The Prospects for Sub-Exponential Time

Thomas E. O'Neil
Computer Science Department
University of North Dakota
Grand Forks, ND 58202-9015
oneil@cs.und.edu

Abstract

This paper provides a survey of some research literature on the question of whether some NP-complete problems are easier than others. The classic open problem regarding NP-completeness is usually framed as determining whether P = NP, that is, whether the class of problems that can be deterministically decided in polynomial time is the same as the class of problems that are non-deterministically decided in polynomial time. It is widely believed that P ≠ NP, but proof of this result remains elusive. In the decades that have passed waiting for the major open question to be resolved, some researchers have investigated the question of whether some of the NP-complete problems might be easier than others. We know that NP is a subclass of EXP – the class of problems deterministically decided in exponential time, and we expect that none of the hardest problems in NP (the NP-complete problems) are in P. Is it possible that some or all of the NP-complete problems might be in a deterministic complexity class SUBEXP, which lies between P and EXP? If we assume that the Satisfiability problem is strongly exponential, can we prove that all or some of the other NP-complete problems are also strongly exponential?

A survey of the research literature shows that there is no firm consensus on which of the hard problems are easier than the others. In this paper we examine various definitions of sub-exponential time, discuss the hypothesis that Satisfiability is strongly exponential, and review the results regarding which hard problems might be in SUBEXP. Our goal is to provide a clear summary, for both faculty and students, of research on a major open question at the center of computer science that remains intriguing and somewhat controversial, even fifty years after Stephen Cook first defined the class of NP-complete problems.

1 Introduction

We understand the class NP to be the class of languages whose membership functions can be computed by non-deterministic Turing machine programs in polynomial time. P, likewise, represents the class of languages whose membership functions can be computed by deterministic Turing machine programs in polynomial time. We know that every language in P must also be in NP, since a deterministic Turing machine can easily be modified to make it non-deterministic. We don't know, however, whether every language in NP is also in P. The time required for a deterministic Turing machine to simulate a non-deterministic one is exponential in the worst case, so NP must be a subclass of EXP (the class of languages with deterministic, exponential-time Turing machine deciders). But to date, no one has actually proven that some language in NP does not have a deterministic, polynomial-time decider. At the same time, it seems highly unlikely that every NP language has such a decider.

Stephen Cook was the first to identify a problem in NP that is complete [1]. He showed that any polynomial-length computation by a non-deterministic Turing machine can be modeled by a polynomial-size Boolean expression such that the expression is satisfiable if and only if the computation accepts its input. Thus if the satisfiability of a Boolean expression can be deterministically determined in polynomial time, any non-deterministic Turing machine computation and be deterministically simulated in polynomial time. Shortly after Cook defined the first NP-complete problem, Karp demonstrated that numerous other problems in NP are complete by defining polynomial-time reductions from the known NP-complete problems to the new ones [4]. The NP-complete class was soon recognized to be a rather large subclass of NP, containing many combinatorial problems that regularly occur in computing applications.

We normally think of complete problems to be the hardest in their class, so it is somewhat counter-intuitive to believe that some complete problems are harder than others. But as it turns out, the requirement that reductions between NP-complete problems take no more than polynomial-time leaves room for deterministic complexity distinctions. While worst-case, exact algorithms for most NP-complete problems require strongly exponential time, it appears that some can operate in sub-exponential (but still super-polynomial) time. The research literature on this topic, however, is not particularly consistent. The purpose of this paper is to review and clarify some of the results that have been published over the years, with the expectation that clarifications will eventually lead to a consensus on which NP-complete problems are harder than others.

2 The Meaning of Sub-Exponential Time

We can define a deterministic time class called SUBEXP that lies between P and EXP. It contains all languages in P plus all the languages whose deterministic deciders have time functions that grow faster than n^c, for any positive constant c (super-polynomial), but slower than c^n, for any constant $c > 1$ (sub-exponential). The function $f(x) = 2^{\sqrt{x}}$ is a good example of such a time function. We find various definitions for sub-exponential

in research articles. Stearns and Hunt [6] used the concept of power index for exponential functions. The power index for $f(x) = 2^x$ is one, while the power index for $f(x) = 2^{\sqrt{x}}$ is one-half. In general, the power index of $f(x) = 2^{x^i}$ is i, and we can call a function sub-exponential if its power index is strictly less than one.

In other articles (such as [2]), a function $f(x)$ is classified as sub-exponential if it is strictly less than 2^{cx} for every constant $c > 0$. This can also be expressed using little-oh notation: a function whose complexity is $2^{o(x)}$ can be called sub-exponential. With any of the asymptotic notations (O, Ω, Θ, o, ω), we may find an expression such as $2^{o(x)}$ used as alternative to $O(2^x)$. The former notation is useful for characterizing time functions for combinatorial problems, which frequently combine a polynomial function $p(x)$ with an exponential function 2^x. It is not technically correct to say that a time function $t(x) = p(x) \cdot 2^x$ is $O(2^x)$. The polynomial factor makes the function exceed $c \cdot 2^x$ for any constant c. But certainly, for large enough x, $p(x) \cdot 2^x < 2^x \cdot 2^x$, and $2^x \cdot 2^x = 2^{2x}$, so we can say that $p(x) \cdot 2^x$ is $2^{o(x)}$. We also find O^*-notation used to suppress polynomial factors in complexity classifications. Gerhard Woeginger [7], for example, uses $O^*(T(m(x)))$ as a shorthand for $O(T(m(x)) \cdot p(|x|))$.

We can provide one more alternative for the definition of sub-exponential growth based on the quotient $f(x)/f(x\text{-}1)$. This definition facilitates simple empirical verification of sub-exponential complexity, which is perhaps easier than solving for constants in the definitions of asymptotic notation. For polynomial functions, the quotient $f(x)/f(x\text{-}1)$ decreases, approaching the limit of 1, as x increases. For exponential functions of the form $f(x) = c^n$, the quotient is constant at $f(x)/f(x\text{-}1) = c$. Sub-exponential functions

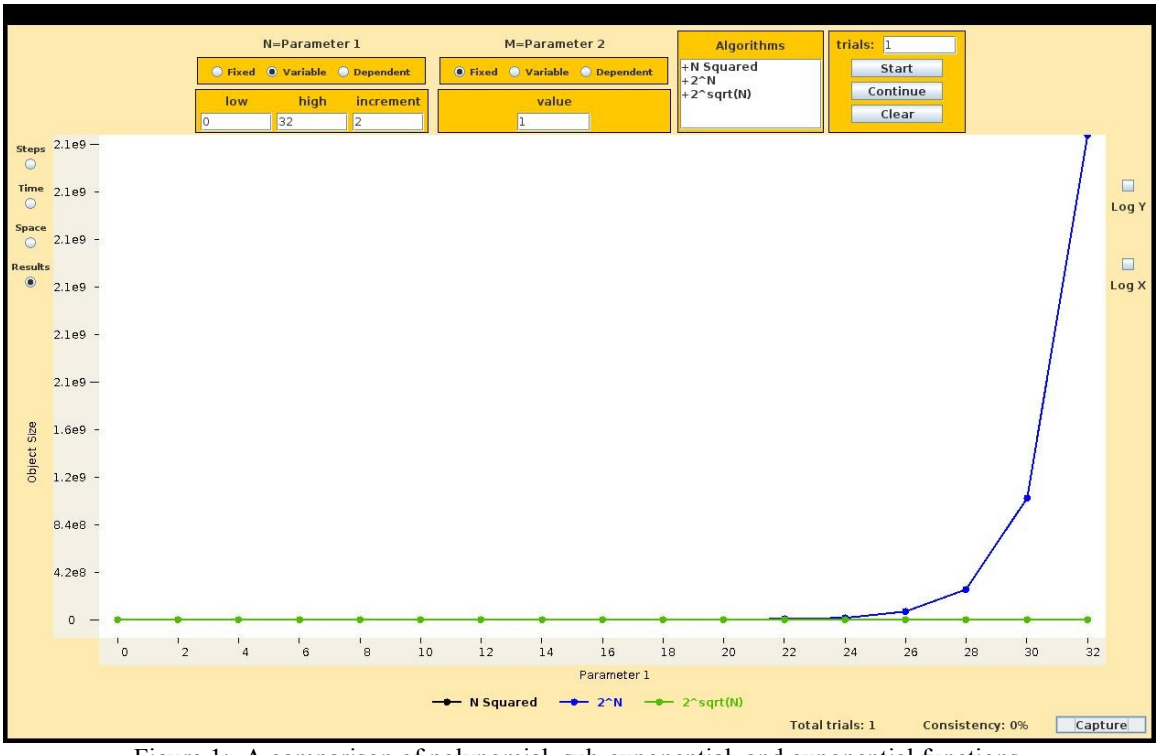

Figure 1: A comparison of polynomial, sub-exponential, and exponential functions.

behave like polynomials with respect to this quotient -- it decreases approaching 1 as x increases. We can easily distinguish a truly sub-exponential function like $f(x)=2^{\sqrt{x}}$ from a function like $(1/x)\cdot 2^x$, which appears to grow more slowly than $c\cdot 2^x$. For $f(x) = (1/x)\cdot 2^x$, the quotient $f(x)/f(x\text{-}1)$ increases, approaching 2 as x increases, rather than decreasing towards 1. So $f(x) = (1/x)\cdot 2^x$ is exponential, as it can be shown algebraically to be $O(c^x)$ for a constant $1 < c < 2$.

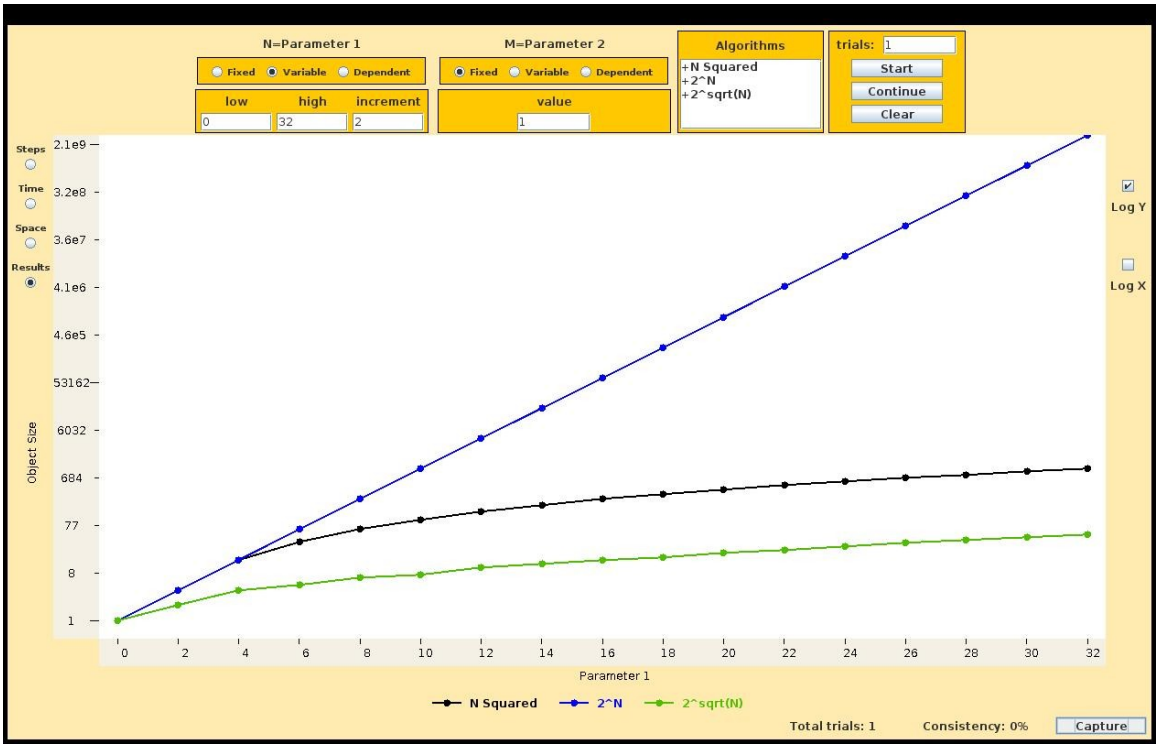

Figure 2: Exponential-scale comparison of polynomial, sub-exponential, and exponential functions.

Finally, we consider whether sub-exponential time is a significant improvement over exponential time. In light of the analysis in the previous paragraph, we might expect sub-exponential functions to behave more like polynomials, and, for at least a while longer as x increases, this is the case. Stearns and Hunt [6] observe that sub-exponential algorithms can be run on data sets with thousands of items, compared to perhaps hundreds of items for exponential algorithms. A comparison of exponential, sub-exponential, and polynomial growth can be found in Figures 1 through 3. Figures 1 and 2 present two views of the same information -- with a standard scale in Figure 1, and with an exponential y-scale in Figure 2. They compare the values of three functions, x^2, 2^x, and $2^{\sqrt{x}}$ for values of x between 0 and 32. The highest curve is 2^x in both figures. It grows so fast that the scaling factor needed to accommodate it makes the other two curves overlay one another in Figure 1. With the exponential y-scale in Figure 2, the curves for x^2 and $2^{\sqrt{x}}$ are distinct, with $2^{\sqrt{x}}$ at the bottom. The sub-exponential function actually has lower than quadratic values in this range. Figure 3 extends the comparison of x^2 and $2^{\sqrt{x}}$ beyond the x-value at which they cross, which is about 250. From this point onward, $2^{\sqrt{x}}$ rapidly becomes higher than x^2. But it's worth noting that for data sets with a few hundred items, a sub-exponential algorithm might actually out-perform a

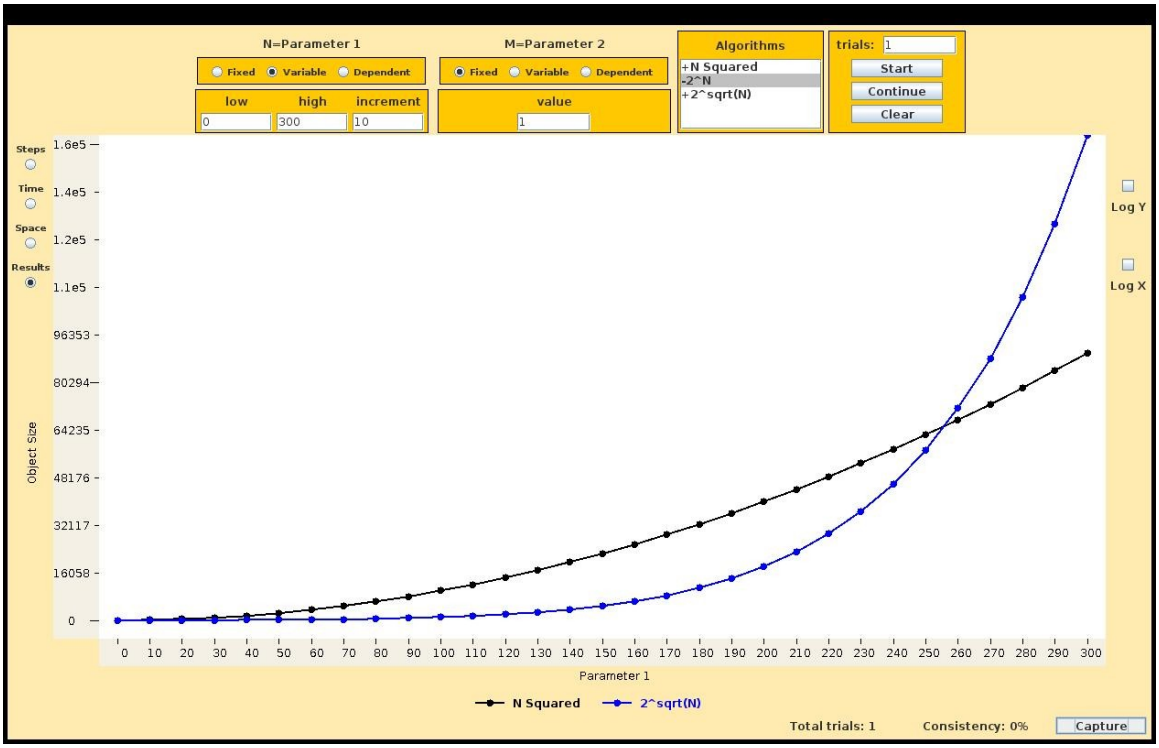

Figure 3: Quadratic vs. sub-exponential growth.

quadratic algorithm. For some applications this might deliver exact solutions to combinatorial problems in a reasonable amount of time.

3 Which NP-Complete Problems are Sub-Exponential?

Two major research contributions can be found in the research literature, about a decade apart, dealing with the question of which NP-complete problems are easier than the others. Both make the assumption, consistent with widely-held expectation, that the Satisfiability problem has strongly exponential time complexity -- $2^{\Omega(n)}$ for a Boolean expression with n variables. This is called the Satisfiability Hypothesis. Given this conjecture, the authors determine what other problems might be easier or not easier than Satisfiability. According to the first of these papers [6], the Clique problem and the Partition problem are sub-exponential with power index one-half. According to the second [2, 3], Clique is strongly exponential, and no problems are identified as sub-exponential.

The Clique problem is the problem of finding the largest complete subgraph in an undirected graph. The Partition problem is the problem of determining whether a set of positive integers can be divided into two subsets with the same sum. The two problems are not closely related, except that both are NP-complete. In [6], Stearns and Hunt describe algorithms for these problems that have time complexity $2^{O(\sqrt{x})}$ where x is the length in bits of the problem input. In the case of Clique, the input is a list of edges (the adjacency list for the graph) and the algorithm is simple recursive backtracking. It is

proven by induction that the number of recursive calls in the backtracking program is no more than $n \cdot 2^{\sqrt{2e}}$ where e is the number of edges in the graph and n is the number of nodes. For the Partition problem, the input is a list of numbers and the algorithm is a hybrid that employs both backtracking and dynamic programming. Suppose the input list contains n numbers with maximum m, and let b be the bit length of the entire list (then $b \leq n \cdot \lg m$). The list of numbers is divided into two subsets A and B, where B contains all the numbers with fewer than $\sqrt{b}$ bits, and A contains the numbers with $\sqrt{b}$ bits or more. Backtracking is applied to A, dynamic programming is applied to B, and the results are combined to determine whether a partition can be found. This hybrid approach exploits the fact that backtracking works better on a sparse list (a shorter list relatively large numbers) and dynamic programming works better on a dense list (a longer list of relatively small numbers). It is shown that the number of steps for the hybrid algorithm is bounded by $b^2 \cdot 2^{\sqrt{b}}$. So both Clique and Partition have algorithms with sub-exponential time functions when the bit length of the input is the complexity parameter.

About a decade later, Impagliazzo, Paturi, and Zane [3] published a study that sought to determine which NP-complete problems have strongly exponential complexity (under the assumption that Satisfiability is strongly exponential). They noted that incremental improvements in the $O(p(n) \cdot c^n)$ exponential complexity for many NP-complete problems raised the issue of whether strict lower bounds for the values of c actually existed. To address this issue, they defined a form of reduction that would preserve sub-exponential time complexity. This family of reductions is called SERF (sub-exponential reduction family), and the authors defined two problems to be SERF-equivalent if there is a SERF reduction between them. The reductions are defined to have an explicitly specified complexity parameter, so measures like the number of nodes or the number of edges in a graph can be used instead of input length. The SERF reductions do not change the size of the complexity measure by more than a constant factor, thus insuring that the reduction has linear space complexity when input length is the measure. The time complexity of the reductions can actually be super-polynomial, provided it remains sub-exponential. The main result is that a large number of NP-complete problems, including Clique, are SERF-equivalent to k-Satisfiability.

The proof that Clique and k-Satisfiability are SERF-equivalent directly clashes with the claim that Clique, but not Satisfiability, has sub-exponential complexity. The authors address this conflict by suggesting that input length is not an appropriate complexity measure for distinguishing exponential from sub-exponential complexity. This seems a little disconcerting, since input length is the classic complexity measure that is used for the definition of all the complexity classes, including P, NP, and NP-complete. While Impagliazzo, Paturi, and Zane do not explicitly address whether Partition is sub-exponential, their critique of Stearns and Hunt's Clique analysis implies that the Partition analysis is also flawed. An earlier version of the Impagliazzo, Paturi, and Zane article [2] contains the statement that the Subset Sum problem is SERF-hard, implying that Partition (which is a special case of Subset Sum) is not appropriately classified as sub-exponential.

A few years later, Gerhard Woeginger [7] published a survey of exact algorithms for many NP-complete problems. His introduction states the commonly held belief that

super-polynomial time is the best we can hope for with regard to these problems, and that while there is a scattering of results for different problems across the literature, there is no general theory of how the problems relate to each other, and there is no explanation for the wide variation in the values of c in complexities of the form $O(p(n)\cdot c^n)$. Figure 4 below shows the best known complexities for some of the problems discussed by Woeginger.

Problem	**Best known c for complexity $O(p(n)\cdot c^n)$**
Travelling Salesman	2
k-Colorability in graphs	2
Max-Cut in graphs	2
Bandwidth in graphs	2
Max-Cut with maximum degree 3	1.5
3-Satisfiability	1.48
3-Colorability	1.33
Subset Sum or Knapsack	1.14
Maximum Independent Set	1.1

Figure 4: Base constants for exponential complexities of some NP-complete problems.

Woeginger's survey does not mention the Partition problem, nor does it cite sub-exponential time for any of the problems it discusses. The complexity parameter n for the Subset Sum complexity in Figure 4 is the number of integers in the input list, not the total bit length of the input list. Woeginger summarizes the results of Impagliazzo, Paturi, and Zane in a discussion of how to prove that a problem has no sub-exponential time exact algorithm. Woeginger offers as an open problem whether a proof that one of the SERF-complete problems is sub-exponential would imply that P = NP. Woeginger concludes that for all the literature generated about NP-complete problems over the past fifty years, the study of their exact solutions remains a rich and promising area for research.

4 Attempting to Resolve the Inconsistency

It is possible to shed some light on the inconsistent complexity results for the Clique problem by examining the effects of a symmetric representation for the edge list of a graph. This approach is described in detail in [5]. When a set of objects is dense, that is, when it contains more than half of the universe of possible objects, a simple list of objects is not the most efficient representation. For a dense set, the list of missing objects would be shorter, perhaps significantly shorter. It is easy to devise a list with a header that identifies the universe of possible values, has a field indicating the polarity of the list, and enumerates whichever is shorter -- the set or its complement. A list with positive polarity is the same as a conventional list. A list with negative polarity is a list of all

missing items. A polarized list is one of many representation schemes that can be characterized as symmetric. With symmetric representation, the length of the representation for a set is the same as the length of the representation for the set's complement.

If we use symmetric representation for the edge list in a graph, we can no longer claim sub-exponential complexity for Clique. The worst case occurs for a dense graph, where the number of edges e is $\Theta(n^2)$, where n is the number of vertices, and the number of missing edges is $\Theta(n)$. In this case the length of the symmetric adjacency list is $\Theta(n)$, and the number of steps in the backtracking algorithm is $n \cdot 2^{\sqrt{2e}}$, which is $2^{O(\sqrt{n^2})}$, or $2^{O(n)}$. Since a symmetric list is more space efficient than a conventional adjacency list, we can claim that it is the most appropriate representation to use for complexity classification. So it looks like Impagliazzo, Paturi, and Zane are right with regard to Clique -- it has the same complexity as k-Satisfiability. However, we have achieved this result without disputing the use of input length as a valid complexity parameter.

It is interesting to pursue this same analysis for the Partition problem. Symmetric representation can be easily applied to a list of numbers, and the representation length for a dense list can be significantly shorter than the conventional list. But the worst case for Partition is a sparse set of numbers, not a dense one. Solving Partition for a dense set is very easy. It appears possible to establish that Partition can be solved in linear time for any set that's dense enough to have significantly shorter representation using a symmetric list. This would allow us to devise an algorithm that adapted its method to the density of the input set, achieving sub-exponential complexity for sparse and dense sets alike, even under symmetric representation.

5 Conclusion

In conclusion we share Gerhard Woeginger's view that we still have a lot to learn about exact solutions for NP-complete problems. There is a consensus that exact algorithms will remain super-polynomial, but there is no apparent consensus on whether some of the hard problems have sub-exponential complexity, particularly Partition, Subset Sum, and various related problems that are known to have pseudo-polynomial time solutions. Anticipating that such a consensus will eventually emerge, we would expect the relationship between the non-deterministic and deterministic time classes to look something like the diagram in Figure 5 below. The non-deterministic classes are indicated with dashed lines. NP is somewhere within EXP, containing all of SUBEXP and P. The NP-complete class is inside NP and outside of P, spanning the boundary between SUBEXP and EXP.

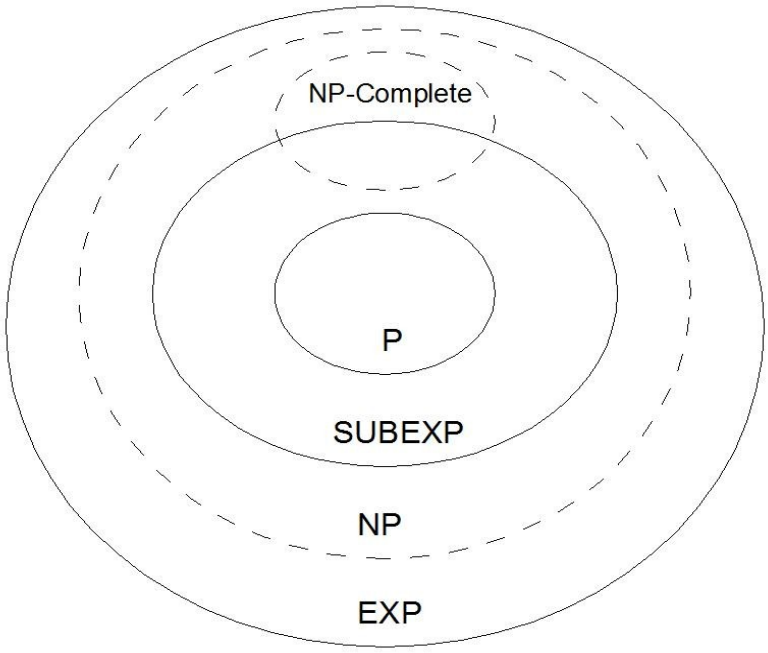

Figure 5: Conjectured relationship between time classes.

References

[1] S. Cook, "The Complexity of Theorem-Proving Procedures", *Proceedings of the Third ACM Symposium on Theory of Computing*, pp. 151-158, ACM, New York (1971).

[2] R. Impagliazzo, R. Paturi, and F. Zane, "Which Problems Have Strongly Exponential Complexity?", *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pp. 653-662, IEEE (1998).

[3] R. Impagliazzo, R. Paturi, and F. Zane, "Which Problems Have Strongly Exponential Complexity?", *Journal of Computer and System Sciences* 63, pp. 512-530, Elsevier Science, USA (2001).

[4] R. Karp, "Reducibility Among Combinatorial Problems," in *Complexity and Computer Computations*, ed. R. E. Miller and J. W. Thatcher, pp. 85-103, Plenum Press, New York (1972).

[5] T. E. O'Neil, "The Importance of Symmetric Representation," *Proceedings of the 2009 International Conference on Foundations of Computer Science* (*FCS 2009*), pp. 115-119, CSREA Press (2009).

[6] R. Stearns and H. Hunt, "Power Indices and Easier Hard Problems", *Mathematical Systems Theory* 23, pp. 209-225, Springer-Verlaug, New York (1990).

[7] G. J. Woeginger, "Exact Algorithms for NP-Hard Problems: A Survey," *Lecture Notes in Computer Science* 2570, pp. 185-207, Springer-Verlaug, Berlin (2003).