Distributed Computing for Genetic Approximation Algorithms for the Kissing Problem (n-dimensions)

Andrew M.K Nassief
Genomics Researcher at Bleunomics
Quantum Engineer at the Lonero Foundation
58757 #100 VAN DYKE
RD WASHINGTON, MI 48094-9998

February 12, 2021

Abstract

Utilizing distributed computing for genetic optimization algorithms with n-dimensions and kissing numbers in their approximation, helps offload data and increase efficiency in approximation. In order to demonstrate this, we shall utilize mathematical proofs centered around N-dimensional vectors and arrays, as well as exponential dimensional analysis. Utilizing these proofs in optimization algorithms can have processed data offloaded through a shared network of computers running simultaneous multi-threaded computational processes. One can build a computational model based off of mathematical constraints viewed as higher dimensional complexity. Formulating such proofs is based off of degree of certainty versus uncertainty in the approximation, and which processing task should be optimized in order to yield the best result.

1 Introduction

Kissing numbers are the known greatest number of non-overlapping spheres that can be arranged in a space in which they touch each orther**[1]**. The higher the dimension, the higher the number of spheres. In regards to complexity, you are looking at exponential time 2poly(n) or time with a linear component 2O(n). The higher the dimension for sphere packing in n-dimensional space, the longer the time to find the number of spheres, and one doesn't know the time approximation to find said sphere numbers. The KNP problem, is a model problem in that area of formulas in regards to time complexity. Time optimization is hard when taking into account polynomial complexity. Exponential approximation algorithms have been built around the KNP problem**[2]**, and it is possible to look into computational algorithms with polynomial complexity. Since currently there is no publicised practical solution to predict when such problems can be solved, there is still a general idea in regards to the hardness of such problems.

We want to look into mathematical hardness in regards to optimization and approximation efficiency. The idea is to look into non-proven computational time hypothesises, and model approximation algorithms off of such said hypothesises. In fact, while different, some may argue that the KNP problem may be similar to a space complexity problem. One can think of dimensions running out of space for spheres instead of a computer. This paper will look at approximation and genetic algorithmic approaches to NP-completeness through the proposed ideal scenario of computational processing. Applying these forms of time complexity to parallel processing architecture will allow us to forumate an algorithmic model for proposed approximation efficiency.

1.1 Structure

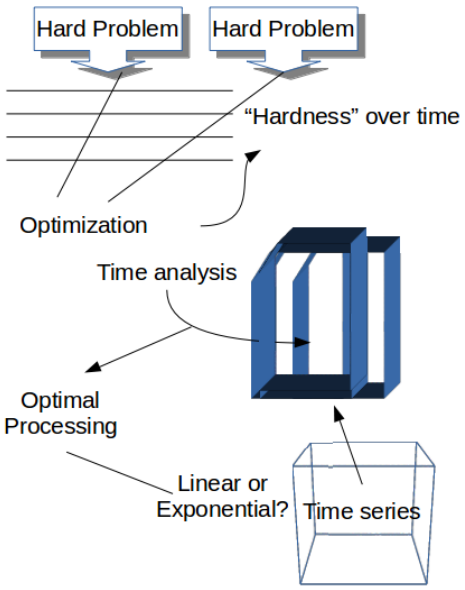

Figure 1: Structure UML

In regards to algorithmic structure, I believe the proposed algorithm can take complexity problems that increase in difficulty over time (n-dimensions), categorize certainty vs. uncertainty in regards to hypothetical hardness, and offload processing based on the category. One can follow a deterministic optimal flow over time based on the hypothetical hardness decision-making process. One can look at this algorithm as a time approximation algorithm, and in this case likely a Polynomial-time approximation algorithm or PTAS scheme**[3]**. In laymann terms, you are developing a pipeline in regards to computational hardness from a deterministic and probabalistic standpoint. Likely one can apply some sort of PCP Thoerem**[4]** or validation in regards to time series for attempted hardness problems.

1.2 Architecture

Likely one of the more optimal algorithmic structures for its ideal scenario, is an algorithm that runs through parallel processing architecture. More and more processing is offloaded in ways that focus on solving said complexity. You are offloading the data based off of what is determined as the most optimal way to potentially "soften" hypothetical hardness. In regards to genetic optimization, over time the decision making could exponentially improve as the complexity is becoming more difficult.
For example, if you look at the indefinite integral:

$$\int Log(2n)dn = n(log(2n) - 1) + constant$$

which is for log(2(n)), you will notice it is a complex logarithm for continuing functions. In exponentiation**[5]**, for complex bases, a complex logarithm is needed. Something to take into account is that in regards to complexity schemes, the higher the base number, the higher the complexity (such as kissing numbers). A deterministic algorithm, (while may not understand time), may understand hypothetical levels of complexity. As problems are being solved over time, it creates a sort of probabilistic determination in regards to where processing should be offloaded and whether to thread concurrently, multithread, or even create purposeful halts. The is similar to instructional pipelining or categorizing execution, buffer or batch processes. ILP concurrency**[6]** in regards to instructions also work quite similarly.

2.0 Usecases

Given that this is in the experimentation phase, we have still yielded optimal results. An example is a research firm with a propietary algorithm was able to increase the efficiency of their algorithm combining it with this algorithm for parallel processing and complexity pipelines. Other fields such as computational genomics and drug discovery have yielded positive results on somewhat low spec hardware. Other experimental tests were done in the cryptocurrency mining space to speed up crypto-mining.

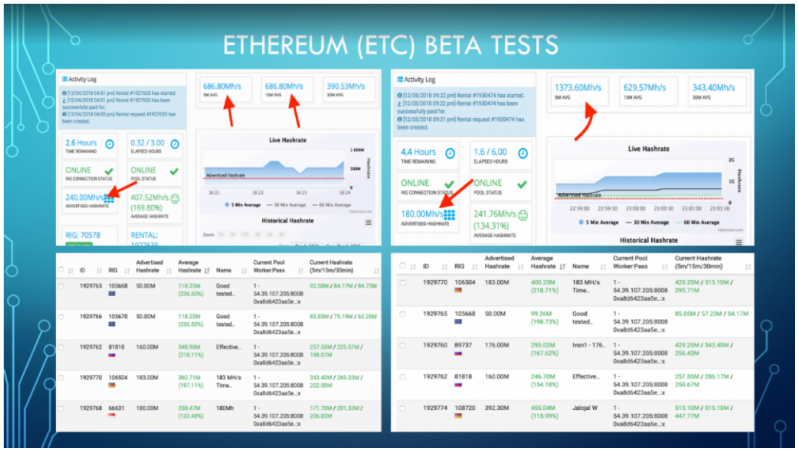

Figure 2: Cryptocurrency Mining Tests

The cryptocurrency startup, ChainTerra**(7)** was able to speed up cryptocurrency mining on a

very limited test[8] that also had latency issues given the usage of a data mining and visualization program. They were also able to apply the same tests to finding blocks on Peercoin in 2018 for SHA(256) mining.

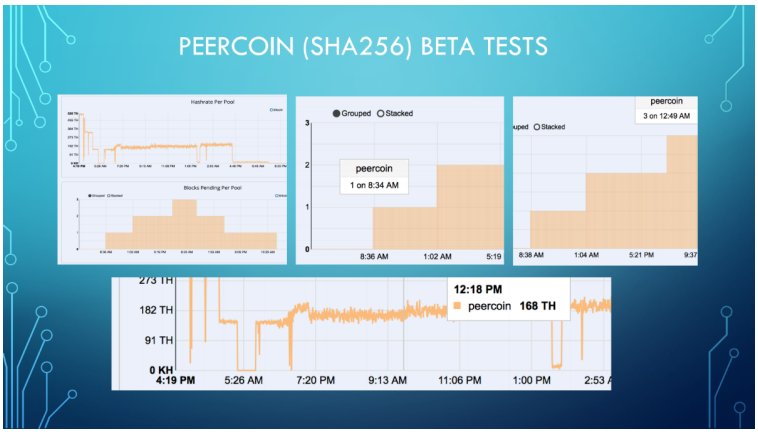

Figure 3: Peercoin Mining

In regards to crypto mining, this is a given. Fields such as computational drug discovery, have been able to utilize optimization in regards to quickly annotating various VCF files, protein modeling, and protein mining. An algorithmic pipeline like this, may also be utilized in various areas such as cryptography. In regards to processing time, this is optimal. It is also optimal with various forms of data compression. While data compression algorithms might not necessarily be able to increase compression ratio, (if specific algorithms already yielded high compression ratios), they may use it to offset processing time. The mining example was merged algorithmic pipelines. Both propietary research firms and universities as well, are working on similar optimization for processing times in regards to simulating super massive blackholes or space-time interaction.

References:

[1] Kissing Numbers - Numberphile. YouTube. Numberphile, 2018. https://www.youtube.com/watch?v=LZ7X_YOfJqY.

[2] Dr. Daniele Micciancio University of California, Panagiotis Voulgaris University of California, San Diego. “Faster Exponential Time Algorithms for the Shortest Vector Problem.” Faster exponential time algorithms for the shortest vector problem | Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms, January 1, 2010. https://dl.acm.org/doi/abs/10.5555/1873601.1873720.

[3] “Polynomial-Time Approximation Scheme.” Wikipedia. Wikimedia Foundation, February 1, 2021. https://en.wikipedia.org/wiki/Polynomial-time_approximation_scheme.

[4] Safra, S. Probabilistically Checkable Proofs Course. Accessed February 12, 2021. https://people.csail.mit.edu/dmoshkov/courses/pcp/index.html.

[5] “Complex Logarithm.” Wikipedia. Wikimedia Foundation, February 2, 2021. https://en.wikipedia.org/wiki/Complex_logarithm#cite_note-Ahlfors-1.

[6] “Instruction-Level Parallelism.” Wikipedia. Wikimedia Foundation, September 23, 2020. https://en.wikipedia.org/wiki/Instruction-level_parallelism.

[7] “Building The Future of Digital Currency Mining.” ChainTerra, December 12, 2018. https://www.chainterra.com/.

[8] “ChainTerra” Skillshare. Accessed February 12, 2021. https://www.skillshare.com/projects/170167.