Simulation-based approaches for understanding conjectures in cs-theory

Omar Madhani Sajin Koroth

This research was supported by the Valerie Kuehne Undergraduate Research Awards, University of Victoria. Supervised by the Department of Computer Science.
This research was enabled in part by software provided by Compute Canada.

Introduction

Algorithmic challenge: You are given a number n which is guaranteed to be the product of two distinct prime numbers p, q; can you find p and q?

If $n = 323$, what would p and q be? $p = 19,\ q = 17$

Now attempt this... If $n = 12997$, what would p and q be? $p = 317,\ q = 41$

You may have found p and q then, but what if n is:

$$1522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006 13$$

Why is this important?

It is believed to be hard to identify two distinct prime numbers from a large product. Your browser uses this idea to encrypt data on secure https web-pages (indicated by a 🔒 symbol in your search bar 💻), by sending a 2048-bit RSA public key (which is the composite number n) and the web-traffic would be encrypted using this key.

To decrypt the 2048-bit RSA public key without a private key would take trillions of years using the current best algorithms. What if someone comes up with a better algorithm? Is this problem actually hard?

The fundamental question in theory of computer science

Given an algorithmic problem, what is the time taken by the best algorithm solving the problem?

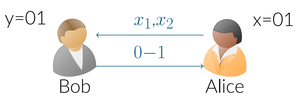
We believe that there are problems where no efficient algorithmic solutions exist. An algorithm is said to be "efficient" if its running time is polynomial in its input length. Most of our proofs showing the lower bounds of an algorithm's running time are algorithms themselves. However, using algorithms to prove limitations of other algorithms is naturally limited in its scope.

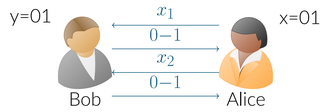
Given the hardness of this fundamental question, a natural approach is to severely limit the computational power of an algorithm and then study this question. An example of a restricted model is the model of decision trees. A decision tree can only query input bits one at a time. For example, given an n-bit binary string $z \in \{0,1\}^n$, to compute the sum, a c-program only needs a step whereas a decision tree would need n steps in which it queries $z_1, \ldots, z_n$. Given an algorithmic problem, it is very easy to answer whether a decision tree can compute it efficiently or not.

In this work, we study a line of research known as lifting theorems which "lift" lower bounds of decision trees to a significantly more powerful model of communication complexity. This model is comparable to an algorithm's running time.

Communication Complexity

The following 2-party Alice/Bob model of communication is a simple demonstration of communication complexity. We prioritize minimizing communication cost rather than run time. In this model, Alice and Bob are given n-bit binary strings and do not know which input the other has. Below are two protocols Alice and Bob could follow for checking whether their inputs are equal at every bit-position.





Research Methodology

In this work, we study a famous conjecture about "lifting" theorems. If true, it would make a lot of progress towards the fundamental question in the introduction and also have immediate applications in many areas of computer science. However, making progress towards proving these conjectures often requires new ideas and techniques.

Numerical simulations based approach

A fruitful approach, especially within theory-cs and mathematics in general, has been based on using numerical simulations to study these conjectures. However, designing simulations that verify these conjectures, even for inputs of length 10 bits, could take billions of years if done most straightforwardly. In this project, we will design and simulate efficient experiments which would lead to a better understanding of a fundamental conjecture related to lifting theorems. We use tools and techniques from optimization, algorithm design, and combinatorics to efficiently run the simulations.

Lifting theorems

Lifting theorems have played a key role in many recent breakthrough results in communication complexity. A typical lifting theorem translates the query complexity of a Boolean function f to the communication complexity of a function $f \circ g$ obtained by composing f with an inner function g. The inner function g is referred to as the gadget. Most lifting theorems work for any Boolean function f and depend upon the pseudo-random properties of the inner function g. The main parameter of efficiency in lifting theorems is the input size of the inner function g. Obtaining lifting theorems for constant-sized gadgets would give us breakthrough results and a nearly complete understanding of lifting phenomena; current techniques are far from achieving this goal.

Index Function

The index function is a well-known example of a gadget used in lifting theorems. Alice is given n pointers ($x = (x_1, \ldots, x_n)$-values). Each pointer/x_i-value ranges from 0 to $m-1$ and can be expressed in binary using $\log_2 m$ bits.

$$m = 2^k = \log_2 m = k \qquad x_i \in \{0,1\}^{\log_2 m} = \{0,1\}^k \quad (x_1, x_2, \ldots, x_n)$$

Bob is given n strings (y-values) of m-bit length.

$$y_i \in \{0,1\}^m \quad y_i = (y_{i1}, y_{i2}, \ldots, y_{im}) \quad (y_1, y_2, \ldots, y_n)$$

The output of the index gadget is an n-bit string z. Each z_i is determined by applying the index of x_i on y_i. There are 2^n possible z-values

$$z_i = (y_i)_{x_i} \quad Ind_m^n(x,y) = z \in \{0,1\}^n \qquad z = (z_1, z_2, \ldots, z_n)$$

Lovett-Meka-Mertz-Pitassi-Zhang Disperser Property Conjecture [2]

Statement of the conjecture: Let U be any subset of pointers (x-values) and V be any subset of n, m-bit strings (y-values). For a sufficiently large m, if $|U|$ and $|V|$ are large, meaning $|U| \geq m^n/2^{\Delta}$ and $|V| \geq 2^{mn}/2^{\Delta}$, all z-values are expected to occur as the result of applying Ind_m^n to inputs in $U \times V$.

We know that the conjecture is false when $m < \log_2 n$ and true when $m > n \log_2 n$. Our simulations attempt to find out whether the conjecture is true or false when $\log_2 n < m < n \log_2 n$

As all inputs of the Index gadget are initially hidden, all z-values occurring reveals no meaningful information about the inputs of f. The conjecture says that the only way to gain any meaningful information about z's is to increase deficiency (Δ) by communicating more. This is related to the pseudo-random property known as a "disperser" which is essential for proving lifting theorems and improving their gadget size. For example, reducing the size of $m \leq \text{poly} \log n$ for the Index gadget would already imply breakthrough results in two areas of theoretical-cs known as circuit complexity and proof complexity [1].

Our Results

Since we do not know whether the conjecture is true or false for the range $\log_2 n \leq m \leq n \log_2 n$, we designed two separate simulations.

Simulation #1:

The first simulation tries to verify the conjecture for the parameter range $\log_2 n \leq m \leq n \log_2 n$. Given a set of inputs U and V on which to test the conjecture, a straight forward simulation already takes time proportional to $|U| \times |V|$. $|U|$ could be as large as m^n and $|V|$ could be as large as 2^{mn}. Thus, even testing the conjecture for a given U, V alone can take an astronomical amount of time if m and n are large. Moreover, verifying a conjecture is an even harder challenge as it considers all possible set of inputs U, V which are "sufficiently" large. Naively running a simulation over all U, V, could take time proportional to 2^{m^n} and $2^{2^{mn}}$, thus making it impossible to straightforwardly verify the conjecture for even very small values of m and n. To overcome this issue, we decided to test the conjecture using randomly sampled large inputs U, V. This significantly improves the performance of our simulations. Understanding the conjecture using random inputs of smaller m values lead to an earlier work disproving the conjecture when $m < \log_2 n$. Thus, understanding the conjecture on random inputs U and V could lead to meaningful insights.

Simulation #2:

The second simulation tries to gain insight into whether the conjecture is true. We first simplify the conjecture to include only input sets V of $m \cdot n$-bit strings which are monotone. A set of binary strings, V, is said to be monotone if we can take any $y \in V$ and turn any number of 0's to 1 in y and still get an element in V. On inputs U,V where V is monotone, Ind_m^n produces all 2^n z-values if and only if it produces the all-zero string. The general conjecture is true only if the monotone version of the conjecture is true and the if the monotone version of the conjecture is false, so is the original. Due to the monotonicity of V, studying the $m \times n$-bit y input that does not produce the all zero z, with the least number of 1's in y, for a given U, can help us in proving the conjecture for $m \leq \text{poly} \log n$. Since we cannot iterate over all possible U's, we wrote a linear program that would find the minimal y for a randomly generated input set U. Simulation #2, is significantly faster than Simulation #1, as we do not have to iterate through all possible $y \in V$.

Experimental setup:

We used SageMath [3], an open-source mathematical modeling software, to design our simulations. We used the Compute Canada computing cluster resources to run our simulations. We used many ideas from design and analysis of algorithms to significantly improve the performance of our simulations. Some of the improvements came from using better data-structures (using a list to check if all z's exists), coming up with faster sub-routines (designing a faster random sampling algorithm for generating U and V), and using advanced tools like linear programming. Our simulations take a long time and we are in the initial exploratory phase of testing out the parameter range of $\log_2 n \leq m \leq n \log_2 n$ using small values of m and n.

References

[1] Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In *STOC*, pages 378–387. ACM, 2000.

[2] Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with sunflowers. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 104:1–104:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[3] William Stein. Sage mathematics software. *http://www. sagemath. org/*.

We acknowledge and respect the lək̓əŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

omarmadhani@uvic.ca