NP Complete Problems – A Survey

Saranya Vatti **November, 2017**

“Once more, we have decreased the number of open questions in the field - without, alas, increasing much the number of answers!”
(C.H. Papadimitriou and M. Yannakakis, "Optimization, Approximation, and Complexity Classes." JCSS 43:425-440, 1991.)

Introduction

In 1936, Alan Turing invented something he called the a-machine – known to us as the Turing Machine. He defined a theoretical computational model that can be constructed into a machine, given any algorithm, that would edit the symbols on an infinite tape. This formed the foundation for the computational complexity theory that would be defined formally much later. A number of scientists worked on the problem of complexity and on solutions with proofs for the computational complexities. The paper by Hartmanis and Stearns that was based on the work of Turing, formally defined and established the field of computational complexity. Stephan Cook, now considered one of the forefathers of computational complexity theory, and Leonid Levin independently published papers that not only defined but also showed the existence of NP complete problems. In his paper, Cook gave the first proof for the decision problem, Boolean Satisfiability problem based on Turing reductions. In 1972, Richard Karp used Cook’s paper to publish “Reducibility among Combinatorial Problems,” consolidating 21 problems including Clique, Vertex Cover, Hamiltonian Cycle, Knapsack Problem to be NP-Complete. The was among the first of its kind and among the most important. It paved the way for many other notable scientists to build on existing proofs and to reduce and connect more problems.

NP-Completeness and its importance

Shortly after the spurge of initial papers that define and prove complexity theory, there came a volley of publications that give a more precise and a more formal definition of complexity classes and proofs. One of the classic book that covers most of the problems is Garey Johnson’s “Computers and Intractability: A Guide to the Theory of NP-Completeness”. Over time and with many man-efforts spent in trying to solve some hard problems that would later be classified as NP-Hard or NP-Complete, scientists gradually searched for a formal proof that the problems could not be solved as opposed to searching for an efficient solution. NP-Completeness filled this need for the proof.

By definition, NP-Complete problems are a set of problems that belong to both NP and NP-hard. A problem belongs to the “NP” complexity class if there is a Non-deterministic algorithm that can solve the problem in Polynomial time. Informally, the problem has to be verifiable in polynomial time but no known algorithm exists that can solve the problem in the polynomial time (order of growth being a polynomial function of the problem size). A problem belongs to NP-Complete if it is in NP and every problem in NP is reducible to it in polynomial time. NP-Complete problems are also among the hardest problems to solve. This makes the identification and reduction of NP-Complete problems all the more important. There are many real world algorithms and practical applications that demand the solution of any one of NP-Complete problems. These span a wide array of application areas. Having a repository of pre-reduced and pre-proved problems will enable future scientists to further the research on diverse problems, perhaps coming ever so close to solving them. With this classification and reduction, an advance in the algorithm of one problem can easily help improve the efficiency in the current solution of another problem.

Once a problem was proved to be NP-Complete, scientists could then look to other ways of solving the problem like approximating, heuristics or randomized algorithms. This was of a great boost to the Computer Science community as practical algorithms were invented, where none existed earlier. Computational problems are now classified into a number of classes and the relationships between the properties of problems within the same classes can be studied. The years of scientists' work on NP-Hard problems shows that there is a less probability of finding an optimum solution to such problems. Hence, more scientists now work on restricted/modified questions that will answer the real world needs rather than trying to find the optimal solution for the problem.

Knapsack Problem

The knapsack problem is an optimization problem: given a set of items each with a weight and a value, the objective is to maximize the value while having an upper bound on the weight W. The decision problem form of the knapsack problem is: given a set of items each with a weight and a value, can a value V be achieved while having the upper bound on the weight W? The decision problem is NP-complete, meaning there is no algorithm yet that can optimally answer the YES/NO question. In 1975, Sartaj Sahni published "Computing partitions with applications to the knapsack problem" which showed a polynomial time approximation scheme for the knapsack problem. Polynomial time approximation scheme takes an instance of the problem and provides a solution within a $1 + \varepsilon$ for some ε which is relatively a very small number. While Sahni's algorithm had a running time of $1/\varepsilon$, it provided a basis for a much improved algorithm – a fully polynomial time approximation scheme by Ibarra and Kim. Now, there are approximate/exact solutions to variations of the knapsack problem, most notable a dyanamic programming solution for the 0-1 knapsack problem (restricted number of each item to zero or one). Subset-sum, one of the 21 problems famously proved in Karp's paper and quite a few problems are reduced to subset-sum, which itself reduces to the knapsack problem. The knapsack problem is a basic problem seen in many areas – it can be seen as a resource allocation problem. It can be used to efficiently pack shipping containers, or getting the maximum returns from a set of stock shares, or getting the maximum value players with a salary limit in a daily fantasy sport game.

Conclusion

While it seems counter intuitive that proving a problem cannot be solved may actually help the client who needs the problem solved, the short history of NP-Completeness shows us that this is true. Once a problem's insolvability is proved and it is classified in other similar problems, scientists can turn to alternative problem statements with greater returns. In some cases, these suboptimal solutions do run better in practice than they have been proved formally on paper. Such algorithms still contribute a lot to the computing world and all the areas that it affects.

While NP-Completeness and classification of problems does open new avenues for potential scientific work, the two major and most important classes still remain in the limelight as the core quest in computational complexity. P vs NP problem is one of the seven problems still open for the Millenium Prize by the Clay Mathematics Institute with a one million dollar fund allocated to the winner who solves the $P \overset{?}{=} NP$ problem. Although not proven, it is believed that P (the class of problems that can be solved in some known polynomial function of the input time and space) is strictly greater than NP. Complexity theorists are still trying to get a concrete proof for this problem or to prove the separation of the two classes.

References

1. Wikipedia [https://en.wikipedia.org/wiki/List_of_NP-complete_problems; https://en.wikipedia.org/wiki/NP-completeness]
2. Johnson DS (2012) “A brief history of NP-completeness, 1954–2012”. Documenta Mathematica. Extra Volume ISMP, pp 359–376
3. Hartmanis, J., and R. E. Stearns, [1965]. "On the computational complexity of algorithms," Trans. Amer. Math. Soc., 117, 285--306.
4. Fortnow, L. and Homer, S. “A short history of computational complexity”. Bulletin of the European Association for Theoretical Computer Science 80, (June 2003).