A $O(2^{n/2})$ Universal Maximization Algorithm

Thesis by
Charles Dana

In Partial Fulfillment of the Requirements for the
Degree of
MSc X-HEC Entrepreneurs

HEC PARIS
Jouy-en-Josas, France

2023
Defended November 2023

Charles Dana
ORCID: 0000-0002-0364-5379

Statement 1.

$$\sup_{X \in \mathbb{X}} \inf_{w \in \mathbb{W}} \Pr(w(X) \neq f(X)) = 0$$

Statement 2.

$$\inf_{w \in \mathbb{W}} \sup_{X \in \mathbb{X}} \Pr(w(X) \neq f(X)) = 0$$

In other words, solving MAXSAT[**11**] Instances just got much more interesting. No matter the random variable $X : \Omega \to \mathbb{R}^n$, under the assumption that $X(\Omega)$ is discrete, there exists a model w such that the probability of discrepancy between f the cost function and w the model, can be driven down to zero. Note that Statement 2. is not always satisfied.

ABSTRACT

This Master's thesis introduces a novel framework for maximizing a cost function defined over various domains such as natural numbers, real numbers, complex numbers, and more. The framework leverages a set of Boolean variables to represent these domains and formulates optimization problems as weighted clauses involving these variables. The main idea is to use a local search-based algorithm to iteratively manipulate these Boolean variables to maximize the given cost function. The approach is demonstrated through various experiments including polynomial factorization, cryptography hash functions, elliptic curve cryptography, and factoring. The results indicate that the proposed method can achieve promising outcomes across different domains and problem types. The study highlights the potential of this approach for addressing complex optimization problems in diverse fields.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

This chapter outlines the notations and conventions employed within this article.

1.1 Common Notations

By the American Convention, $\mathbb{N}$ represents positive integers, excluding zero. $\mathbb{R}$ stands for the set of real numbers, while $\mathbb{C}$ denotes the set of complex numbers. $\mathbb{R}^n$ signifies the set of vectors containing n real numbers. Additionally, due to the frequent use of the modulo two operator in various formulas, it will be introduced as expressed in most programming languages: $x\%2 = 1_{x\in 2\mathbb{Z}+1} = 1_{x \text{ is odd}}$ where $x \in \mathbb{Z}$.

1.2 Definition of a Logical Condition Operator

Let $A(x) \in \{\top, \bot\}$ denote a property of the variable x. In this article, the following notation will be employed:

$$[A(x)] = 1_{A(x)=\top} = 1 - 1_{A(x)=\bot}$$

Here, $\top$ represents an always true tautological statement, and $\bot$ represents an always false contradictory statement.

1.3 Definition of a Densely Countable Subset of $\mathbb{R}^n$

Consider $E \subsetneq \mathbb{R}^n$ as a countable subset of $\mathbb{R}^n$. Within this article, E is deemed dense within $\mathbb{R}^n$ if we can present a Cauchy sequence of elements from E that converges to any element $x \in \mathbb{R}^n$. This characteristic is implied by the following assertion:

$$\forall x \in \mathbb{R}^n, \inf_{e\in E} \|x - e\| = 0$$

1.4 Definition of Algorithm Complexity

In the analysis of a generic cost-function Global Optimization Algorithm, establishing a foundational measure for the algorithm's complexity becomes essential. The assumption made here is that the evaluation of the cost function f is $O(1)$, meaning that the number of operations is bounded by a constant value. Consequently, an upper bound on the overall algorithm complexity amounts to $O(2^{n/2})$ evaluations of the cost function.

Chapter 2

OVERVIEW OF STATE-OF-THE-ART GLOBAL MAXIMIZATION ALGORITHMS

This chapter presents the fundamental principles that can be employed to address the general task of maximizing a cost function, utilizing strategies of increasing complexity. It is an overview of generic optimization techniques, it is not exhaustive.

2.1 Gradient Descent: A Greedy Approach to Local Minima

Gradient descent stands as a widely employed optimization method that aims to locate local minima within a given cost function. Its process involves iterative movement in the direction of the most rapid decrease in the function's value. Nevertheless, it can encounter difficulties in escaping local minima.

2.2 Local Search: Enhancing Local Minima Solutions

Local search algorithms endeavor to enhance existing solutions by exploring neighboring options within a specific range. This strategy is particularly advantageous for refining solutions generated through other techniques, such as gradient descent. The local search approach aids in avoiding local minima and elevating the overall quality of the solution.

2.3 Simplex Algorithm: A Swift Alternative to Gradient Descent

The simplex algorithm[**4**] serves as a linear programming method geared towards optimizing linear functions under linear constraints. In contrast to gradient descent, which operates on continuous functions, the simplex algorithm is ideally suited for discrete optimization problems. It efficiently explores feasible solutions confined within a convex polytope.

2.4 MAXSAT Solvers: NP-Hard Complexity, Approximation Strategies

MAXSAT solvers[**10**][**11**][**12**] are tailored for addressing NP-hard optimization problems, wherein the objective is to discover assignments to Boolean variables that maximize the count of satisfied clauses in a Boolean formula. These solvers make use of diverse heuristics and approaches, frequently integrating local search or approximation algorithms. They focus on obtaining approximate solutions[**12**].

Chapter 3

$\mathbb{W}$ AND GLOBAL MAXIMIZATION OF A GIVEN COST FUNCTION

This chapter presents a MAXSAT Optimization model involving positively-real-weighted clauses for a given cost function. We denote the set of such functions as $\mathbb{W} = \{w : w(x) = c + \sum_{i=1}^{M} w_i C_i(x)\}$.

3.1 Introduction of $\mathbb{W}$

Consider a subset C of non-zero integers within the range $[-n, n]$, denoted as $C \subset \{-n, 1-n, \ldots, -1, 1, \ldots, n-1, n\}$. We refer to C as a clause, represented as $C : x \mapsto \{0, 1\}$. Utilizing the binary decomposition of $x \in \mathbb{N}$, such that $B_x = \{l_1, \ldots, l_m\}$ and $x = \sum_{k=1}^{m} 2^{l_k - 1}$, we can state that:

$$C(x) = 1 - \prod_{l \in C} ([l < 0][-l \in B_x] + [l > 0][l \notin B_x])$$

$$C_{\{-1,2\}}(x) = 1 - [1 \in B_x][2 \notin B_x]$$

In simpler terms, a clause (a finite subset of non-zero integers) is satisfied if and only if either one of its positive elements ($l > 0$) is present in B_x, or one of its negated elements ($l < 0$) is not negated in B_x.

$$\mathbb{W} = \left\{ w : w(x) = c + \sum_{i=1}^{M} w_i C_i(x), c \in \mathbb{R}, w_i > 0, C_i \subset \mathbb{Z} \cap [-n, n] \right\}$$

3.2 Maximization of $w \in \mathbb{W}$: Theoretical Background

This section introduces a Python implementation of the "solve" function, which aims to efficiently solve a cost function $w \in \mathbb{W}$. The "problem" parameter is an array of pairs $[w_i, C_i]$, where $C_i = [l_1, l_2, \ldots]$ is an array, and $w_i > 0$ is a positive real weight. The algorithm operates similarly to an $A*$ search for pathfinding. It utilizes the "solution" variable and compares weights to the current average score. With a limited number of considered variables, it efficiently explores the solution space, seeking better-than-average solutions. Importantly, the "shuffle(problem)" action before each iteration introduces shuffling of the pairs (weight, clause), aiding in escaping local minima in most cases.

3.3 Maximization of $w \in \mathbb{W}$: Code Implementation

The subsequent code is a generalized Python implementation of the local search approach used in this article.

```
from random import choice, shuffle
def solve(problem, n=20, iterations=100):
  solution = [0 for k in range(n+1)]
  for it in range(iterations):
    shuffle(problem)
    score = [0]
    for pair in problem:
      if sum((abs(x) * solution[abs(x)] == x for x in pair[1])) > 0:
        score = [pair[0]] + score
      else:
        undefined = [x for x in pair[1] if solution[abs(x)] == 0]
        if len(undefined) > 0:
          x = choice(undefined)
          solution[abs(x)] = int(x / abs(x))
          score = [pair[0]] + score
        elif sum(score) / len(score) > pair[0]:
          score = [-pair[0]] + score
        else:
          x = choice(pair[1])
          solution[abs(x)] = 0
          score = [0] + score
  return solution
```

3.4 Maximization of $w \in \mathbb{W}$: Remarks

The provided code implements a local search-based algorithm for solving optimization problems. It takes a problem defined as a list of clauses (pairs of a score and a list of variables) and strives to discover a solution that maximizes the total score of satisfied clauses. Through a set number of iterations, the algorithm shuffles the problem clauses, makes decisions to enhance the solution based on clause satisfaction and associated scores, and thereby attempts to find better solutions, while acknowledging the potential of reaching local maxima. The "shuffle" step before each iteration is especially instrumental in preventing convergence to local minima.

Chapter 4

BOOLEAN DISSECTION OF SEARCH SPACE

This chapter delves into the exploration of Boolean variables within a set E taken from the domains $\mathbb{N}, \mathbb{R}, \mathbb{C}, \mathbb{R}^n$. A boolean variable is defined as a function $b : E \to \{0, 1\}$. We will differentiate the polarity of $x \in E$, denoted as $b : x \mapsto [x < 0]$. Another representation of a Boolean variable is linked to a power of two: $b : x \mapsto \lfloor x/2^k \rfloor \% 2$, where the $\cdot \% 2$ operation calculates the remainder modulo two, and $|\cdot|$ represents the absolute value.

4.1 Boolean Breakdown of $\mathbb{N}$

Any natural integer can be expressed through the following formula. By considering $x/2^k < 1$ for sufficiently large k, this series expression always results in a finite sum over $\mathbb{N}$.

$$I_{\mathbb{N}}(x) = x = \sum_{k=0}^{\infty} 2^k (\lfloor x/2^k \rfloor \% 2)$$

4.2 Boolean Breakdown of $\mathbb{R}$

The depiction of a real number is slightly distinct, given its potential for infinite floating point digits (e.g., π). The sign of the real number can be efficiently captured using the trick $1 - 2[x < 0] = (-1)^{[x<0]}$.

$$I_{\mathbb{R}}(x) = x = (1 - 2[x < 0]) \sum_{k \in \mathbb{Z}} 2^k (\lfloor |x|/2^k \rfloor \% 2)$$

4.3 Boolean Breakdown of $\mathbb{C}$

The complex plane $\mathbb{C}$ is approached by distinguishing its real value $\Re(z)$ and imaginary value $\Im(z)$, employing the formula from $\mathbb{R}$ simultaneously.

$$I_{\mathbb{C}}(z) = z = \sum_{k \in \mathbb{Z}} 2^k (1-2[\Re(z) < 0])(\lfloor |\Re(z)|/2^k \rfloor \% 2) + i 2^k (1-2[\Im(z) < 0])(\lfloor |\Im(z)|/2^k \rfloor \% 2)$$

4.4 Boolean Breakdown of $\mathbb{R}^n$

For a generalization to $\mathbb{R}^n$, where $n \geq 1$, variable differentiation is achieved through $(1_{i=j})_{1 \leq j \leq n} = (0, \ldots, 0, 1, 0, \ldots, 0)$, while the underlying principle remains unaltered.

$$I_{\mathbb{R}^n}(x_1, \ldots, x_n) = (x_1, \ldots, x_n) = \sum_{k \in \mathbb{Z}} 2^k \sum_{i=1}^{n} (1_{i=j})_{1 \leq j \leq n} (1-2[x_i < 0])(\lfloor |x_i|/2^k \rfloor \% 2)$$

Chapter 5

A UNIVERSAL EXPRESSION FOR $f : \mathbb{N} \to \mathbb{R}$

Within this chapter, we will establish the proposition that any function mapping natural integers to real numbers can be unfolded as a series involving weighted products of Boolean variables over $\mathbb{N}$.

5.1 An Assortment of Boolean Variables on $\mathbb{N}$

Consider an integer $l \geq 1$:

$$b_l : x \mapsto \lfloor x/2^{l-1} \rfloor \% 2$$

For all $x \in \mathbb{N}$, the following relation holds:

$$x = \sum_{l \in \mathbb{N}} 2^{l-1} b_l(x)$$

Moreover, for every $l \in \mathbb{N}$:

$$b_l(0) = 0$$

5.2 A Proposed Formula $g : \mathbb{N} \to \mathbb{R}$

$$g : x \mapsto \sum_{p \subset \mathbb{N}}^{|p|<\infty} a_p \prod_{l \in p} b_l(x)$$

Here, for all finite subset p of natural integers:

$$a_p = \sum_{q \subseteq p} (-1)^{|p|-|q|} f\left(\sum_{l \in q} 2^{l-1} \right)$$

5.3 A Key Outcome in Discrete Analysis

$$\forall x \in \mathbb{N}, g(x) = f(x)$$

The significance of this outcome lies in the fact that $g(0) = f(0)$ implies, through induction on $(a_p)_p$, that the binary decomposition of the function $x \mapsto f(x)$ is explicitly tied to the binary decomposition of $x \in \mathbb{N}$.

Chapter 6

APPROXIMATING $C(\mathbb{C}, \mathbb{R})$

This chapter delves into an approximation method for the function space $C(\mathbb{C}, \mathbb{R})$.

6.1 Boolean Variables of the Complex Plane

We assume your familiarity with $I_{\mathbb{C}}$, and we will now introduce Boolean variables for $\mathbb{C}$ as follows:

- For $l = 1$, $b_l(z) = [\Re(z) < 0]$
- For $l = 2$, $b_l(z) = [\Im(z) < 0]$
- For $l > 2$, and $l \equiv 3 \mod 4$, $b_l(z) = \lfloor |\Re(z)/2^{\frac{l-3}{4}}| \rfloor \% 2$
- For $l > 2$, and $l \equiv 0 \mod 4$, $b_l(z) = \lfloor |\Im(z)/2^{\frac{l-4}{4}}| \rfloor \% 2$
- For $l > 2$, and $l \equiv 1 \mod 4$, $b_l(z) = \lfloor |\Re(z)/2^{\frac{1-l}{4}}| \rfloor \% 2$
- For $l > 2$, and $l \equiv 2 \mod 4$, $b_l(z) = \lfloor |\Im(z)/2^{\frac{2-l}{4}}| \rfloor \% 2$

6.2 Dense Countable Subset F within $\mathbb{C}$

Consider the subset F defined as:

$$F = \{z \in \mathbb{C}, |\{l \in \mathbb{N} : b_l(z) = 1\}| < \infty\}$$

To make it countable, define z_k as the unique value that satisfies:

$$\sum_{l \in \mathbb{N}} 2^{l-1} b_l(z_k) = k$$

6.3 Random Variable X on F

Let X be a random variable over $\mathbb{N}$. We create $Z = z_X$, which becomes a new random variable over F. Notably, F is both countable and densely distributed across the complex plane, rendering Z a discrete random variable over the complex plane.

6.4 Upper Bound on Approximation Failure Probability

It can be demonstrated that regardless of the random variable Z on $Z(\Omega) \subset F \subset \mathbb{C}$, even for any $\varepsilon > 0$ representing the desired approximation precision, and any

$\delta > 0$ serving as an upper bound for the probability of approximation failure ($|f(Z) - g(Z)| > \varepsilon$), the following holds:

$$\forall \varepsilon, \delta > 0, \inf_{N \in \mathbb{N}} \Pr(|f(Z) - g(Z)| > \varepsilon) < \delta$$

Here, $g(z)$ is parametrized by N:

$$g(z) = \sum_{p \subset \mathbb{N}}^{\max p \leq N} a_p \prod_{l \in p} b_l(z)$$

Where for all finite subset p of natural integers:

$$a_p = \sum_{q \subseteq p} (-1)^{|p|-|q|} f\left(z_{\sum_{l \in q} 2^{l-1}}\right)$$

Intuitively, the proof's core idea is as follows:

$$N > \log_2(k) \Rightarrow g(z_k) = \sum_{p \subset \{l_1, \ldots, l_m\}} a_p = f(z_k)$$

Given $\sum_{i=1}^{m} 2^{l_i - 1} = k$, and utilizing the binomial coefficient property:

$$(1-1)^{|p|} = \sum_{k=0}^{|p|} \binom{|p|}{k} (-1)^k = 1_{p=\varnothing}$$

Once this equality holds for any $k \in \mathbb{N}$, one can isolate $G_\delta \subset F \subset \mathbb{C}$, a finite subset of F, as follows:

$$\Pr(Z \in G_\delta) > 1 - \delta$$

Choosing $N^* = \max_{z_k \in G_\delta} N_{z_k}$, where $f(z_k) = g(z_k)$ for all $z_k \in G_\delta$, the probability of approximation failure converges to zero, irrespective of the random variable X on $\mathbb{N}$, and the random variable $Z = z_X$ over $F \subset \mathbb{C}$.

Chapter 7

APPROXIMATING COMPLEX ROOTS OF $\mathbb{C}[X]$

7.1 Idea

To extend the boundaries of this theory, specifically in the context of finding roots[**5**], which essentially involves maximizing $z \mapsto -|P(z)|$ across the complex plane, we propose the following approach: assuming you have a complex root for a given polynomial P expressed in summation form, approximate the coefficients up to degree $\deg(P) - 1$ of the Taylor series of P at zero, divided by one of the output complex roots, z_0. This results in $\frac{1}{z-z_0}P(z) \approx c_0 + c_1 z + c_2 z^2 + c_3 z^3 + c_4 z^4$. By increasing the degree of the polynomial, it can be argued that with a sufficient N, as detailed in Chapter 3, the computation will avoid local maxima that are not global maxima (roots).

For the computation of c_k, you require $(f(l\epsilon))_{0\leq l\leq k}$ where $f(z) = \frac{1}{z-z_0}P(z)$ and $\epsilon = 0.001$ yields satisfactory outcomes. Subsequently, compute $f_1(l\epsilon) = \frac{f((l+1)\epsilon)-f(l\epsilon)}{\epsilon}$ to generate $(f_1(l\epsilon))_{0\leq l\leq k-1}$. This process can be iterated until $f_k(0) \approx k!c_k$.

7.2 Implementation

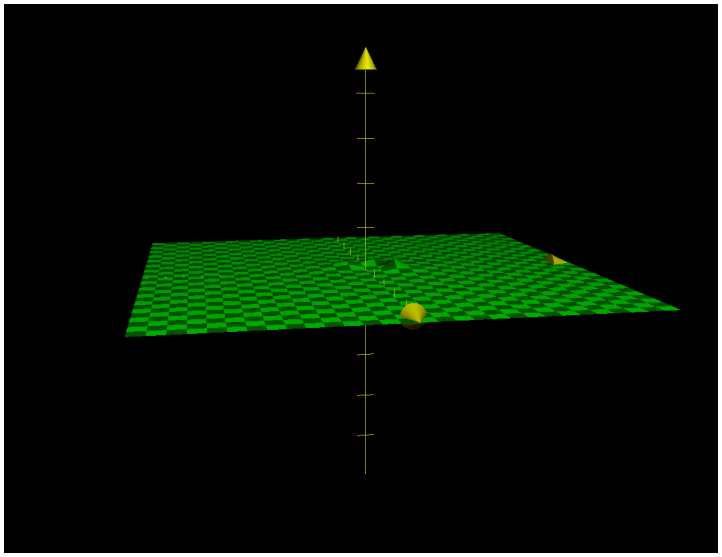
I have developed several Python prototypes as well as a C implementation of this algorithm based on Chapters 1-3. The source code is accessible on GitHub.

7.3 Experiment

The experiment involved computing the roots of $P(X) = X^5 + 1 + 2i + (3+4i)X + (5+6i)X^2 + (7+8i)X^3 + (9+10i)X^4$, utilizing the generic cost function $z \mapsto -|P(z)|$.

7.4 Results

Within a computational timeframe of 60 seconds, the algorithm produced a model $Q(X) \approx P(X)$ where $Q(X) = (X + 8.276 + i9.961)(X - 0.085 - i0.677)(X + 0.490 + i0.358)(X + 0.565 - i0.305)(X - 0.138 + i0.633)$. The effectiveness of the model was assessed through visualization.



$$z \mapsto |1 - P(z)/Q(z)|$$

7.5 Conclusion

Although the algorithm is generally effective, some challenges remain. Local search and gradient descent sometimes struggle with poles, where they may not achieve optimal performance. However, the approximation is nearly flat and close to $z \mapsto 0$ except for the poles. The use of local search on the cost function has enhanced algorithm performance. Given limited computational resources, the algorithm might not always avoid local maxima that aren't global maxima. The incorporation of successive derivatives at $z \approx 0$ yields a factorized polynomial $Q(X)$ that adequately represents the behavior of the $\sum$ form $P(X)$.

Chapter 8

EXPERIMENTING WITH THE CRYPTOGRAPHIC HASH FUNCTION SHA256

8.1 Idea

Given that the algorithm operates on polynomials, I decided to explore whether it was possible to maximize the variance of leading zeros in a batch of 1200 hashes using a cryptographic hash function[**7**] like SHA256[**6**]. The concept involved applying a mask to the binary representation of the hash in such a way that when the mask bit was set to 1 and the corresponding hash bit was also set to 1, all the bits to the right of that position would be flipped. The hashing process used a '1'-'0' string of 50 randomly-generated characters.

8.2 Implementation

I developed a Colab notebook and implemented a controlled experiment to compare different strategies and assess which one performed best. The aim was to determine if a strategy could indeed enhance the variance in the number of leading zeros across a representative batch of 1200 hashes.

8.3 Experiment

The analysis yielded a strategy represented as [0,-,-,-,-,-,+,+,0,0,0,0,0,0,0,0]. The experiment involved simulating this strategy (X_1), a strategy that applies the mask at random when zero (X_2), and several control strategies: random strategy (Y_1), all-zero strategy (Y_2), and all-one strategy (Y_3).

8.4 Results

After generating 50 batches of 1200 hashes each, the ratio $\mathbb{E}\left(\frac{2X}{X+Y}\right) \approx 1.02$ was obtained, where $X = X_1 + X_2$ represents the strategy and $Y = \frac{2(Y_1+Y_2+Y_3)}{3}$ denotes the control.

When considering the variance of leading zeros:

- 1 Golden Ticket $2X/(Y + X) \approx 1.0256280925848993$
- 2 Golden Ticket $2X/(Y + X) \approx 1.0446854793002325$

- 3 Golden Ticket $2X/(Y+X) \approx 1.0376863578667561$

When considering the maximum of leading zeros:

- 4 Curiosity $2X/(Y+X) \approx 0.9613259668508288$
- 5 Golden Ticket $2X/(Y+X) \approx 1.1475409836065573$
- 6 Golden Ticket $2X/(Y+X) \approx 1.054945054945055$
- 7 Curiosity $2X/(Y+X) \approx 0.9866666666666667$

8.5 Conclusion

Determining whether the results are significant is challenging, but if the 2% difference persists at a larger scale, it could potentially have financial implications. Notably, Bitcoin[**9**] mining relies on conditions like SHA256(SHA256(block + nonce)) $\leq 2^{256-D}$, where D is the difficulty. However, applying the experiment's strategy to Bitcoin mining[**8**] is not straightforward, as miners have limited control over the output of SHA256(block + nonce).

Chapter 9

EXPERIMENTING WITH ELLIPTIC CURVE CRYPTOGRAPHY

9.1 Idea

Elliptic Curve Cryptography (ECC)[**3**] involves finding integer solutions x and y that satisfy the equation $y^2 \equiv x^3 + 7 \mod p$, where p is a large prime. Currently, breaking ECC relies on brute-force attempts to find suitable (x, y) pairs in $\mathbb{Z}^2$.

9.2 Implementation

The same SHA256 algorithm used earlier was adapted to work with Elliptic Curve Cryptography.

9.3 Experiment

The goal was to find integer solutions (x, y) that satisfy the ECC equation, with a chosen prime $p = 123863$. The cost function to be maximized was defined as:

$$(x, y) \mapsto -\min((y^2 - x^3 - 7)\%p, p - (y^2 - x^3 - 7)\%p)$$

9.4 Results

During the experiment, a pair $(-207, 220)$ was discovered as a possible solution. This result was achieved after evaluating the cost function around 10^5 times and considering around 2^{20} candidate solutions.

9.5 Conclusion

While Elliptic Curve Cryptography is not immune to this algorithm, the prime number used in this experiment was relatively small (around 10^6). In practical ECC systems, prime numbers of the order of 256 bits (approximately 10^{77}) are commonly employed. Given that the algorithm's complexity is $O(\sqrt{2^n})$ for an n-bit solution space that encodes $(x, y) \in \mathbb{Z}^2$, a supercomputer might utilize this approach to exploit ECC, although ECC's strength typically lies in the use of large prime numbers for which brute-force attacks are infeasible[**7**].

Chapter 10

EXPLORING FACTORING THROUGH AN EXPERIMENTAL APPROACH

10.1 Idea

Factoring[**1**][**2**] involves finding two prime numbers p_1 and p_2 such that their product equals N, i.e., $N = p_1 \cdot p_2$.

10.2 Implementation

The same approach used in a previous Colab notebook was employed. A cost function was formulated, aiming to minimize the distance of N to the nearest multiple of p_1.

10.3 Experiment

The experiment was conducted on a relatively small composite number $N = 1488391 \approx 10^7$. The goal was to determine if prime factors could be extracted from the cost function:

$$x \mapsto -\min(N \% x, x - N \% x)$$

10.4 Results

Using a model with $4096 = 2^{12}$ candidates, and after 376 evaluations of the cost function, the prime factors 1217 and 1223 were successfully identified, validating the approach.

10.5 Conclusion

The experiment suggests that the factoring problem could be addressed using this algorithm. However, it's important to note that the prime number considered in this experiment was relatively small, around 10^7. In real-world factoring challenges, such as those in RSA, numbers of the order of 256 bits (approximately 10^{77}) are commonly used. The algorithm's complexity of $O(\sqrt{2^n})$ in a n-bit solution space implies that a supercomputer might exploit this approach, although, for larger numbers, dedicated factoring methods are more practical[**7**].

Chapter 11

CONCLUDING REMARKS AND FUTURE DIRECTIONS

11.1 Complexity and Model Refinement

The experiments described in this work have followed a consistent principle: generating random samples of a_p values, where p represents finite subsets of natural integers. By considering the sign of a_p and transforming $a_p \prod_{l \in p} b_l$ into positively-real-weighted clauses, the problem is converted into a maximization task in various domains like $\mathbb{N}$, $\mathbb{Z}^2$, $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{R}^n$. However, the sample's representativeness is crucial for the effectiveness of the approach. A larger sample, possibly involving larger $|p|$, is required to capture the essence of the maximization problem accurately.

11.2 Binomial Coefficients Connection

Exploring the relationship with binomial coefficients reveals that most values tend to lie within the range $\left[\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right]$, as suggested by the study of the binomial distribution with a success probability of $1/2$. By constraining $|p| < K$, the algorithm's complexity can be reduced to $O(2^K \binom{n}{K})$. However, truncating the instance may compromise representativeness and the attainment of global optima. The current work employed a uniform random sample of $\{0, 1\}^n$, with a maximum sample size of 1200 a_p values computed.

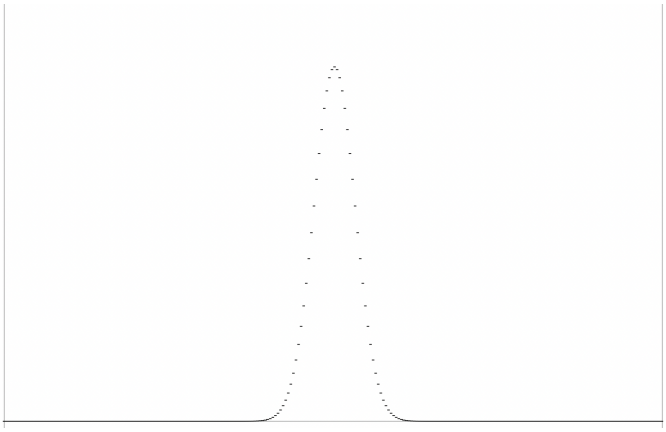
11.3 Future Prospects

As the author holds a position at Aplo, there is an intention to further develop and extend this research based on the promising outcomes. The next step involves conducting large-scale experiments on practical business problems, which can be quantified into suitable cost functions for maximization. This approach's potential for optimization and problem-solving warrants the exploration of more significant real-world challenges.

11.4 On the P vs NP Status

In 1971, Cook's theorem established the equivalence of a class of computing problems called NP-Complete. Any NP-Complete problem can be formalized as SAT or Boolean Satisfiability[**11**][**13**]. The theoretical complexity is $O(\lambda^n)$ with $\lambda \in (1, 2]$. Using the presented method in this article on the number of satisfied clauses for a

given solution x, one can use the proposed framework to drive down the complexity of maximizing this quantity. $P \neq NP$ would imply that there is no random subset of a_p with $|p| < K$ that leads to a global maximum of the quantity (solution to the instance). On the flip hand, $P = NP$ if no matter this quantity we can derive in polynomial time a $w \in \mathbb{W}$ a model of the cost function, that we can exploit to figure out a solution to the problem in polynomial time.



$$x \mapsto \binom{256}{\lfloor x \rfloor}$$

To efficiently tackle 256-bit encryption, setting $K = 64$ would avoid computing the center values that would take an exponential computation time. The main question that remains, is about the representativeness of leftmost elements $x < 64$ on the overall 2^{256} coefficients, centered at $x = 128$.

11.5 Relationship with Entropy

Let x^* be the unique solution to the cost function $f : x \mapsto 1_{x=x^*}$. The algorithm, will not be able to leverage the different values of the cost function to determine the solution. Such a cost function has an almost zero Entropy and a 2^{n-1} expected complexity under the assumption that x^* can be depicted by a $n-$bit solution vector. From a computational standpoint, it is analogous to a quick sort algorithm that performs in $O(n^3)$ in the worst-case scenario.

Appendix A

A RIGOROUS PROOF OF STATEMENT 1.

The primary objective of this appendix is to establish a rigorous proof for the following proposition:

$$\sup_{X\in\mathbb{X}} \inf_{w\in\mathbb{W}} \Pr(w(X) \neq f(X)) = 0$$

A.1 Constructing an Ideal Model on a Countable Dense Subset of $\mathbb{R}^n$

Let n be a natural number, and consider $x \in \mathbb{R}^n$ as a vector comprising n real numbers. The set $C(\mathbb{R}^n, \mathbb{R})$ denotes the collection of continuous functions. Define $\mathbb{W}$ as the set of weighted MAXSAT functions in the form $w = c + \sum w_i C_i$, where the Clauses C_i consist of the following Boolean variables on $\mathbb{R}^n$:

- $b_{i,k} : x \mapsto \lfloor |x \cdot (1_{i=j})_j| 2^k \rfloor \% 2$, where $i \in \mathbb{N} \cap [1, n]$ and $k \in \mathbb{Z}$
- $b_i : x \mapsto [x \cdot (1_{i=j})_j < 0]$, where $i \in \mathbb{N} \cap [1, n]$.

By introducing the set E, which encompasses any finite set of (i) and (i, k) indices of the Boolean variables, the formula below can be considered:

$$g : x \mapsto \sum_{p\in E} a_p \prod_{id\in p} b_{id}(x)$$

On the domain $\{x \in \mathbb{R}^n : |(i, k) : b_{i,k}(x) = 1| < \infty\}$, this formula consistently yields a finite sum that can precisely match the function's value. This is achievable through either exponential time construction of $2^{|p|}$ or estimation methods for quicker approximations. The central concept involves sampling $2M$ function values, negating M of them, and then summing these pairs. In the event of a constant function, the contributions cancel out, yielding $a_p = 0$.

$$x = \sum_{i=1}^{n} (1_{i=j})_j (-1)^{b_i(x)} \sum_{k\in\mathbb{Z}} \frac{b_{i,k}(x)}{2^k} \Rightarrow g(x) = f(x)$$

A.2 Deriving $\mu-$density

It is crucial to acknowledge that since the function f is continuous and perfectly matches g on a countable dense subset of $\mathbb{R}^n$, what can be inferred about g and its

continuity? Although it's reasonable to assume that g aligns exactly with the model, a rigorous proof is needed. As an alternative, a smaller property can be derived, indicating the $\mu-$density of the partial sum by imposing a condition $|k| < N$ in the indexes of Boolean variables of E_N.

$$g_N : x \mapsto \sum_{p \in E_N} a_p \prod_{id \in p} b_{id}$$

g_N serves as an estimator for f, guaranteeing that for any discrete random variable $X : \Omega \to \mathbb{R}^n$, a finite subset $G \subset X(\Omega)$ can be derived such that $\Pr(X \in G) > 1 - \varepsilon$. The claim is that due to the finite number of values required to achieve $X \in G$ with an approximation error of ε, coupled with constructing an exact model on a finite set of points using g_N (for sufficiently large N) and the continuity of f, a weak version of $\mu-$density is established:

$$\inf_N \Pr(|g_N(X) - f(X)| > \delta) = 0$$

Although not precisely $\mu-$density, the exact version can be expressed as:

$$\inf_N \Pr(g_N(X) \neq f(X)) = 0$$

This is the desired outcome.

A.3 The Function $\overline{f}$ with Respect to G

Given a fixed subset G and corresponding random variable X, where $X(\Omega)$ is constant and can't be changed, and we want the failure probability $w(X) \neq f(X)$ to be less than ε, define $d_G = \min\{\|x - y\| : x \neq y, (x, y) \in G\}$.

By selecting $N = |\log_2(d_G)| + |\log_2(\max\{\|x\| : x \in G\})|$, an approximate measure of the necessary precision to model the function emerges. Introduce $\overline{f} : x \mapsto f(proj_G(x))$, where $proj_G(x)$ represents the projection onto the finite set G of elements in $\mathbb{R}^n$.

$$\forall y \in G, \|x - proj_G(x)\| \leq \|x - y\|$$

For any $y \in G$ and $x \in B(y, d_G/2)$, $f(y) = \overline{f}(x) = \overline{g}(x) = \overline{g}(y)$.

It becomes evident that:

$$\overline{g} : x \mapsto \sum_{p \in E_N} a_p \prod_{id \in p} b_{id}(x) = c + \sum_{i=1}^{m} w_i C_i(x)$$

Consider:

$$\bigvee_{id \in p} \neg b_{id} + \bigwedge_{id \in p} b_{id} = \top$$

Hence, a model w can be deduced by analyzing the sign of a_p where (i) and (i, k) can be part of p, with $|k| < N$. This construction is achievable in a finite number of operations.

$$\forall \textit{(discrete)} X : \Omega \to \mathbb{R}^n, \forall \varepsilon > 0, \exists w \in \mathbb{W}, \Pr(w(X) \neq f(X)) \leq \varepsilon$$

In essence, for the set of discrete random variables on $\mathbb{R}^n$, Statement 1. always holds.

Appendix B

IMPLICATIONS OF $P \sim NP$

Consider a mathematical problem where a cost function can be explicitly formulated to maximize the solution. If this cost function possesses a significant amount of entropy, the article's framework can be applied to optimize it. While this might appear modest, it's significant, as an A* approximation for the global maximum can be achieved within a manageable number of operations.

So, where does the polynomial connection to NP problems arise? By examining the relationship between the Boolean variables' parameters of an NP function and its image, a real number, a model w can be derived and solved. Dynamic construction of w emerges as a preferable strategy.

The upcoming months hold significance for the field, potentially bringing changes to the cryptography landscape.

BIBLIOGRAPHY

[1] Hardy, G. H., & Wright, E. M. "An introduction to the theory of numbers." 2008. *Oxford University*

[2] Richard Crandall & Carl Pomerance. "Prime Numbers: A Computational Perspective." 2001. *Springer*.

[3] Koblitz, N. "Elliptic curve cryptosystems". 1987. *Mathematics of Computation*

[4] Murty, Katta G. "Linear programming" 2000. *John Wiley & Sons Inc.*

[5] VY Pan, AL Zheng, "New progress in real and complex polynomial root-finding" 2011. *Computers & Mathematics with Applications, 2011 - Elsevier*

[6] Penard, Wouter. Van Werkhoven, Tim. "On the Secure Hash Algorithm family" 2016. *staff.science.uu.nl*

[7] Stallings, William. "Cryptography and Network Security: Principles and Practice." 3 May 1990. *Prentice Hall. p. 165*

[8] Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008. *https://bitcoin.org/bitcoin.pdf*

[9] Heusser, Jonathan. "SAT solving - An alternative to brute force bitcoin mining" 2013. *https://jheusser.github.io/2013/02/03/satcoin.html*

[10] Marques-Silva, J., & Sakallah, K. (Eds.). "SAT 2009: 12th International Conference on Theory and Applications of Satisfiability Testing (Vol. 5584)". 2009. *Springer*.

[11] Biere, Armin (Ed.). *Handbook of Satisfiability*. IOS Press, 2009.

[12] H. Zhang and M. Stickel. "An efficient algorithm for unit-propagation" 1996. *In Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics.*

[13] Cook, Stephen. "The Complexity of Theorem-Proving Procedures." In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 1971. pp. 151-158.