A SUBLINEAR APPROXIMATION METHOD FOR NP-HARD PROBLEMS ON LIMITED HARDWARE

A PARADIGM SHIFT

Rohan Jhunjhunwala[*]
Undergraduate, Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720
`rjhunjhunwala80@berkeley.edu`

March 7, 2019

ABSTRACT

We present a straightforward polynomial time approximation method which is applicable to many problems considered conventionally intractable. This implementation is realizable under any model of computation, and any hardware schema. The breakthrough comes from the following realization. The winning move is not to play. We leverage existing work from Tseng[2] out of CMU in order to develop a new approximation method where we grossly misinterpret the problem, and deliver an efficient solution.

1 Introduction

Recall that decision problems involve an acceptance or rejection of an n length bit string. A P problem is one where this decision can be made in $O(n^k)$ time. An NP problem is one where the process of validating a proof of a yes answer to an n length string is in $O(n^k)$. NP Hard problems are problems, not necessarily decision problems, such that any NP problem solved in $O(n^k)$ time given an Oracle that solves the NP hard problem in constant time. The halting problem is NP Hard because any NP complete problem, reduces to 3Sat which is NP hard, and it's easy to construct a Turing machine which enumerates all the possibilities, and only halts if one of them satisfies the predicate. We now hope to solve NP Hard problems, to solve the classic, Halting problem, losslessly compress random data, and settle the age-old dilemma. Our hopes and dreams are summarized in the following theorem, which this paper will certainly prove!

Theorem 1

$$P \approx NP$$

2 Proof

In order to prove our theorem, we must first only demonstrate one algorithm to solve this problem for a single NP Hard problem. Then, the rest of the proof is left as an exercises to the reader. Without further ado, let's suggest the optimal algorithm.

```
Result: Solve NP-Hard Problems
Recieve inputs, a problem p, and an input i;
while $d<c$ do
    run fast
end
```

Algorithm 1: How to solve intractable problems. (Computational or otherwise)

[*]https://rohanjhunjhunwala.com/

The optimal algorithm, d being a quantity that defines the geodesic distance from the observer to the computer running the problem, and c being a reasonably large constant. Say one mile, or 1609 meters in a more 'conventional' set of units. Let's now determine the competitive ratio c of our algorithm. The largest home computers are on the order of a cubic meter (obviously less, but small, bounded constant factors are irrelevant for our asymptotic errors, especially since we want an upper bound), so we continue on for our analysis. Once $d = c = 1609$ the following equation gives us the error bound, because the size of the problem, is bounded physically by the size of our problem. In comparison our displacement is large therefore, the total error the size of the original problem is a small fraction of our actual world

$$e = 1/(1 + c)$$

$$e < .001$$

We've achieved an error bound on the order of

$$10^{-3}$$

and because we made no assumptions of the type of problem we've provided a $\theta(1)$ solution, which satisfies not only polynomial but sublinear runtime restrictions.

3 Explicit bounds, and further work

We still want an explicit runtime constant. Previous work by Hicham El Guerrouj sets our bound at 223 seconds as a best case implementation. The author considers himself to be a mediocore developer, and human being and has attained a 347 second leading constant, so we can consider that an average case, and a worst case can be attained by the reader by heading out to a track, and going for a walk. This naive implementation should offer a leading constant of around 900. Work by Einstein suggests that this leading constant can never be less than 1/186282 barring major changes in our paradigm. The last open question is whether or not an exact answer exists in constant time. For this we conjecture the following algorithm.

```
Result: Solve NP-Hard Problems
return 42;
```

Algorithm 2: How to solve intractable problems. (Computational or otherwise)

We also offer an alternate algorithm which involves $\theta(0)$ space, which involves doing exactly nothing until the heat death of the universe. Thus concludes our discussion, and I will collect my Turing award and Millennium prize. I accept payment in bitcoin, or actual proofs of other Millennium problems.

References

[1] Pessimal Algorithms and Simplexity analysis. https://www.mipmip.org/tidbits/pasa.pdf Andrei Broder and Jorge Stolfi

[2] Thomas Tseng. SIGBOVIK 2018. Sublinear colorings of 3-colorable graphs in linear time http://sigbovik.org/2018/proceedings.pdf

[3] Wikipedia, the free encyclopedia, Mile Time Progression. https://en.wikipedia.org/wiki/Mile run world record progression