like back-references, it's obvious that full optimization network complies with the classical NP-complete problem as Traveling Salesman Problem (TSP). The above fact shows that there could be non-countable number of pre-permutations before visiting the layer on the arbitrary state of automaton.

5.19 Token Sliding Problem on Graphs in Polynomial Time

The quadro-linear polynomial algorithm is given for the token sliding problem on graphs.

Graph theory has a long history and meaning as a model , the token sliding problem is well-known also , we will show further that this problem can be optimized on a produced tree from an arbitrary graph.

At each step of optimization we change the order of independent set series according to the orientation of the leaf and its sub-tree due to this orientation – this gives a quadratic worst case method to compute the number of swaps in order to get the right configuration of independent sets.

5.20 Solving Optimization Problems on Graphs using Automata Composition

We give the full definition of the optimization problem and its isomorphic transformation to the non-deterministic finite automata as per the order of traversal and its corresponding settlement as in undirected as well as in directed graphs.

The optimization problems on graphs are known to be NP-complete, since the optimization function can be easily verified and hard to get to the optimal point . Graph theory is a model on which even finite automata can be operated in a pre-defined method. The strongly connected components of directed graphs give the notion towards the ordered relation between each of the node which can be optimized as per the sample problem like .

We can construct the correct automaton recognizing the language of the paths in the graph after which we apply the optimization according to the order in the strongly connected components of the directed graph, for general case of undirected graphs we can consider the same option with respect to the search strategy.

5.21 Polynomial Algorithm for Clique Problem using Matrix Space

We give a polynomial algorithm in quadratic complexity for finding cliques in graphs according to the matrix space with single operation like boolean multiplication of the adjacency matrix of the given set of vertexes, we will show that this is a fully polynomial solution with the current lower bound on the number of operations in order to find the clique in graph of the defined rank.

The hardness of the problem is a key of its classification , graph theory is described in , the clique problem is known to be NP-complete and, thus, is to be solved efficiently using conceptual algorithmic approach.

We give the matrix of adjacency an algebra with single closed operation like boolean multiplication, after forming the maximal independent set and applying the multiplication of this matrix and its transposition we can devise the sets for which this can be implied according to the matching within the kernel of the clique – this operation reduces the size of sought input up to the given order.

5.22 Subgraph Enumeration Problem on Graphs

The polynomial algorithm is presented along which the number of subgraphs of the given graph can be counted.

The NP-hardness of this problem is defined as there are many subgraphs and only isomorphic certify for the given subgraph in order to count all its isomorphisms in the given graph. Graph theory is well-defined during the past time , subgraphs and their isomorphisms are defined in . The counting problem is presented in .

We give the solution towards finding the number of subgraphs or simply enumerating them during the descent on a produced tree for the given graph and subgraph, thus, we can solve the problem by applying recurrent relation on the edge which divides the tree in several parts with respect to the structure of the subgraph.

5.23 Solution to Triangle Finding Problem in Graph

The fully quadratic algorithm in maximal number of edges is presented in order to find the number or enumerate all subgraph triangles in graph.

Graph theory is presented in , the triangles are discussed in , the problem of finding triangles in graphs is in .

We use at maximum quadratic space and time and number of edges at most which is the most optimal exact solution to the stated problem. At first we build the sub-tree of a graph and the adjacency of any two pairs for the pre-computed set of vertexes where the third vertex is a middle and, thus, has the adjacent two vertices in the edge of the tree.

5.24 Solution to Disjoint Paths Problem on Graphs

The linear algorithm in the number of edges and vertices in graph is given for finding k-disjoint paths.

Graph theory has a long-lasting history and application . The path or vertex disjoint set problem in a graph is given in .

We start from the set of each pairs from the left to right and from right to left by building the fully directed tree ascending in both directions so that there would be a cut of size more than k, thus, satisfying the condition of disjoint path on vertexes or edges.

5.25 Solution to Maximum Satisfiability Problem and Minimal Vertex Covers on Graphs

We give the solution to partial maximum satisfiability problem on the example of enumerating minimal vertex covers which coincide and are non-polynomial, our solution is fully polynomial and exact.

The problem of not more three variables in logical satisfiability problem was proved to be NP-complete as well as its case on graphs for finding the minimal vertex cover .

We build the tree in which we descend from the produced tree and use memory in order to store the bitmap of all satisfied conditions as well as per model of minimal vertex cover on graphs.

5.26 Solution to Even-Path Problem in Arbitrary Graphs

We give a polynomial solution to even-path problem on graphs between two given vertices in arbitrary graph as it can be either directed or undirected, our approach also states the minimal bound of number of edges and vertexes in graph.

The problem can be seen as NP-complete , graph theory was well described in , the even-path problem is defined in .

5.27 Solution to Minimal Decomposition Problem on Graph

We give an algorithm and minimal bound for linear decomposition of the graph with given maximal degree of its vertex.

Graph theory is described in , the linear arboricity conjecture is stated in , according to which there is not more than half of the maximum degree of the paths.

6 Recurrent Diversification of Counting Alternation Permutations

We give the recurrence relation towards the counting of alternation permutations thus providing the exact formula in order to compute the number of alternating iterations within the insertion operation and the union of the sets.

The permutations are well presented in , alternating permutations are permutations with pre-defined order .

Since implication we give the upper bound using the recurrent relation which is defined as the oracle function $PA(n)$: $PA(n) = f(n) * PA(n-1)$.

Where $f(n)$ is a function defined as the error factor for which the alternation decision holds true, obviously $f(0) = 1$ and $PA(0) = 1$.

To define the function $f(n)$, we are using each triplet consideration with respect to each triplet in the form: $a_i < a_{i+1} > a_{i+2}, a_i > a_{i+1} < a_{i+2}$,

The above definition is a result of the term alternating permutation in its canonical sense. As we see from above the second condition cannot hold true as we cannot insert the biggest element n in any of the position when this fact is satisfied. Let's consider this occurrence: $a_i < n > a_{i+1} > a_{i+2} < a_{i+3} : a_i < n > a_{i+2} < max(a_{i+1}, a_{i+3}) > min(a_{i+1}, a_{i+3})$.

For the second condition we have: $n > a_{i+1} < max(a_i, a_{i+2}) > min(a_i, a_{i+2}), n > min(a_i, a_{i+2}) < max(a_i, a_{i+2}) < a_{i+1}, min(a_i, a_{i+2}) < max(a_i, a_{i+2}) > a_{i+1} < n$.

Thus, we have four subsets to devise the function $f(n)$, thus giving us the following exact relation like: $f(n) = 4 * PA(n-1) : A_1 \cup A_2 \cup A_3 \cup A_4$.

Obviously: $4_n * P_{A_1 \cup A_2 \cup A_3 \cup A_4} \leq f(n) \leq 4_n$.

Where in above relation probability is the union of all the cases when the four insertion conditions hold true, which is recursive and can be counted.

6.1 Reductions of Graph Edge Coloring Problem and Chromatic Number

In this short note we are to give the note towards graph edge coloring problem (ECP) and its reduction to graph vertex coloring problem (VCP), which gives the significant result in deciding the minimal number of colors for edge coloring problem.

The graphs are widely studied in , chromatic numbers in VCP denote the minimal number of colors required to color it so that no two adjacent vertices bear the same color. The latest research aims also towards Euler's lattices problem.

As we can construct the graph for the given graph $G(V, E) : G(E, (a, b), (a, i) \& (b, i) \in E \forall i)$, it follows that the chromatic number can encode the number of edge coloring in ECP, so that this number is at least greater than the same number of the initial graph by induction.

6.2 Relation between Chromatic Number and Length of Hamiltonian Paths in Graph

We give the strict computational relation between chromatic number of graph and sum of lengths of Hamiltonian paths using set exclusion theorem as well as the addition towards inverse graph.

Graph theory was steadily studied in , the graph coloring problem and its chromatic number are known to be NP-complete , the partial relation between these numbers and the length of the maximal path were studied in .

As per the set theory, graph can be considered as a set if we would at each step of iteration remove some 2-vertex graph with a single edge or not, this will look like as follows:

$|G(V,E)| = |G(V_1,E_1)| + |G(V_2,E_2)| - |G(V_1 \cap V_2, E_1 \cap E_2)|$.

The above relation can be approximated within any path if we would get at each iteration the pair of nodes (u,v), then our relation will look like:

$|G(V,E)| = |G(V-v, E-v)| + 1, 2 - 1$.

As in both division operator we divide the parts along the maximal length and an optional edge in graph, obviously this function is to be minimal, thus we have to find a path of maximal length in the inverse graph $G(V,E)$.

Thus, we get to the following relation:

$\chi(G(V,E)) = |V| - max\{\sum_{p \in H(\ G)} |p|\}$

Where $H(\ G(V,E))$ is a set of longest paths through the whole set of vertexes in inverse graph, the paths are to be disjoint.

The proof can be done by induction to the general graph G(V, E) as we approximate towards minimal possible number. This proof gives an evidence of the connection between Dirac's formula for graph containing Hamiltonian and the chromatic number of the inverse graph.

We have given the strict relation between longest paths which can be either Hamiltonian of size —V— - 1 or any other maximal possible of all the paths in inverse graph, thus, giving observation of the Hamiltonian cycle presence for Dirac's formula on general graphs. This fact gives us the observation of using the divide algorithm on inverse graph in order to find the maximal longest path of the maximal size within Dirac's equivalence relation. We will use our equivalence to establish connection between chromatic numbers of the graph and its inverse, thus we have:

$\chi(\ G(V,E)) = |V| - max\{\sum_{p \in H(G)} |p|\}$

From this point, we get:

$\chi(G(V,E)) - \chi(\ G(V,E)) = max\{\sum_{p \in H(G)} |p| - max\{\sum_{p \in H(\ G)} |p|$

In addition we give the definition of the complete graphs or cliques Kn: the paths are actually Hamiltonians in these decomposition.

6.3 Optimal Labeling Algorithm for Vertex Coloring Problem in Graph

We present the labeling algorithm for Vertex Coloring Problem (VCP) which runs in product linear time on number of vertexes and edges in graph at mini-

mum with the chromatic number as parameter.

VCP refers to graph theory , it's known to be NP-complete and, thus, optimal or approximate algorithm is to be applied .

We start from forming the system of inequalities between each of adjacent vertexes in graph $G(V, E)$. We start by labeling with choosing the minimal label index from adjacent vertexes with stabilization principle on the obtained index which can be reverted in time $O(m)$ using each iteration on maximal chromatic number with total complexity of $O(nm)$.

We claim that this algorithm is most optimal as to the consensus of simplex system formed by inequalities and the target function to be minimal possible. The stabilization, thus, runs, each time the node changes its correct labeling according to the selection rule.

6.4 Application and Theory of Several Aspects in Optimization

The 3SUM problem was viewed from a singleton point of view for the past time. In this work the experimental results along with proof are presented: the state explosion doesn't occur in specific cases after decomposition of regular expression into non-deterministic finite automata (NFA), thus, the P-complete procedure to take turn for converting NFA into deterministic finite automaton (DFA) with construction according to the De Morgan Law. We give the notion of the equivalence of the complexity classes due to the recent research according to Rabin-Scott subset construction. We also give the linear algorithm for lookahead and lookbehind assertions in regular expressions by implementing the intersection operator which was well studied before in our prior research, the work also includes: the experimental part of research in our investigation of the "P versus NP" theorem and the optimization principle within the physical layout, the full proof of inequivalence of P and NP complexity classes which can be addressed to the famous "P versus NP" theorem by Stephen Cook, our approach summarizes all the results obtained before in our prior research of this topic and its failure during the decades of its first appearance in the scientific press, definition of the single operation for giving the output to the new state in Berry-Sethi approach of building deterministic finite automata (DFA), address the output, produced by the Turing tape automaton, or Turing Machine, which is further divided as deterministic and non-deterministic, to the set of regular languages recognized by finite automata. It's known that the subset sum problem lies in the NP-class of complexity, however, due to the integer factorization of any number it states another argument towards P = NP. The unified system based upon Ford-Fulkerson maximum flow method for solving the civil engineering problems like flooding and human evacuation during earthquakes or other disasters is also presented. According to the present time the normal forms are consequent to the efficient data manipulation, still there's no universal method for solving this problem according to the criteria of data to be small and the modeled solution provides the sufficient normalization of the source data. The $X + Y$ sum problem as observed wasn't solved before, so we provide a fast and

simple solution to this problem using algebraic properties of the vector as well as the general case. The recent research and study in the theory of Computational Complexity gives new perspectives in studying the "P versus NP" question.

We build the binary tree for each of the elements in binary notation in the given input array, after which we concord the search according to the valid combinations. The conversion of NFA to DFA, or subset construction, and its possibility proof first appeared in has an exponential complexity of $O(2^n)$ and thus is EXP or NP-complete.

Many techniques were done before in order to avoid the effect of state explosion , however, we present the De Morgan law for rewriting both union and intersection operators as well as in extended regular expressions, which leads to P-complete result.

For the past decade the "P versus NP" problem was well studied with conformance that P is a subset of NP. If this happens, then there's a set of problems which are strictly in NP and not in P assuming P not equal NP .

We simply close the circuit in our algorithm when converting non-deterministic finite automata (NFA) to deterministic finite automata (DFA). The same is true even for NFA constructions which give rise to the question of relation between regular language algebra and features like lookahead and lookbehind assertions.

As it was stated before the "P versus NP" question has a long-standing history in the theory of Computational Complexity and Mathematics as well, where the symbol of infinity isn't defined as operand due to the inconvenience of its relation to the operands in the mathematical expression in the algebra of numbers .

Thus, by showing that P equals NP we still cannot devise the relation in this algebra, however, if P not equals NP, we can proceed further with the modern aspects.

Before we have shown that there's a functional relation between complexity classes, i.e. there could exist the function $f(x)$, so that $f(P) = NP$.

We will proceed to the above publication further and give the strict proof of inequivalence of complexity classes and as it follows from this proof the hierarchy of classes which give consolidated proof of the relations between the variety of complexity classes in Computer Science and Theory of Computation .

Before our research the Theory of Complexity was well underlined and it follows that first we have to postulate and only then give the question of the relation between complexity classes, basically polynomial "P" and non-polynomial "NP" classes.

The main problem in the Berry-Sethi approach is the conformance of the new state to the states added before during each iteration of the algorithm. We use the single test application of our method for the equivalence of the states to the regular language they represent.

Since each of the Turing machines has a limited set of states during which it can transit to the next step, or iteration, of processing the input and, thus, going to the halting or accepting, or rejecting, state, we are to define the set of words which are written are well-defined as programs produced by this machine. The regular language is formed from the Deterministic Turing Machine (DTM) or

Non-deterministic Turing Machine (NDTM) can arbitrarily produce the regular set of languages, known as programs of this machines according to the finite state of states in the transition diagram as it can be seen in various sources . On the account of "P versus NP" theorem we are to define the proved equality of P and NP-classes of complexity as Finite Automata (FA) are isomorphic with to the regular language they accept .

The "P versus NP" problem is the main problem stated before.

We simply factorize numbers within the prime number factorization algorithm and build on-level tree structure for finding the structure of the method. As we know the prime factorization is reminiscent of the tractable logic of computer numbers which tend to limit the Ackermann numbers . The prime factorization itself isn't studied nowadays and is well-known to be P-complete within the subquadratic algorithm which is still inefficient against big numbers which are met in cryptography . Still it's omitted that the subset sum problem can be devised from the whole set of problems within the multi-cubic trees along prime factorization and prime number notation system. Still it's posed that linear complexity of introduced parameters harms the overall magnitude of complexity, which is well-known and can be factored according to the optimal notation of consecutive prime numbers to which the parameter tends to grow linearly with predefined maximal speed of Ackermann numbers. We build the growing structure of the tree on each level having the prime number in prime notation of the parameter whose limit is to be deduced from even factorial decomposition which leads to blow and speed-up and, thus, makes the free parameter less playing the role.

The normalization problem is to be presented as the mathematical programming problem within the constraints and the main function as the size of data to be minimized. Earlier we have shown that the factorial number of possible data in each table depends on the number of columns and number of rows. This fact makes it possible to seed the data and store them in a fast and efficient way.

The "$X+Y$" problem is in general still unsolved and plays a role in effective optimization .

We simply sort the items by the normal vector distance to the line $X + Y$ going through the given point.

This simply gives the minimal possible running time on average as $O(n * log(n))$.

As it was pointed out by the modern research the "P versus NP" question is to be studied from a different point of view and the broad horizons of its decidability are to be omitted due to the insufficient approach in the formalization of this theorem.

6.5 Proof

We prove the above fact by the assumption already given in the statement before: thus, as we know any non-deterministic finite automata (NFA) can be converted to analogous deterministic finite automata (DFA).

According to our latest research the subset construction is P-complete and, thus, there exist no automata with the strict property given above, thus, it follows that $P = NP$.

We have devised the parallel computing law as follows: $lim_{N\to\infty}\frac{NP}{N} = P$.

Let's assume that the above law is correct according to the number of threads N, which operate on a Non-deterministic Turing Machine (NDTM). Also let's assume that $P = NP$, then it follows:

$lim_{N\to\infty}\frac{1}{N} = 1 \Rightarrow 0 = 1 \Rightarrow P \neq NP$.

The output above demonstrates the simplest way of proving the inequivalence of complexity classes according to the parallel computing law.

6.6 Experiment

We build the integral circuit on a board and give the experimental projection of the abstract processes and processes which are put in a different environment, or physically. The VLSI devices are used today in many aspects. Hamiltonian is met in the Traveling Salesman Problem (TSP) where it defines the shortest possible path visiting each of the city once. TSP by itself was an argument of "P versus NP" theory and practice . On the experimental electrical board we build the layered circuit by using elements for satisfying the "visit-once" condition. Thus, the shortest path of a limited number of mediate elements can be found.

We develop the maximum flow network on map using any applicable source and present each cell with the incoming or outgoing edges as of the each of the bordering cells on the map.

The maximum flow applied to the above-described model gives the result of simple prediction scheme according to which the residue flow can be pushed forward as well as backward, and the possible dangerous zones with blocking flow can be detected and successfully mapped to the physical map where the ecological disaster happens.

At each model we give the maximum capacity as the maximum capacity of the fluid or human factor.

7 Conclusion

We have given a fully polynomial algorithm for the md-DFA problem which is NP-complete - this fact gives the experimental proof of the equivalence of complexity classes like polynomial "P" and non-polynomial "NP" as the benchmarks above state as the argument as they are almost linear to the size of the expressions and running time depends also linearly, also, the problem described was proved to be NP-complete before. We have also shown the experimental proof of tractability of the problems like md-DFA and Non-emptiness-DFA which are known to lie in NP-class of complexity. We have also concluded that within the new proof, back-referencing problem can be computed fast within arbitrary number of capturing groups. Thus, we claim experimental proof of "P versus NP" theorem: $P = NP$, which could be used in solving other problems like

Rhiemann hypothesis by Akram Louiz . The reader is invited to use Regex+ software package and provide examples. I also have some notes about theories like fixed-parameter tractability and classification - all these theories and similar to them are all about the direct solution or final resolution of P-NP theorem by Cook, while our conjecture of functional hypothesis gives the final outcome with to the full statement of the problem: are there solution to NP-complete problem or not.

There could be parallels between the P vs. NP problem and quantum mechanics, particularly in relation to the concept of superposition. In quantum mechanics, a system can exist in multiple states simultaneously until a measurement is made. Similarly, NP problems could be seen as a kind of "superposition" of many possible solutions that exist at the same time until verification or computation collapses them into a final solution.

Thus, we have also proved that subset construction, or powerset construction, is polynomial, or P-complete, with respect to the prior obtained results.

The common misinterpretation of the "P versus NP" theorem lies between the fact that it can be solved effectively, still, it doesn't follow that from this consequence we can devise the relationship between two classes.

The potential of the method above gives the main result of the past research for efficient implementation of lookaround assertions. We have presented the experimentation theory for which there could be conjectured that each of the shortest paths found on the simulated integral circuit can form the Hamiltonian by itself in its decomposed state.

We have shown the inequivalence of complexity classes around our final proof which is given as a final contribution to the field of Computational Complexity.

The above method is expensive, however, in static mode it can be more productive.

We have come to the end of the "P versus NP" continued story and the positive output of the proof of equivalence of these complexity classes gives the horizons of the universality of automata and their isomorphic properties as well.

The overall can be considered as the other argument towards "P versus NP" theorem and the proof P = NP, as the subset sum is both in P and NP and NP-complete classes of complexity.

We presented the safe method for detecting possible bottlenecks during flooding and evacuation which can lead to a more humanistic approach in science and engineering.

As the much earlier works were towards the static structure of the database, for now we have defined the universal approach towards data normalization.

The above case can be extended to the problem where the normal vector of a line is given with arbitrary weights which open new horizons to the studying of the application and theoretical acquisition of this problem.

The sorting problem for dual coords can be solved in minimal possible time $O(n)$ by converting it to co-NP problem as integer sorting.

Thus, we came through the inequality of the question "P versus NP" which clearly gives the argument towards the preliminary axiomatization of the complexity classes which we name as "decidable" and "undecidable".

8 Acknowledgements

The authors of this work express gratitude to Dr. Akram Louiz for his steps towards "P versus NP" theorem.

References

[1] Cook S. "The P versus NP problem" , *Clay Mathematics Institute* (2000): 2(6), 3.

[2] Rabin M. O., Scott D. "Finite automata and their decision problems" , *IBM journal of research and development* (1959): 3(2), 114-125.

[3] Kupferman O., Zuhovitzky S. "An improved algorithm for the membership problem for extended regular expressions" , *In International Symposium on Mathematical Foundations of Computer Science* (2002, August): pp. 446-458, Berlin, Heidelberg: Springer Berlin Heidelberg.

[4] Sen K., Rosu G. "Generating optimal monitors for extended regular expressions" , *Electronic Notes in Theoretical Computer Science* (2003): 89(2), 226-245.

[5] Martens J. "Deciding minimal distinguishing DFAs is NP-complete" , *arXiv preprint* (2023): arXiv:2306.03533.

[6] Hsieh S. C. "Product construction of finite-state machines" , *In Proc. of the World Congress on Engineering and Computer Science* (2010): pp. 141-143.

[7] Fernau H., Hoffmann S., Wehar M. "Finite automata intersection non-emptiness: Parameterized complexity revisited" , *arXiv preprint* (2021): arXiv:2108.05244.

[8] de Oliveira Oliveira M., Wehar M. "Intersection non-emptiness and hardness within polynomial time" , *In International Conference on Developments in Language Theory* (2018, August): pp. 282-290, Cham: Springer International Publishing.

[9] Pitt L., Warmuth M. K. "The minimum consistent DFA problem cannot be approximated within and polynomial" , *In Proceedings of the twenty-first annual ACM symposium on Theory of computing* (1989, February): pp. 421-432.

[10] Gelade W. "Succinctness of regular expressions with interleaving, intersection and counting" , *Theoretical Computer Science* (2010): 411(31-33), 2987-2998.

[11] Bastos R., Broda S., Machiavelo A., Moreira N., Reis R. "On the state complexity of partial derivative automata for regular expressions with intersection" , *In International Conference on Descriptional Complexity of Formal Systems* (2016, June): pp. 45-59, Cham: Springer International Publishing.

[12] Collins E. R., Kocher C., Zetzsche G. "The complexity of separability for semilinear sets and Parikh automata" , *arXiv preprint* (2024): arXiv:2410.00548.

[13] Louiz A. "The proof of a series as an upper bound for the number of positive square-free numbers and thus for the Mertens function" , (2025)

[14] Kardeis Y. L. "The Universal Fractal Chirality of Riemann Zeta-Function: On the Symmetric Distribution of Prime Numbers, the Uniform Distribution of Nontrivial Zeros along the Critical Line, their Physical and Chemical Correlations" , (2024): DOI - 10.13140/RG.2.2.34775.69285.

String length	DFA Number of States	Time (sec)
6	2	0.135
13	2	0.008
14	3	0.005
21	3	0.009
22	3	0.004
23	7	0.007
30	7	0.008
31	7	0.006
32	7	0.009
33	13	0.006
40	13	0.011
41	13	0.014
42	13	0.013
55	61	0.016
77	421	0.197
93	841	1.003
109	2521	6.094
125	2521	5.678

Figure 7: Results for non-emptiness test

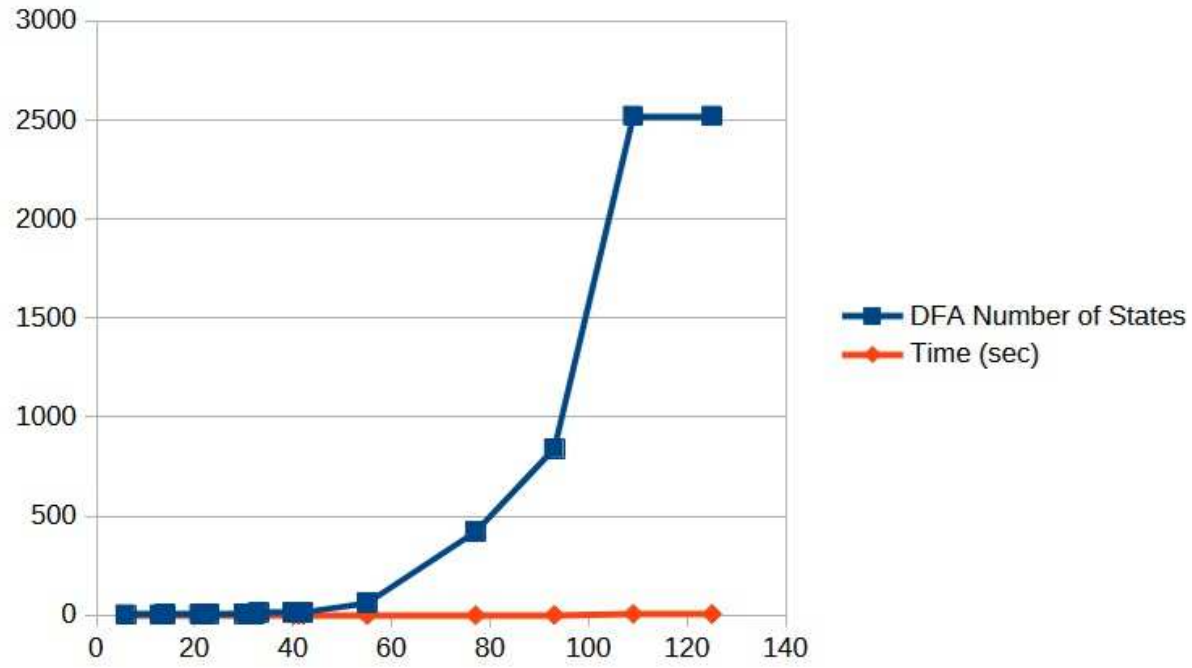

Figure 8: The visualization of the performance of the non-emptiness test algorithm

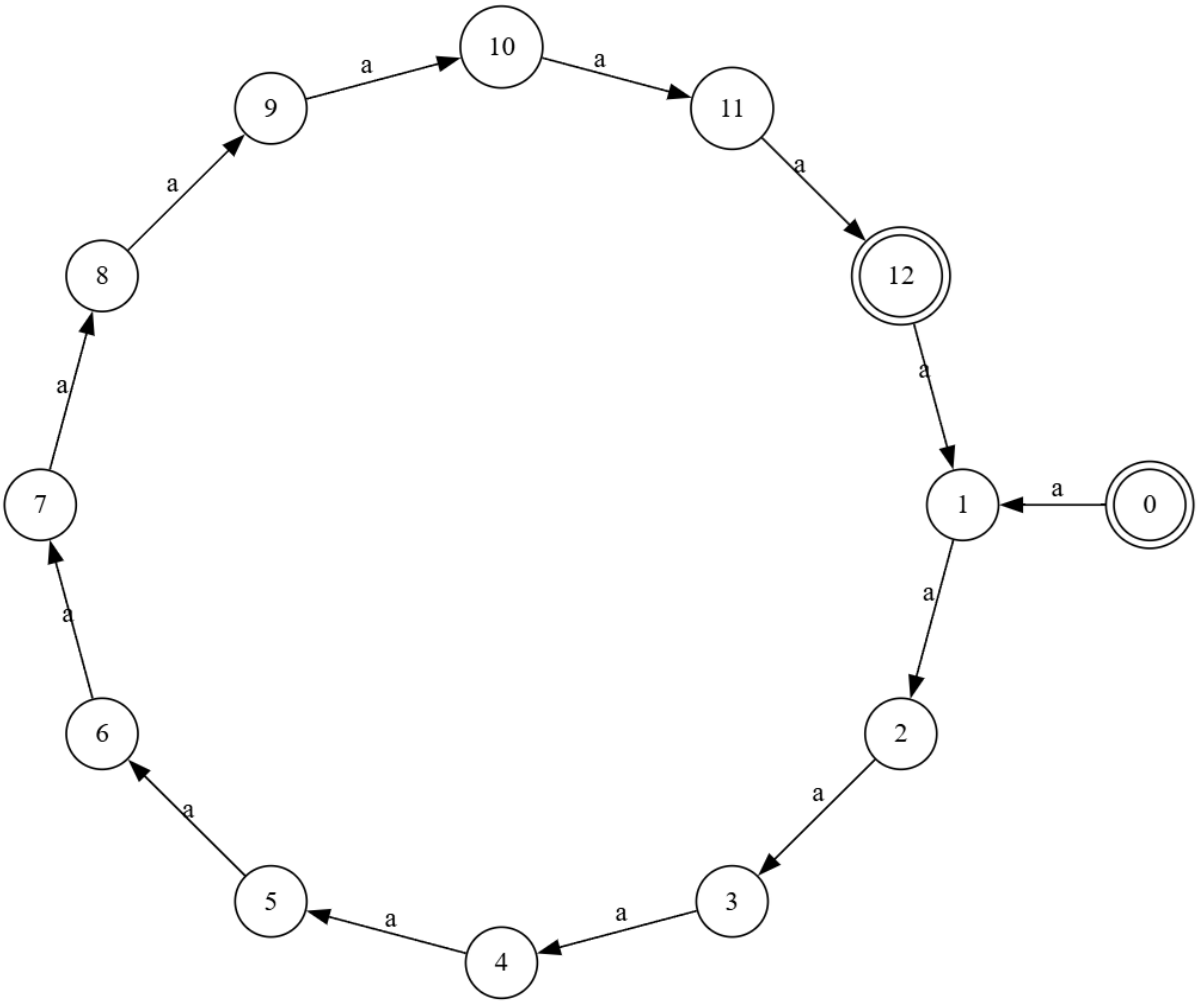

Figure 9: The non-emptiness DFA for the expression $((a)*)\&((aa)*)\&((aaa)*)\&((aaaa)*)\&((aaa)*)$

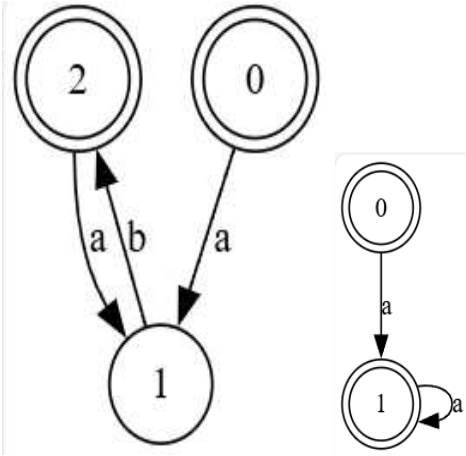

Figure 10: The resulting DFAs for expressions $((ab)*)\&...and(a*a|)\&...$

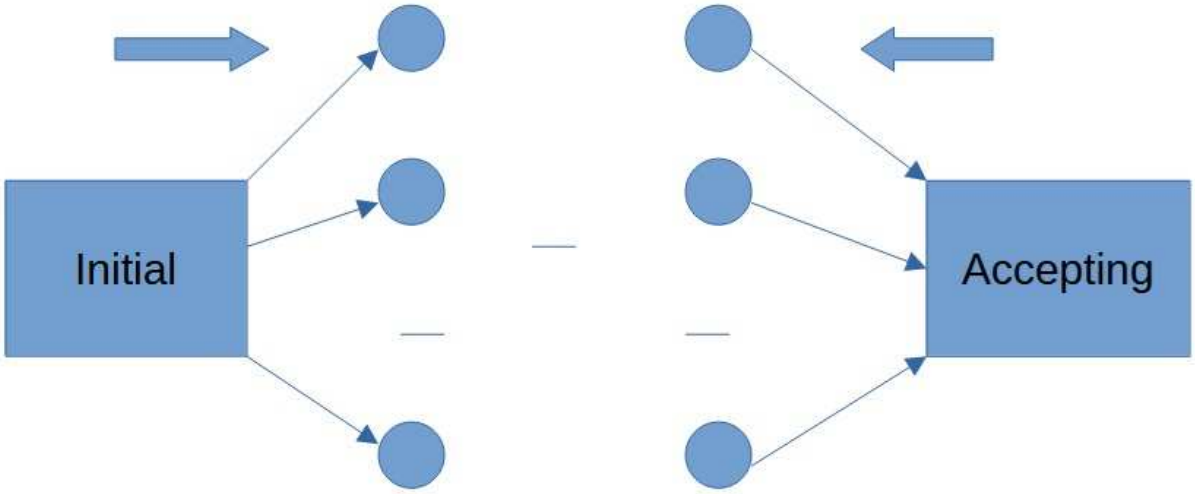

Figure 11: The viewing and strategy of optimal subset construction algorithm