Computing Partial Solutions to Difficult AI Problems

Roman V. Yampolskiy

Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville
roman.yampolskiy@louisville.edu

Abstract

Is finding just a part of a solution easier than finding the full solution? For NP-Complete problems (which represent some of the hardest problems for AI to solve) it has been shown that finding a fraction of the bits in a satisfying assignment is as hard as finding the full solution. In this paper we look at a possibility of both computing and representing partial solutions to NP-complete problems, but instead of computing bits of the solution our approach relies on restricted specifications of the problem search space. We show that not only could partial solutions to NP-Complete problems be computed without computing the full solution, but also given an Oracle capable of providing pre-computed partial answer to an NP-complete problem an asymptotic simplification of problems is possible. Our main contribution is a standardized methodology for search space specification which could be used in many distributed computation project to better coordinate necessary computational efforts.

Keywords: *NP-Complete, Partial Solution, Search Space*

Introduction

In "*Computing from Partial Solutions*" Gal et al. (Gal, Halevi et al. 1999) consider the question: "Is finding just a part of a solution easier than finding the full solution?" For NP-Complete problems, such as 3-CNF, they prove that finding a fraction of the bits in a satisfying assignment is as hard as finding the full solution. Specifically they proof that any CNF formula F can be encoded in another formula F', is such a way that given a small fraction of bits in a satisfying assignment to F', it is possible to recover a full satisfying assignment to F (Gal, Halevi et al. 1999):

Theorem 1: For any $\varepsilon > 0$, there exist an efficient probabilistic algorithm A, and an efficient deterministic algorithm B such that:

1. If F is a CNF formula over n variables, then $F' = A(F)$ is a CNF formula over $N = n^{O(1)}$ variables, with $|F'| = |F| + n^{O(1)}$.

2. With probability $1-2^{-n}$ the formula F' has the following property: If s' any assignment to $N^{.5+\varepsilon}$ of the variables in F' which can be extended to a full satisfying assignment, then $B(F,F',s')$ is a satisfying assignment for F.

Proof of Theorem 1. If we are given polynomial number of random linear equations in n variables, then any sufficiently large subset of these equations is of dimension at least $n-O(\log n)$, and thus leaves only a polynomial number of candidate solutions to the entire equation system. This, combined with the ability to verify solutions, yields an 'erasure-code' with the ability to correct $n - sqrt(n)$ erasures. This improved erasure code can be used to satisfy the claims of Theorem 1. Which was to be shown (Gal, Halevi et al. 1999).

The result is hardly surprising since if finding a part of the solution was possible in polynomial time P = NP would trivially follow. In fact numerous researchers have realized that a related problem of NP-Complete problem re-optimization is not polynomial time solvable unless P = NP (Archetti, Bertazzi et al. 2003; Böckenhauer, Forlizzi et al. 2006; Kralovic and Momke 2007; Ausiello, Escoffier et al. 2009). The proof of that fact due to Archetti et al. follows (Archetti, Bertazzi et al. 2003):

Theorem 2: No polynomial time algorithms can exist for the Re-Optimization of TSPunless P = NP.

Proof of Theorem 2, by contradiction. Suppose that there exists a polynomial time algorithm, for example ReOptTSP, which accomplishes the Re-Optimization of TSP. Then, an optimal solution of any TSP with $n+1$ nodes can be obtained in polynomial time by applying $n-2$ times the algorithm ReOptTSP. We begin by applying the algorithm ReOptTSP to find an optimal solution of the Re-Optimization of TSP with 4 nodes, given that any 3-city TSP problem is trivially optimally solvable. Then, ReOptTSP is applied to find an optimal solution of the Re-Optimization of TSP with 5 nodes, given an optimal solution of the TSP with 4 nodes, and so on until it is applied to find an optimal solution of the Re-Optimization of TSP with $n+1$ nodes. Thus, by contradiction, no polynomial time algorithms exist for the Re-Optimization

of TSP unless P = NP. Which was to be shown (Archetti, Bertazzi et al. 2003).

In this paper we look at a possibility of both computing and representing partial solutions to NP-complete problems, but instead of considering bits of the solution our approach relies on specifications in the problem search space. We show that not only could partial solutions to NP-Complete problems be computed without computing the full solution, but also given a pre-computed partial answer to an NP-complete problem an asymptotic simplification of the problem is possible. Our main contribution is a standardized methodology for search space specification which could be used in many distributed computation project to better coordinate remaining computational efforts. NP-Complete problems are inherently easy to parallelize and so could benefit from a common language aimed at describing what has already been evaluated and what remains to be analyzed.

Search Space Specification

Gal et al. conclusively demonstrate that computing a part of an answer to an NP-Complete problem is as difficult as computing the whole solution (Gal, Halevi et al. 1999) their results are further reaffirmed in (GroBe, Rothe et al. October 4-6, 2001). We propose representing a solution to an NP-Complete problem as a mapping of the total search space subdivided into analyzed and unsearched parts. A full solution to an NP-Complete problem can be represented by the sequential number of the string in an ordered set of all potential solutions. A partial solution can be represented by the best solution found so far along with the description of the already searched potential solutions. The already searched space need not be continuous; the only requirement is that the remaining search space could be separated from the already processed regions. It is easy to see while the smallest possible partial solution can be computed in constant time (this requires analyzing only one potential answer) progressively larger partial solutions are exponentially harder to compute with respect to the size of the problem.

Let's analyze a specific example of our representation of partial solutions. Travelling Salesperson Problems (TSPs) are easy to visualize and make for an excellent educational tool. Let's look at a trivial TSP instance with 7 cities numbered from 1 to 7 as depicted in Figure 1. Assuming that the first city in the list is connected to the last one, potential solutions can be represented as a simple numbered list of cities: [1, 2, 3, 4, 5, 6, 7]. The complete search space for our problem consists of all possible permutations of the 7 cities. This complete set could be trivially ordered lexicographically or by taking the value of the permutation condensed into an integer form resulting in non-continuous numbers from 1234567 to 7654321. The position number in which a potential solution appears in the list could be taken as a pointer to that specific solution, with solution 1 refereeing to the [1 → 2 → 3 → 4→ 5 → 6 → 7 → 1] path in our example and solution 2 mapping to [1 → 2 → 3 → 4→ 5 → 7 → 6 → 1], and so on. It is obvious that the same approach can be applied to other NP-Complete problems as they all could potentially be reduced to an instance of TSP. Alternatively a specialized representation could be created for any problem as long as it could be mapped on a countable set of integers. The specific type of ordering is irrelevant as long as it is reproducible and could be efficiently included as metadata accompanying any partial solution.

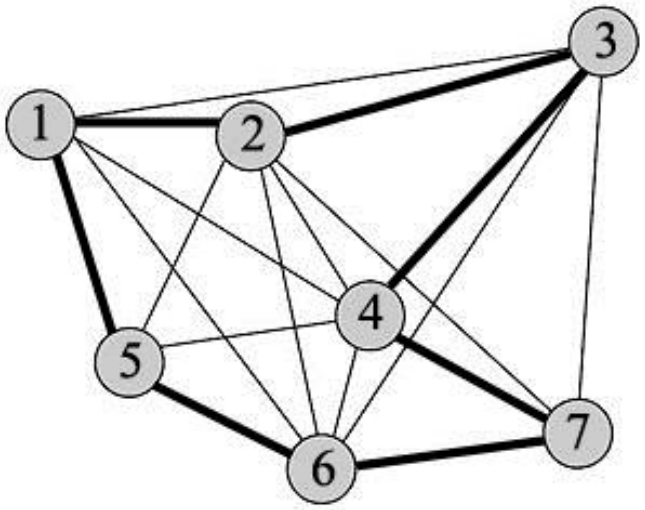

Figure 1. A seven city instance of TSP

Given an ordered set of potential solution it is trivial to specify the regions which have already been processed. In our 7 city TSP example the total search space consist of 7! = 5040 possible paths. A partial solution may be represented by stating that solutions from 1 to 342 have been examined and the best solution is the one in position 187. This means that 4698 additional path remain to be examined and that the partial solution could be said to represent 6.79% of the final answer.

A simple visual diagram can be used to illustrate computed partial solution via visualization of the search space. In Figure 2 a search space of 400 potential solutions is converted to a 2D representation by tilling 20-unit blocks of solutions on top of each other. Solution number 1 is represented by the top left most square with other solutions being introduced sequentially from left to right. Bottom right square is the potential solution numbered 400. Black squares represent already analyzed solutions. White squares are yet to be processed. The example in Figure 2 is a particularly non-contagious partial solution, having no 3 or more continuously examined candidate solutions in a row.

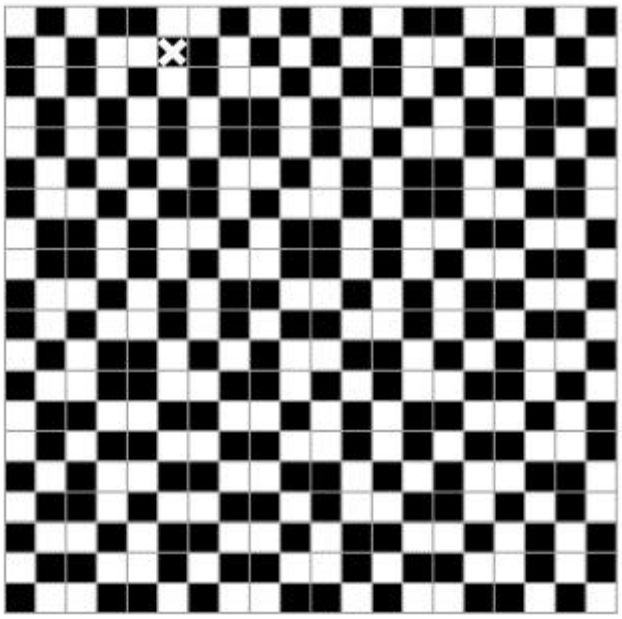

Figure 2. 2D visualization of the search space with respect to searched/unsearched regions and optimal solution found so far indicated by an X.

Pre-computed Partial Solutions

A more natural way of representing partial solution is to directly look at a subset of bits comprising the answer to the problem. Unfortunately finding such bits is as hard as solving the whole problem (Gal, Halevi et al. 1999) and so makes computation of partial solutions represented in this way unfeasible for NP-Complete problems. But suppose that such a partial solution could be computed by an Oracle and provided to us at no additional computational cost. In this section we will look at such situation and analyze difficulty of NP-Complete problems with supplied partial answers.

Returning to our 7-city TSP example and using decimal instead of binary representation of solution (for better readability) a partial solution could be represented as: [1, 2, ?, ?, ?, 6, 7], where "?" represent missing information. The missing information need not be continuous as in: [?, ?, 3, ?, 5, ?, 7] and in the most trivial cases ([1, 2, 3, 4, 5, 6, ?]) may be computed in a constant number of steps. Under this encoding for partial solutions an Oracle may provide enough help to make the problem solvable either in constant time (a small in comparison to N constant number K of missing elements), polynomial time (log N of missing elements), or essentially provide no help by providing only a constant amount of information ([?, ?, ?, 4, ?, ?, ?]). Essentially the Oracle for finding partial solutions to NP-Complete problems has the power to make a problem as easy to solve as it desires, all the way up to single computation.

Conclusions

In this paper we presented a novel way of representing solutions to NP-Complete problems in terms of search space subsets. The proposed methodology allows for easy parallelization of difficult computational problems and is not limited only to NP-Complete problems. Any computational effort can be expressed in terms of search space locations making such computationally intensive projects as Prime Search (mersenne.org), BitCoin (bitcoin.org), Factoring (escatter11.fullerton.edu/nfs), SETI (setiathome.berkeley.edu), Protein Folding (folding.stanford.edu), Game Solving (Schaeffer, Burch et al. September 2007), TSP (Yampolskiy and EL-Barkouky 2011) and Theorem Proving by Computer (Appel, Haken et al. 1977) easier to formalize, verify and break up among numerous computers potentially separated in space and time. While the projects mentioned above all have an internal way of representing the unexplored search space, a common way of specifying such information may lead to standard software capable of shifting unused computational resources among all such efforts.

The proposed solution encoding approach does not represent a breakthrough in our ability to solve NP-Complete problems (Yampolskiy 2011) but it does provide a way to store partial solutions to computationally challenging problems some of which may span decades of effort (Schaeffer, Burch et al. September 2007). Consequently, we are no longer limited to describing particular instances of such problems as solved or unsolved but we can also talk about percentage of the solution we have obtained so far. In the future we plan on addressing such issues as compressibility of representations for multiple non-contiguous sectors in the search space as well as looking into finding optimal orderings for the space of possible solutions to the NP-Complete problems. Additionally, we would like to investigate if by combining our approach with such methods as Monte Carlo simulation (over multiple small partitions of the search space) one can quickly arrive at sufficiently good solutions to very hard problems in cases where optimal solutions are not required.

References

Appel, K., W. Haken, et al. (1977). "Every Planar Map is Four Colorable." Illinois Journal of Mathematics 21: 439-567.

Archetti, C., L. Bertazzi, et al. (2003). "Reoptimizing the Traveling Salesman Problem." Networks 42(3): 154-159.

Ausiello, G., B. Escoffier, et al. (2009). "Reoptimization of Minimum and Maximum Traveling Salesman's Tours." Journal of Discrete Algorithms 7(4) 453--463.

Böckenhauer, H.-J., L. Forlizzi, et al. (2006). Reusing Optimal TSP Solutions for Locally Modified Input Instances 4th IFIP International Conference on Theoretical Computer Science (IFIP TCS).

Gal, A., S. Halevi, et al. (1999). Computing from Partial Solutions. Fourteenth Annual IEEE Conference on Computational Complexity: 34-45.

GroBe, A., J. Rothe, et al. (October 4-6, 2001). Relating Partial and Complete Solutions and the Complexity of Computing Smallest Solutions. 7th Italian Conference on Theoretical Computer Science. Torino, Italy, Springer-Verlag: 339-356.

Kralovic, R. and T. Momke (2007). Approximation Hardness of the Traveling Salesman Reoptimization Problem. 3rd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science: 97-104.

Schaeffer, J., N. Burch, et al. (September 2007). "Checkers is Solved." Science 317(5844): 1518-1522.

Yampolskiy, R. V. (2011). "Construction of an NP Problem with an Exponential Lower Bound." Arxiv preprint arXiv:1111.0305.

Yampolskiy, R. V. and A. EL-Barkouky (2011). "Wisdom of Artificial Crowds Algorithm for Solving NP-Hard Problems." International Journal of Bio-Inspired Computation (IJBIC) 3(6): 358-369.