NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

Master's Thesis

Fine-Grained Complexity: Exploring Reductions and their Properties

Stavros I. Petsalakis

ATHENS

July 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΊΑ

Ραφιναρισμένη Πολυπλοκότητα: Εξερευνώντας Αναγωγές και τις Ιδιότητές τους

Σταύρος Ι. Πετσαλάκης

ΑΘΗΝΑ

Ιούλιος 2018

Master's Thesis

Fine-Grained Complexity: Exploring Reductions and their Properties

Stavros I. Petsalakis

SUPERVISOR: Aris Pagourtzis, Associate Professor

THREE-MEMBER EXAMINATION COMMITTEE:

Aris Pagourtzis, Associate Professor

Stathis Zachos, Emeritus Professor

Vassilis Zissimopoulos , Professor

Examination Date: July 20, 2018

Διπλωματική Εργασία Μεταπτυχιακών Σπουδών

Ραφιναρισμένη Πολυπλοκότητα: Εξερευνώντας Αναγωγές και τις Ιδιότητές τους

Σταύρος Ι. Πετσαλάκης

Ημερομηνία Εξέτασης: 20 Ιουλίου 2018

ABSTRACT

Algorithmic design has been one of the main subjects of interest for Computer science. While very effective in some areas, this approach has been met with some practical dead ends that have been very problematic in the progress of the field. Classical Computational Complexity practices have also not been able to bypass these blocks. Understanding the hardness of each problem is not trivial. Fine-Grained Complexity provides new perspectives on classic problems, resulting to solid links between famous conjectures in Complexity, and Algorithmic design. It serves as a tool to prove conditional lower bounds for problems with polynomial time complexity, a field that had seen very little progress until now. Popular conjectures such as SETH, k-OV, 3SUM, and APSP, imply many bounds that have yet to be proven using classic techniques, and provide a new understanding of the structure and entropy of problems in general. The aim of this thesis is to contribute towards solidifying the framework for reductions from each conjecture, and to explore the structural difference between the problems in each case.

SUBJECT AREA: Fine Grained Complexity

KEYWORDS: Reductions, Complexity, Hardness in P

ΠΕΡΙΛΗΨΗ

Η σχεδίαση αλγορίθμων αποτελεί ένα απο τα κύρια θέματα ενδιαφέροντος για τον τομέα της Πληροφορικής. Παρά τα πολλά αποτελέσματα σε ορισμένους τομείς, η προσέγγιση αυτή έχει πετύχει κάποια πρακτικά αδιέξοδα που έχουν αποδειχτεί προβληματικά στην πρόοδο του τομέα. Επίσης, οι κλασικές πρακτικές Υπολογιστικής Πολυπλοκότητας δεν ήταν σε θέση να παρακάμψουν αυτά τα εμπόδια. Η κατανόηση της δυσκολίας του κάθε προβλήματος δεν είναι τετριμμένη. Η Ραφιναρισμένη Πολυπλοκότητα παρέχει νέες προοπτικές για τα κλασικά προβλήματα, με αποτέλεσμα σταθερούς δεσμούς μεταξύ γνωστών εικασιών στην πολυπλοκότητα και την σχεδίαση αλγορίθμων. Χρησιμεύει επίσης ως εργαλείο για να αποδείξει τα υπο όρους κατώτατα όρια για προβλήματα πολυωνυμικής χρονικής πολυπλοκότητας, ένα πεδίο που έχει σημειώσει πολύ λίγη πρόοδο μέχρι τώρα. Οι δημοφιλείς υποθέσεις/παραδοχές όπως το SETH, το OVH, το 3SUM, και το APSP, δίνουν πολλά φράγματα που δεν έχουν ακόμα αποδειχθεί με κλασικές τεχνικές και παρέχουν μια νέα κατανόηση της δομής και της εντροπίας των προβλημάτων γενικά. Σκοπός αυτής της εργασίας είναι να συμβάλει στην εδραίωση του πλαισίου για αναγωγές από κάθε εικασία και να διερευνήσει την διαρθρωτική διαφορά μεταξύ των προβλημάτων σε κάθε περίπτωση.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ραφιναρισμένη Πολυπλοκότητα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αναγωγές, Πολυπλοκότητα

CONTENTS

List of Figures

1. INTRODUCTION

Theoretical computer science involves the study of computation for its own sake, irrespective of particular implementations. This field includes as sub-fields the theory of algorithms, i.e. the design and analysis of computational procedures, and complexity theory, which involves classification of computational tasks as well as efforts to prove that no efficient algorithms exist in certain cases. The three traditionally central areas of the theory of computation are automata, computability, and complexity and they are linked by the question of what are the fundamental capabilities and limitations of computers. This question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation.

The history of algorithms can be traced back to the ancient Babylonians as, for example, the standard algorithms for addition and multiplication taught in elementary schools are included in their inventions. The term ”algorithm” itself derives from the 9th Century Persian mathematician Muhammad ibn Musa al-Khwarizmi. In the present days the term ”algorithm” is central to the Theory of computation and ”algorithms form the basis of multi-billion dollar industries. Of the different algorithms which may be applicable to the same computational task, an important goal is to find the best one. Finding a faster algorithm can make a much bigger difference than better technology.

Throughout the years, Algorithmic design has made many leaps of progress in solving problems that were thought unsolvable. One of the main goals of such a field is to be constantly improving on the thought process involved in producing algorithms, understanding each problem’s inherent structural complexity and entropy, resulting in either faster algorithms for these problems, or proofs that said problems cannot be solved in a better way than it has already been solved. Specifically, when studying a problem *P*, we decide a computational model on which to study it (such as a Turing Machine or a WordRAM machine) and try to create an algorithm that solves problem *P* for input *n* in time $O(t(n))$ for a function *t*. We say that this problem is in complexity class *TIME(t(n))*.

The theories of computability and complexity are closely related. A central question in Computability Theory is how to classify problems as being solvable or unsolvable whereas in complexity theory, the objective is to classify problems as easy ones and hard ones.

Some basic concepts of **Computational Complexity theory** will be mentioned here.As noted, the central question in Complexity Theory is to classify problems according to their degree of ”difficulty” and secondly, to give a rigorous proof that problems that seem to be ”hard” are really ”hard”. Computer problems come in different varieties; some are easy, and some are hard. For example, the sorting problem is an easy one since even a small computer can sort a million numbers in ascending order rather quickly. However, a scheduling problem can be much harder than the sorting problem. For example, coming up with a schedule of classes for a university, which must satisfy some reasonable constraints, such as that no two classes take place in the same room at the same time, can be a very hard problem to solve, depending on the total number of classes.

As mentioned, Computational complexity theory focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other. The basic frame-work of computational complexity was developed in the late 60's and early 70's, and the years that followed witnessed a fast spread of its ideas to various branches of mathematics because this theory lead to an exact definition of the difficulty of a problem. **Complexity classes** are related to the rate of growth of the requirement in resources as the input n increases. For example, the class **NP** is the set of decision problems whose solutions can be determined by a non-deterministic Turing machine in polynomial time, while the class **PSPACE** is the set of decision problems that can be solved by a deterministic Turing machine in polynomial space.

The simplest complexity classes are defined by the type of computational problem, the model of computation, and the resource (or resources) that are being bounded and the bounds. Parameters that are used to define complexity classes include the model of Computation (Turing Machine, RAM, Circuits), the mode of Computation (Deterministic, Nondeterministic, Probabilistic) the complexity Measures (Time, Space, Circuit Size-Depth)and other Parameters (Randomization, Interaction). Automata Theory deals with definitions and properties of different types of the so called computation models, i.e. the definitions and properties of mathematical models of computation. Some examples of computational models include (a) Finite Automata which are used in text processing, compilers, and hardware design, (b) Context-Free Grammars which are used to define programming languages and in Artificial Intelligence, (c) Turing Machines, which form a simple abstract model of a computer.

A useful tool of complexity theory is the notion of **reducibility**. For a plethora of computational problems that arise in real-world applications, we still know little about their deterministic time or space complexity. And yet, even without such hard knowledge, it has been useful in practice to take some new problem A whose complexity needs to be analyzed, and announce that A has roughly the same complexity as a given problem B, by exhibiting efficient ways of reducing each problem to the other. Thus we can say a lot about problems being equivalent in complexity to each other, even if we cannot pinpoint what that complexity is. One reason this has succeeded is that, when one partitions the many thousands of real-world computational problems into equivalence classes according to the reducibility relation, there is a relatively small number of classes of this partition. Thus, the complexity of almost any problem arising in practice can be classified by showing that it is equivalent to one of a short list of representative problems which correspond to complexity classes. In other words, in Complexity Theory, we ”connect” problems in a complexity class with partial ordering relations, called reductions, which formalize the notion of ”a problem that is at least as hard as another.

A **Turing reduction** from a problem A to a problem B, is a reduction which solves A, assuming the solution to B is already known. There are many different types of reductions, based on the method of reduction, such as Cook reductions and Karp reductions.

Although there has been a lot of progress in these goals, there are many cases of problems in which, although studied thoroughly in many years of research, there has neither been an improvement on the running time of the corresponding fastest algorithm, nor has there been a proof that solidifies the current conjectured best algorithm to show that it is actually optimal. This implies that we do not understand the precise hardness of these problems, and we need to turn to new techniques to produce results.

In this manner, reducing problem A to problem B serves as an indicator that, since the solution for problem B can provide a solution for problem A, problem B is in a way harder to solve than problem A, so all structural difficulties of problem A are included in problem B. In classic Complexity Theory, these reductions were most commonly used in the case of NP-completeness and NP-hardness, to show that if there can exist a reduction of polynomial complexity linking said problem to a problem that is in complexity class NP, that consists of a proof that the problem is also in NP. Although this has proven to be very useful, the fact that we allow the reduction to abstractly use any polynomial as actual running time overhead limits these results to problems above polynomial complexity, because if the actual complexity of the best-known algorithm for said problem is less than the time used by the reduction, then the fact that the problem is in NP is a trivial and unimportant result. This train of thought has given rise to the field called Fine-Grained Complexity, which concerns itself with studying these problems by linking them with conjectures that are widely believed to be true. There is one main difference distinguishing Fine-Grained reductions from the classic ones used in NP-completeness. In the Fine-Grained case, we need the reduction to actually constitute a feasible algorithm for the initial problem. This is done by limiting the reduction to running times that are strictly lower than the conjectured best running time for this specific problem, while also restricting the amount of calls that can be made to problem B with respect to its own conjectured best running time, thus resulting in an algorithm that can actually be used to solve problem A within the conjectured best running time using problem B. This serves as a link between these two problems, as solving problem B in time less than its conjectured best running time would result in an improved algorithm for A, thus falsifying its conjectured lower running time bound. The most common practice is to use a problem that is widely believed to have been solved optimally in place of problem A, and reduce it to a specific problem of interest.

The conjectures that are mainly used for this purpose are the **Exponential Time Hypothesis (ETH)** and its stronger variant, the **Strong Exponential Time Hypothesis (SETH)**, which state that the Boolean Satisfiability problem (SAT) requires exponential time (2^n) to be solved, the **k-Orthogonal Vectors Conjecture**, which states that the k-Orthogonal Vectors problem requires n^k time to be solved, the **3SUM Conjecture** which states that the 3SUM problem cannot be solved in less than n^2 time, and the **All Pairs Shortest Paths (APSP) Conjecture**, which states that the APSP problem cannot be solved in less than n^3 time.

In this thesis, we will discuss about the fine-grained reductions that have been developed up to this point, while clarifying and mapping the landscape of the links they provide. We will also briefly discuss some ideas on my current approach and future ideas on the field.

2. PRELIMINARIES

2.1 Algorithms and Interesting Problems

In this section we will discuss some interesting problems and the best currently known algorithms to tackle said problems. We will also give some clarity as to the notation used when talking about the asymptotic complexity of some algorithms.

2.1.1 Asymptotic Notation (*O*, O^* and *Õ*)

Regarding the computational complexity of an algorithm, one is interested in the amount of resources used by the algorithm, as a non-decreasing function on the corresponding input size. That is, the time, space, amount of processors, communication, and randomness used by the algorithm. According to the interests of the analyst, one can discuss about *worst*, *average*, or *best* case complexity of an algorithm. In this thesis, we will discuss mainly about the worst case time complexity of algorithms, as this is more representative of the entropy contained in a problem.

When talking about the asymptotic estimation of the resources used by an algorithm, it is common practice to ignore values that are below the order of magnitude of the main resource cost. This notation is collectively called Bachmann–Landau notation or asymptotic notation. e.g. when talking about polynomial algorithms, an algorithm that runs in time $O(T(n))$ is any algorithm that runs in time less or equal to $cT(n)$ for some constant c.

Furthermore, to increase clarity, studies in the field generally use additional notation as such:

- $O(T(n))$ is used to ignore constant factors
- $\hat{O}(T(n))$ is used to ignore polyloglog factors
- $\tilde{O}(T(n))$ is used to ignore logarithmic factors
- $O^*(T(n))$ is used to ignore sub-exponential factors

The same notation is used to ignore the corresponding factors when talking about o,Ω, ω and Θ.

2.1.2 Basic Problems

For the purposes of this study we will need the reader to be familiar with the following basic problems.

2.1.2.1 Boolean Satisfiability Problem(SAT) and 3-SAT

Definition 1 *The Boolean Satisfiability Problem is the problem determining if, given a boolean formula, there exists an interpretation that satisfies it. That is, if the variables of the given formula can be evaluated to* TRUE *and* FALSE *in a way such that the whole formula evaluates to* TRUE*. If the answer is* Yes*, that formula is called* Satisfiable*.*

SAT is the first problem that was proven to be NP-COMPLETE (Cook-Levin). This means that all problems in the complexity class NP, are at most as hard as the SAT problem.

A special case of SAT called the 3CNFSAT or 3-SAT problem is of interest, as it has also been proven to be NP-complete, and is particularly useful as a starting point for proving that other problems are NP-hard.

This problem constricts the formulas to be in Conjunctive Normal Form, i.e. to be a conjunction of clauses, where each clause is a disjunction of literals. Additionally, each clause has to be consisting of exactly 3 literals.

2.1.2.2 k-Orthogonal Vectors (k-OV)

The Orthogonal Vectors problem is defined as follows:

Definition 2 *Given two sets of vectors $A, B \subseteq \{0,1\}^d$ where d is the dimension of the vectors and $|A| = |B| = n$, determine if there exist $a \in A$ and $b \in B$ such that $a \cdot b = 0$.*

The k-OV problem is the generalization of Orthogonal Vectors where given k sets of vectors $A_1, \ldots, A_k$ each of size n determine if there exist $a_1 \in A_1, \ldots, a_k \in A_k$ such that $a_1 \cdot \ldots \cdot a_k = \Sigma_{i=1}^{d} \Pi_{j=1}^{k} a_j[i] = 0$.

There is an obvious $N^2 d$ algorithm for $OV_{N,d}$ and as we will see in following chapters and in [87], it has been conditionally proved that there can be no algorithm for k-OV that runs significantly faster [1] than $N^2 d$.

2.1.2.3 3-SUM

Definition 3 *The k-SUM problem asks if a given set of numbers $\{C_1, \ldots, C_n\}$ contains k elements that sum to zero.*

The specific case for $k = 3$ was widely conjectured to be unsolvable in $\Omega(n^2)$ time. However, this conjecture has been refuted in 2014 by Allan Grønlund and Seth Pettie [65] who gave a deterministic algorithm that solves 3-SUM in $O(n^2/(logn/loglogn)^{2/3})$ time. This has been further improved, and the current best algorithm for 3SUM runs in

[1] in this context "significantly faster" is used to denote a factor greater than n^ϵ where $\epsilon = \Omega(1)$

$O(n^2/(logn/loglogn))$ time [47, 55]. However, it is still conjectured that 3SUM is unsolvable in time $O(n^{2-\Omega(1)})$, i.e. there can be no algorithm for 3SUM that runs significantly faster than n^2.

This conjecture was initially introduced by Gajentaan and Overmars [52, 53] who used this conjecture to prove conditional lower bounds of $O(n^2)$ for many problems in computational geometry.

2.1.2.4 All Pairs Shortest Paths(APSP)

The APSP problem is, given a graph G(V,E) with integer edge weights, to determine which are the shortest paths connecting each pair of vertices u and v in V. This problem is a very important and central problem in graph theory. The most studied algorithm for this problem is the Floyd-Warshall algorithm that solves this problem in $O(n^3)$ time (for a graph with n nodes). Another way to compute the all pair shortest path is by invoking Dijkstra's algorithm for each vertex. Dijkstra's algorithm finds the shortest path from a given vertex v to all other vertices in the graph. Hence, invoking it n times by choosing a different starting vertex v each time, will result in calculating the all pair shortest path. A single invocation of Dijkstra's algorithm takes $O(n+mlogm)$. Hence, n iterations of this algorithm takes a time of $O(nm+n^2logm)$. In the worst case,m can be $O(n^2)$ and hence the worst case running time using this approach is also $O(n^3)$.

There have been many improvements to the complexity of the problem, with the biggest improvement made by Williams [88] who achieved an algorithm that has a running time of $n^3/exp(\Theta(\sqrt{logn}))$. However, there has been no algorithm for APSP that runs significantly faster than n^3.

Some Related Measures that have been thoroughly studied are:

- the radius R of a graph: Radius of the graph is the smallest possible value for $\alpha = max_v(\alpha_v)$ over all vertices in the graph, where α_v is the maximum value of the shortest distance from a vertex v to any other vertex.
- the center of graph: The vertex v which minimizes the value α is called the center of the graph.
- the diameter of the graph: It is the maximum possible distance between any two vertices in the graph.
- the median of the graph: It is the node v_i that has the minimum value of the sum of distances to all vertices v in the graph.
- the betweeness centrality: It is very closely related to the number of s-t shortest paths, passing through a given vertex v.
- the positive betweeness centrality: It is an indicator function for a vertex v. If there exits a s, t shortest path through v, then the value is 1, else it is 0.

- the Negative Triangle: Given an undirected graph G with integer edge weights in $\{-M, ..., M\}$, the problem is to check if there exists a triangle in the graph, such that the sum of weights on this triangle is negative.

Radius, Diameter, Median, generally referred to as Centrality Measures, appear in a variety of applications including social networks, transportation and allocation problems, biological networks, etc. with real life implications. In this manner the question on the existence of subcubic algorithms is significant.

2.1.2.5 Edit Distance

Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. It is very useful in different fields such as linguistics and bioinformatics. For example, in natural language processing involving the interactions between computers and human (natural) languages, automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T, as well as protein analysis.

A more formal definition of the Edit Distance problem is, given two strings s and r, to determine the minimum-weight series of operations that are required to transform string s to string r. Most of the times in literature, the term 'edit distance' is used to denote Levenshtein distance, for which the operations include deletions, substitutions, and insertions of characters. In Levenshtein's initial definition, each of these operations have unit cost, so the distance is equal to the minimum *number* of operations required to transform s to r. Another definition associates these operations with non-negative costs $w_{ins}(x), w_{del}(x), w_{sub}(x, y), x \in \Sigma$.

2.1.2.6 Boolean Matrix Multiplication

The Boolean matrix multiplication (BMM) problem consists of multiplying two matrices that have as entries either 0 or 1.

BMM is one of the most fundamental problems in computer science. It has applications to triangle detection, transitive closure, context-free grammar parsing, etc. One way to multiply two Boolean matrices is to treat them as integer matrices, and apply a fast matrix multiplication algorithm over the integers. Matrix multiplication can be done in "truly subcubic time", i.e., the product of two $n \times n$ matrices can be computed in $O(n^3 - c)$ additions and multiplications. For example, the latest generation of such algorithms run in $O(n^2.373)$ operations.These algorithms are "algebraic", as they rely on the structure of the field, and in general the ring structure of matrices over the field. There is a different group of BMM algorithms, often called ”combinatorial” algorithms. They usually reduce the redundancy in the computation by exploiting some combinatorial structure in the Boolean matrices. Although these combinatorial algorithms have worse running times than the algebraic ones, they generally have some nice properties. Combinatorial algorithms usually can be generalized in ways that the algebraic ones cannot be. Moreover, in practice, these combinatorial algorithms are usually fast and easy to implement, while in contrast, most theoretically fast matrix multiplication algorithms are impractical to implement.

2.2 Computational Complexity

2.2.1 Computational Models

A **computational model** is a model which describes how a set of outputs are computed given a set of inputs. There are many computational models (RAM, Turing Machines etc). We can restrict attention to a single abstract computational model for studying computational problems - the Turing machine. The reason is that the Turing Machine seems able to simulate all physically realizable computational models with very little loss of efficiency. The Church-Turing Thesis states that all models are computationally equivalent, that is, every computation model can be simulated by a Turing Machine. These are regarding the computability of problems, i.e. questioning if a computational model can solve a specific problem or not, regardless of the running time.

However, when analyzing the complexity of a problem or class of problems, some computational models are faster than others, so clarifications need to be made in each case so as to not confuse the reader. In this thesis, for the most part, the word-RAM model is used, which is a Random Access Model that can perform tasks on words of predefined length l in O(1) time. In our case, for ease of computation we set the word length to be $l = O(logn)$.

2.2.2 Hierarchy Theorems

In computational complexity theory, the time hierarchy theorems are important statements about time-bounded computation on Turing machines. Informally, these theorems say that given more time, a Turing machine can solve more problems. For example, there are problems that can be solved with n^2 time but not n time.

The time hierarchy theorem for deterministic multi-tape Turing machines was first proven by Richard E. Stearns and Juris Hartmanis in 1965.It was improved a year later when F. C. Hennie and Richard E. Stearns improved the efficiency of the Universal Turing machine. Consequent to the theorem, for every deterministic time-bounded complexity class, there is a strictly larger time-bounded complexity class, and so the time-bounded hierarchy of complexity classes does not completely collapse. More precisely, the time hierarchy theorem for deterministic Turing machines states that for all time-constructible functions f(n):

$$\text{DTIME } (o(\frac{(f(n)}{logf(n)}) \subsetneq DTIME(f(n))$$

The time hierarchy theorem for nondeterministic Turing machines was originally proven by Stephen Cook in 1972. It was improved to its current form via a complex proof by Joel Seiferas, Michael Fischer, and Albert Meyer in 1978.Finally in 1983, Stanislav Žák achieved the same result with the simple proof taught today. The time hierarchy theorem for nondeterministic Turing machines states that if g(n) is a time-constructible function, and $f(n+1) = o(g(n))$, then

$$\text{NTIME}(f(n)) \subsetneq NTIME(g(n))$$

2.2.3 Complexity Classes

A complexity class is a set of problems of related resource-based complexity. A typical complexity class has a definition of the form: the set of problems that can be solved by an abstract machine M using O(f(n)) of resource R, where n is the size of the input. A complexity class contains a set of problems that take a similar range of space and time to solve. In order to classify a problem, it is usually proven to be in a particular complexity class by running the problem on an abstract computational model, usually a Turing machine. The relations and the possible equivalence between different complexity classes is an open question.

The following are fundamental time classes and fundamental space classes, given functions t(n) and s(n):

- DTIME[t(n)] is the class of languages decided by deterministic Turing machines of time complexity t(n).
- NTIME[t(n)] is the class of languages decided by nondeterministic Turing machines of time complexity t(n).
- DSPACE[s(n)] is the class of languages decided by deterministic Turing machines of space complexity s(n).
- NSPACE[s(n)] is the class of languages decided by nondeterministic Turing machines of space complexity s(n).

Canonical complexity classes:

- L=DSPACE[logn] (deterministic log space)
- NL=NSPACE[logn] (nondeterministic log space)
- P=DTIME[$n^{O(1)}] = \cup k \geq 1DTIME[n^k$] (polynomial time)
- NP=NTIME[$n^{O(1)}] = \cup k \geq 1NTIME[n^k$] (nondeterministic polynomial time)

- PSPACE=DSPACE[$n^{O(1)}] = \cup k \geq 1DSPACE[n^k$] (polynomial space)
- E=DTIME[$2^{O(n)}] = \cup k \geq 1DTIME[k^n$]
- NE=NTIME[$2^{O(n)}] = \cup k \geq 1NTIME[k^n$]
- EXP=DTIME[$2^{n^{O(1)}}] = \cup k \geq 1DTIME[2^{n^k}$] (deterministic exponential time)
- NEXP=NTIME[$2^{n^{O(1)}}] = \cup k \geq 1NTIME[2^{n^k}$] (nondeterministic exponential time)
- EXPSPACE=DSPACE[$2^{n^{O(1)}}] = \cup k \geq 1DSPCE[2^{n^k}$] (exponential space)

The relations that are known until now are:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq NPSPACE \subseteq EXP \subseteq NEXP$$

2.2.4 ETH, SETH and more

The theory of NP-hardness gives us strong evidence that certain fundamental combinatorial problems, such as 3-SAT, are unlikely to be polynomial-time solvable. However, NP-hardness does not give us any information on what kind of super-polynomial running time is possible for NP-hard problems. Impagliazzo, Paturi, and Zane [62] introduced the Exponential Time Hypothesis (ETH) and the stronger variant, the Strong Exponential Time Hypothesis (SETH), which state lower bounds on how fast satisfiability problems can be solved. These assumptions can be used as a basis for qualitative lower bounds for other concrete computational problems.

ETH states that the 3-SAT problem cannot be solved in subexponential time. That is, there is an $\epsilon > 0$ such that 3-SAT cannot be solved in time $O(2^{\epsilon n})$.

The **Strong Exponential Time Hypothesis** states that the Boolean Satisfiability Problem (SAT) requires exponential time. Formally, **SETH** states that for each $\delta < 1$ there exists a $k \geq 3$ such that kSAT requires $2^{\delta n}$ time.

This conjecture implies that $P \neq NP$ but is actually stronger than this historic argument. Specifically, even if SETH is falsified the matter of P vs NP is still in question. While ETH is generally considered a plausible complexity assumption, SETH is regarded by many as a quite doubtful working hypothesis that can be refuted any time. For this reason, lower bounds proven under the assumption of SETH should not be regarded as supported by very strong arguments, but rather that existence of better algorithms would constitute a major breakthrough in the complexity of satisfiability.

2.2.4.1 Sparsification Lemma

An important tool in this area is the Sparsification lemma defined in [62], which shows that, for any $\epsilon > 0$, any k-CNF formula can be replaced by $O(2^{\epsilon n})$ simpler k-CNF formulas in which each variable appears only a constant number of times, and therefore in which the number of clauses is linear. The Sparsification Lemma essentially says that an arbitrary k-CNF can be expressed (in subexponential time) as the disjunction of a subexponential number of linear size k-CNFs. More precisely, the Sparsification Lemma shows that for all $\epsilon > 0$, k-CNF F can be written as the disjunction of at most $2^{\epsilon n}$ k-CNF F_i such that F_i contains each variable in at most c(k,e) clauses, for some function c. Moreover, this representation can be computed in $O(poly(n)2^{\epsilon n})$ time.

The sparsification lemma is proven by repeatedly finding large sets of clauses that have a nonempty common intersection in a given formula, and replacing the formula by two simpler formulas, one of which has each of these clauses replaced by their common intersection and the other of which has the intersection removed from each clause.

Theorem 1 *Sparsification Lemma [IPZ01]*

$\exists$ algorithm A $\forall k \geq 2$, $\epsilon \in (0, 1]$ and k-CNF formula ϕ where $A_{k,\epsilon}(\phi)$ outputs CNF formulas $\phi_1, \ldots, \phi_s$ in $2^{\epsilon n}$ time, such that :

- *$s \leq 2^{\epsilon n}$; **SOL**$(\phi) = \cup_i$**SOL**(ϕ_i) where $Sol(\alpha)$ denotes the set of solutions of formula α*
- *$\forall i \in [s]$ each literal occurs $\leq O(\frac{k}{\epsilon})^{3k}$ in ϕ_i*

2.2.4.2 SETH $\rightarrow$ ETH

Showing that SETH implies ETH isn't trivial. The idea is to show that k-SAT reduces to sparse k-SAT, which is k-SAT restricted to instances with $O(n)$ clauses.

By applying the sparsification lemma and then using new variables to split the clauses, one may then obtain a set of $O(2^{\epsilon n})$ 3-CNF formulas, each with a linear number of variables, such that the original k-CNF formula is satisfiable if and only if at least one of these 3-CNF formulas is satisfiable. Therefore, if 3-SAT could be solved in subexponential time, one could use this reduction to solve k-SAT in subexponential time as well.

2.2.5 Reductions

Definition 4 *A **Cook reduction** is a polynomial-time Turing reduction from a problem A to a problem B is an algorithm that solves problem A using a polynomial number of calls to a subroutine for problem B.*

Definition 5 *A **Karp reduction** is a polynomial-time many-one reduction from a problem A to a problem B (both of which are usually required to be decision problems). It is a polynomial-time algorithm for transforming inputs to problem A into inputs to problem B, such that the transformed problem has the same output as the original problem.*

With many-one reductions, the solution for B can be invoked only once at the end, and the answer cannot be modified. This means that if we want to show that problem A can be reduced to problem B, we can use our solution for B only once in our solution for A, unlike in Cook reduction, where we can use our solutions for B as many times as needed while solving A.

3. INTRODUCING FINE-GRAINED REDUCTIONS

The basic technique used in fine-grained complexity is to use conjectures about the lower running time bounds of problems that are widely believed to be true, to produce running time lower bounds for other less studied problems. A fine-grained reduction from a problem *A* associated with a widely believed conjecture to another problem *B* links the plausibility of the bound for problem A actually being the correct, with the respective bound of problem B. This results in a conditional proof of a lower bound i.e. as long as the conjecture holds, problem *B* has the respective running time lower bound.

3.1 Basic Conjectures

In this section we will introduce four main conjectures that have been used in fine-grained reductions to prove conditional running time lower bounds for various problems.

3.1.1 SETH, k-OV

The **Strong Exponential Time Hypothesis (SETH)** states that the Boolean Satisfiability Problem (SAT) requires exponential time. Formally, **SETH** states:

Theorem 2 *for each $\delta < 1$ there exists a $k \geq 3$ such that kSAT requires $2^{\delta n}$ time.*

This conjecture implies that $P \neq NP$ but is actually stronger than this historic argument. Specifically, even if SETH is falsified the matter of P vs NP is still in question.

Theorem 3 *The **k-OV conjecture** states that no algorithm gan solve the k-Orthogonal Vectors problem on instances of size n in $n^{k-\epsilon}poly(d)$ time for any constant $\epsilon > 0$.*

It has been shown by [87] that if the k-OV conjecture is falsified then SETH also falls, resulting in many interesting implications that seem implausible to the bigger part of the scientific community such as lower bounds for in circuit complexity, which have never been shown via different methods.

These two conjectures have been the main focus of fine-grained reductions and have been linked to most of the problems discussed in literature.

3.1.2 3-SUM

The **3-SUM Conjecture** states that the 3SUM problem cannot be solved in time $n^{2-\epsilon}$ for any $\epsilon > 0$. In fact, the 3SUM problem has been very much studied because of its significance in Computation Geometry, and there have been many improvements in various specific cases using many structural, algebraic and number theoretic properties. Nevertheless, in the general case the conjecture has not been refuted despite the efforts.

3.1.3 APSP

The **APSP Conjecture** states that the All Pairs Shortest Paths problem on graphs with n nodes with no negative cycles cannot be solved in time $n^{3-\epsilon}$ for any $\epsilon > 0$.

This problem has been key to developing many algorithms for graphs and the Floyd-Warshall algorithm (which is the current best-known algorithm for the generic version of the problem) is one of the more important examples of dynamic programming and its applications.

3.2 Fine-Grained Reductions

In this section we will discuss fine-grained reductions, as defined by V. Williams in [90] and we will explore the more technical aspects of said reductions.

Definition 6 *Assume that A and B are computational problems and a(n) and b(n) are their conjectured running time lower bounds, respectively. Then we say A (a,b)-reduces to B, $A \leq_{a,b} B$, if for every $\epsilon > 0$, there exists $\delta > 0$, and an algorithm R for A that runs in time $a(n)^{1-\delta}$ on inputs of length n, making q calls to an oracle for B with query lengths $n_1, ..., n_q$ where $\sum_{i=1}^{q}(b(n_i))^{1-\epsilon} \leq (a(n))^{1-\delta}$*

If $A \leq_{a,b} B$ and $B \leq_{b,a} A$, we say that A and B are fine-grained equivalent, $A \equiv_{a,b} B$.

In other words, $A \leq_{a,b} B$ means that if B is solvable in $b(n)^{1-\epsilon}$ then A is solvable in $a(n)^{1-\epsilon}$

3.2.1 Some Important Reductions

3.2.1.1 k-SAT $\leq$ k-OV [87]

Theorem 4 *If k-OV on sets with N vectors from $\{0,1\}^m$ can be solved in $N^{k-\epsilon}poly(m)$ for any $\epsilon > 0$ then CNF-SAT on n variables and m clauses can be solved in $2^{n-\epsilon'}poly(m)$ time for some $\epsilon' > 0$ and SETH is false.*

Proof We split the n variables of the CNF formula F into k sets $V_1, ...V_k$, each with n/k variables. We create sets $A_1, ..., A_k$ each having $|A_i| = N = 2^{n/k}$ vectors of length m, corresponding to all possible boolean assignments ϕ to the variables in V_i. The vectors $a^i(\phi)$ have values determined as such:

$a_i(\phi)[j] = 0$ if ϕ satisfies the jth clause of the CNF formula F, and 1 otherwise

One can now see that, if for some $a_1(\phi_1) \in A_1, ..., a_k(\phi_k) \in A_k$ we have $a_1(\phi_1) \cdot \ldots \cdot a_k(\phi_k) = 0$ then for every clause c, there is some vector $a_i(\phi_i)$ that is 0 on the cth dimension, and hence the corresponding assignment ϕ_i satisfies clause c. Thus, if we concatenate the assignments $\phi_1, ..., \phi_k$ we get an assignment that satisfies all clauses of formula F.

Note that this reduction respects the resources available only in the case that the formula of the k-SAT instance has number of clauses $m = O(n)$. However, this does not affect the proof as we can always produce such instances from any formula, using the **Sparsification Lemma**. If k-OV with set size $N = 2^{n/k}$ and vectors in $\{0,1\}^m$ can be solved in $N^{k-\epsilon}poly(m)$ time, then CNF-SAT can be solved in $(2^{n/k})^{k-\epsilon}poly(m) = 2^{n-\epsilon'}$ for $\epsilon' = \epsilon/k > 0$ contradicting **SETH**.

3.2.1.2 OV $\leq$ Graph Diameter [81, 35]

Definition 7 *A graph's diameter is the largest number of vertices which must be traversed in order to travel from one vertex to another. In other words, the length of the Maximum Shortest Path in a graph.*

Theorem 5 *If one can distinguish between Diameter 2 and 3 in an undirected unweighted graph with O(N) nodes and edges in $O(N^{2-\epsilon})$ time for some $\epsilon > 0$, then 2-OV on two sets of n vectors in d dimensions can be solved in $n^{2-\epsilon}poly(d)$ time and SETH is false.*

Proof: Given an instance of 2-OV, $|A| = |B| = n$ vectors in $\{0,1\}^d$. We create the following Graph G: For every vector $a \in A$, create a node a of G. For every vector $b \in B$, create a node b of G. For every $i \in [d]$, create a node c_i. We add two additional nodes x and y.

We add edges as follows. For every $a \in A$ and $i \in [d]$ if $a[i] = 1$, add an edge between a and c_i. Similarly for every $b \in B$.
Now we add edges (x,a) for every $a \in A$, (x,i) for every $i \in [d]$ and (x,y).
We also add edges (y,b) for every $b \in B$, (y,i) for every $i \in [d]$.

Now, for each $a \in A, b \in B$, if a and b are not orthogonal, there exists an $i \in [d]$ such that $a[i] = b[i] = 1$, and so the distance between a and b is 2 (via c_i).
If there exist $a \in A, b \in B$ so that $a \cdot b = 0$, the distance of the respective vertices is ≥ 3 so the diameter of the Graph is 3.
Let N = nd. The number of nodes and edges is at most O(N). If Diameter 2 vs 3 can be solved in $O(N^{2-\epsilon})$ time for some $\epsilon > 0$, then 2-OV is in $O((nd)^{2-\epsilon}poly(d))$ time for $\epsilon > 0$.

3.2.1.3 3-SUM $\equiv$ GeomBase [52, 53]

Definition 8 *Problem: GeomBase Given a set of n points with integer coordinates on three horizontal lines $y = 0, y = 1, y = 2$, determine whether there exists a non-horizontal line containing three of the points.*

Theorem 6 *If GeomBase can be computed in time $O(n^{2-\delta})$ for some $\delta > 0$ on two sequences of length n, then the 3SUM problem can also be solved in time $n^{2-\delta}$*

Proof: We start by showing the equivalence of 3SUM with the 3SUM' problem 3SUM' problem: Given three sets of integers A,B, and C of total size n, determine if there are $a \in A, b \in B$ and $c \in C$ such that $a + b = c$.

3SUM $\equiv$ *3SUM'*

- $\leq$ Set $A = S, B = S, C = -S$. Now obviously when for $a \in A, b \in B, c \in C, a + b = c$ then $a, b, (-c) \in S$ and $a + b + (-c) = 0$.
- $\geq$ w.l.o.g. assume all elements in the sets are positive. Set $m = 2max(A, B, C)$. Construct S as follows: for each element $a \in A$ put $a' = a + m$ in S. For each element $b \in B$ put $b' = b$ in S, and for each element $c \in C$ put $c' = -c - m$ in S. Clearly, if $a + b = c$ then $a' + b' + c' = 0$.

3SUM' $\equiv$ *GeomBase*

- $\geq$ For each element $a \in A$ create point (a,0). For each element $b \in B$ create a point (b,2), and for each element $c \in C$ create a point (c/2,1). It follows that three points (a,0),(b,2) and (c/2,1) are colinear iff $a + b = c$.
- $\leq$ Similarly, for each point (a,0) create an element $a \in A$, for each point (b,2) create an element $b \in B$, and for each point (c,1) create an element $2c \in C$.

Note that all the reductions used require only linear overhead, which is negligible when compared to the quadratic complexity of each (currently best known) algorithm. This proof concludes that if an algorithm is created that can solve either of these problems in time $n^{2-\epsilon}$ then these reductions can be used to provide subquardatic algorithms for the remaining problems.

3.2.1.4 OV $\leq$ Edit Distance [16]

Theorem 7 *Edit distance cannot be computed in $O(n^{2-\epsilon})$ unless SETH is false*

The intuition for this proof can be described as such: They use the gadgets provided below, to force the solution into cases that compare specific parts of the strings with each other. They align the patterns in a way that simplifies the problem into distinguishing two cases from each other, producing a specific cost if there exist orthogonal vectors, in the starting instance of OV, and another distinct one if there do not.

Proof:
Observation 1: For any two sequences x,y, EDIT(x,y) is equal to the minimum, over all

sequences z, of the number of deletions and substitutions needed to transform x into z and y into z. (no need for insertions)

Definition 1, Pattern Matching Distance:

$$Pattern(P_1, P_2) = minEDIT(P_1, x) \text{ where x is a contiguous subsequence of P2}$$

Simplifying assumption:Assume all vectors $b \in B$ have $b_1 = 1$. (w.l.o.g. add 1 to every b and 0 to every a).

- $l_0 = 1000d$,
- $l_1 = (1000d)^2$,
- $l = d(4 + 2l_0)$

$$CG_1(x) := \left\{ \begin{array}{l} 2^{l_0}01112^{l_0} \text{ if } x = 0 \\ 2^{l_0}00012^{l_0} \text{ if } x = 1 \end{array} \right\} CG_2(x) := \left\{ \begin{array}{l} 2^{l_0}00112^{l_0} \text{ if } x = 0 \\ 2^{l_0}11112^{l_0} \text{ if } x = 1 \end{array} \right\}$$

$$EDIT(CG1(x_1), CG_2(x_2)) = \left\{ \begin{array}{l} 1 \text{ if } x_1 \cdot x_2 = 0 \\ 3 \text{ if } x_1 \cdot x_2 = 1 \end{array} \right\}$$

For vectors $a, a', b \in \{0,1\}^d$ we define Vector Gadgets:

$$VG_1(a, a') = Z_1 L(a) V_0 R(a') Z_2 \text{ and } VG_2(b) = V_1 D(b) V_2$$

where

$$V_1 = V_2 = V_0 = 3^{l_1}, Z_1 = Z_2 = 4^{l_1}$$

$$L(a) = \bigcirc_{i \in [d]} CG_1(a_i),\ R(a') = \bigcirc_{i \in [d]} CG_1(a'_i),\ D(b) = \bigcirc_{i \in [d]} CG_2(b_i)$$

Note: $\bigcirc$ is used to denote concatenation

There are only two ways to achieve small edit distance cost between VG_1 and VG_2.

- Case 1: Delete Z_1, L and substitute Z_2 with V_2. This will result in a cost $C' := l_1 + l + l_1$ plus the cost of transforming R to D. By the construction of the coordinate gadgets, the cost of transforming R to D is $d + 2(a' \cdot b)$. Therefore, this case corresponds to ED cost $C' + d + 2(a' \cdot b) = C + 2(a' \cdot B)$.
- Case 2: Delete R,Z_2 and substitute Z_1 with V_1. This will result in a total cost of $C' + d + 2(a \cdot b) = C + 2(a \cdot b)$.

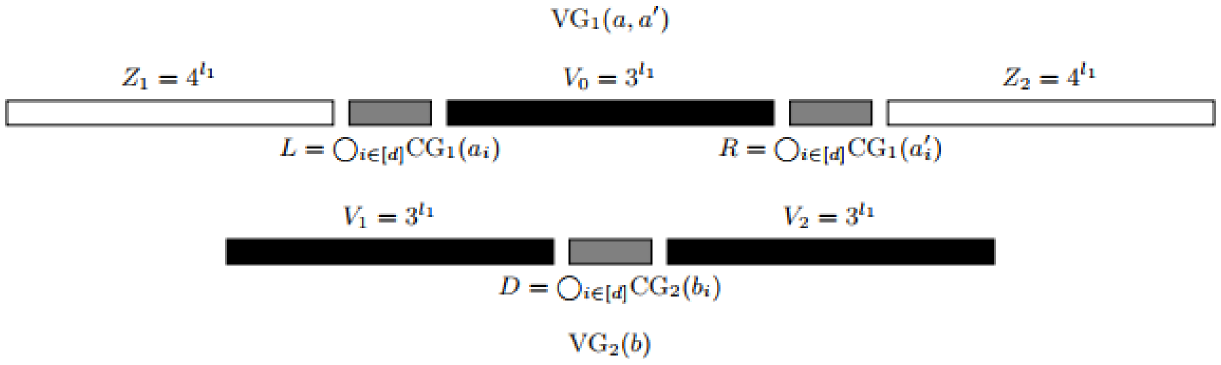

$\mathrm{VG}_1(a, a')$
$Z_1 = 4^{l_1}$
$V_0 = 3^{l_1}$
$Z_2 = 4^{l_1}$
$L = \bigcirc_{i\in[d]} \mathrm{CG}_1(a_i)$
$R = \bigcirc_{i\in[d]} \mathrm{CG}_1(a'_i)$
$V_1 = 3^{l_1}$
$V_2 = 3^{l_1}$
$D = \bigcirc_{i\in[d]} \mathrm{CG}_2(b_i)$
$\mathrm{VG}_2(b)$

Figure 1: Vector Gadgets as seen in [16]

This results to having $ED(VG_1(a,a'),VG_2(b)) = C + 2min(a \cdot b, a' \cdot b)$
We set $a' = 10^{d-1}$ so as to ensure $a' \cdot b = 1$ which results to having

$$ED(VG_1(a),(VG_2(b)) = \left\{ \begin{array}{l} C_0 \text{ if } a \cdot b = 0 \\ C_1 \text{ if otherwise} \end{array} \right\}$$

We will now show that OV $\leq$ PATTERN

- set $t = max(|VG_1|, |VG_2|)$
- set $T = 1000dt = \Theta(d^3)$
- set $f = 1^d$
- set $E_s := 2l_1 + l + d$
- set $E_u := 2l_1 + l + d + 2$

We define $VG'_k(a) = 5^T VG_k(a) 5^T$ for $k \in \{1,2\}$.
Let A and B be the sets from the OV instance.

$$P_1 = \bigcirc_{a\in A} VG'_1(a),$$
$$P_2 = (\bigcirc_{i=1}^{|A|-1} VG'_2(f))(\bigcirc_{b\in B} VG'_2(b))(\bigcirc_{i=1}^{|A|-1} VG'_2(f)).$$

Theorem 8 *Let $X := |A|E_u$. If there are two vectors $a \in A$,$b \in B$ such that $a \cdot b = 0$, then $PATTERN(P_1,P_2) \leq X - 2$; otherwise, $PATTERN(P_1,P_2) = X$.*

Set $P'_2 = P_2$ and $P'_1 = 6^{|P'_2|} P_1 6^{|P'_2|}$.

Theorem 9 *Let $Y := 2|P_2'| + X$. If there are no two orthogonal vectors, $ED(P_1', P_2') = Y$ otherwise $ED(P_1', P_2') \leq Y - 2$*

Theorem 10 *If EDIT can be computed in time $O(n^{2-\delta})$ for some $\delta > 0$ on two sequences of length n, then the OV problem with $|A| = |B| = N$ and $A, B \subset 0, 1^d$ can be solved in time $d^{O(1)}N^{2-\delta}$*

3.2.1.5 Distance/Similarity Problems [1, 26, 16, 23]

Distance Problems, are problems which measure the distance of two given strings of input, according to some similarity or metric, such as amount of operations of a given type needed to convert one string to another. These problems have been thoroughly studied for their applications in many fields such as data mining, DNA and protein analysis, search engines, and many more. In [23] Karl Bringmann showed that the Frechet Distance cannot be computed by an algorithm of subquadratic complexity unless SETH falls. Following that, Artur Backurs and Piotr Indyk showed in [16] that Edit Distance between two strings also requires quadratic time to be computed. After that, [1] showed some tight hardness results based on SETH for the Longest Common Subsequence problem, as well as many problems that have similar definitions as the LCS problem. These reductions made use of gadgets and algorithmic tricks tailored to each problem in order to strictly bind these problems with the SETH and OV conjectures.

In [26] Bringmann and Kunnemann introduced a framework that utilizes ”alignment gadgets” as they call them, which encapsulates all of these problems, as well as the Dynamic Time Warping problem, to easily produce reductions from the OV conjecture and a variant called the unbalanced OV conjecture to any of them. This was a major breakthrough in the field, and the fine-grained study on distance problems is now considered to be successful. This is the only major group of problems that has been so successfully studied by fine-grained complexity, and these results serve as a confirmation of the potential of the field.

We will now discuss briefly on the framework they implemented:
Loosely put, an alignment gadget consists of two instances x and y, whose distance/similarity $\delta(x, y)$ is closely related to $\Sigma_{(i,j)\in A}\delta(x_i, y_i)$ where A is the best possible ordered alignment of the numbers in [m] and [n].

Definition 9 *An alignment is defined as a set $\{(i_1, j_1), ..., (i_k, j_k)\}$ that represents aligning inputs s and r of size n and m respectively in the corresponding points i and j. Any i,j that do not belong in A are called unaligned. An alignment that aligns all neighboring m objects in s with the corresponding alignments in r is called a structured alignment ($\{(\Delta + 1, 1), ..., (\Delta + m, m)\}, 0 \leq \Delta \leq n - m$).*

The set of all alignments is denoted by $A_{n,m}$, and the set of all structured alignments is denoted by $S_{n,m}$ There are two main properties that need to be satisfied by the distance(or similarity) metric in order to use these gadgets:

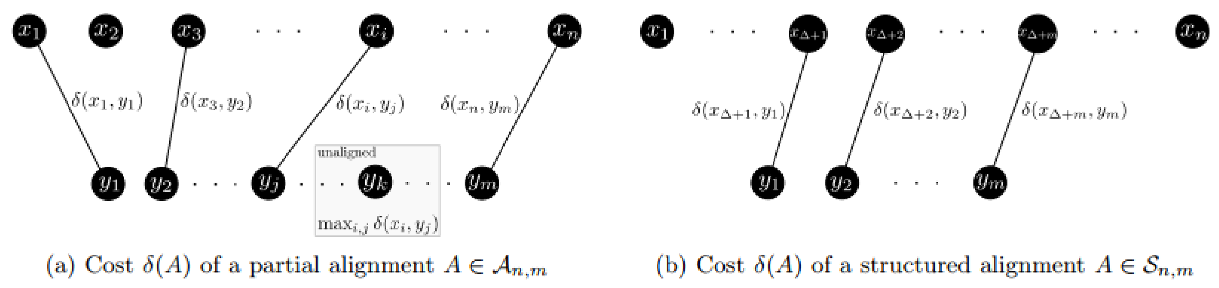

unaligned
$\max_{i,j} \delta(x_i, y_j)$
$\delta(x_1, y_1)$
$\delta(x_3, y_2)$
$\delta(x_i, y_j)$
$\delta(x_n, y_m)$
$\delta(x_{\Delta+1}, y_1)$
$\delta(x_{\Delta+2}, y_2)$
$\delta(x_{\Delta+m}, y_m)$
(a) Cost $\delta(A)$ of a partial alignment $A \in \mathcal{A}_{n,m}$
(b) Cost $\delta(A)$ of a structured alignment $A \in \mathcal{S}_{n,m}$

Figure 2: Alignments as seen in [26]

- δ admits an alignment gadget if given inputs $\{x_1...x_n\}$ and $\{y_1...y_m\}$, we can construct instances x and y, such that the amount $\delta(x, y) - C$ is bounded between the cost of the best alignment, and the cost of the best structured alignment (for an appropriate C).

  $$min_{A \in A_{n,m}} \delta(A) \leq \delta(x, y) - C \leq min_{S \in S_{n,m}} \delta(S)$$

  The authors show that one cannot compute such a value in subquadratic time, unless OV fails.

- δ admits coordinate values, if there exist gadgets for $0_x, 1_x, 0_y, 1_y$ satisfying $\delta(1_x, 1_y) > \delta(0_x, 1_y) = \delta(1_x, 0_y) = \delta(0_x, 0_y)$. Note that this property is exactly the one used in the aforementioned reduction of OV to edit distance by Indyk and Backurs, to make sure that if $x_i y_i = 0$ the cost is smaller than the case that $x_i y_i = 1$

All that is left is to show that computing this value is equivalent to the corresponding problem each time, and to produce specific coordinate gadgets for each problem that respect the properties above.

We will not go into further depth regarding the type setting and more technical issues of the proofs as their main advancement is to generalize the metrics and input types, in a way that helps produce the same arguments as the proof of $OV \leq EditDistance$ or $OV \leq FrechetDistance$. It suffices to say that the padding gadgets used to force the alignment to our problem are a function of the types of input and similarity/distance metric each time. Nevertheless, we will discuss some of the specific gadgets and results so as to give some intuition on the way the framework can be used.

LCS admits coordinate values by setting:

$$1_x = 11100, 0_x = 10011, 1_y = 00111, 0_y = 11001.$$

The alignment gadgets for LCS are realized as such:$G(z) := 1^{\gamma_2} 0^{\gamma_1} z 0^{\gamma_1} 1^{\gamma_2}$

$$x := G(x_1) 0^{\gamma_3} G(x_2) 0^{\gamma_3} \ldots 0^{\gamma_3} G(x_n)$$
$$y := 0^{n\gamma_4} G(y_1) 0^{\gamma_3} G(y_2) 0^{\gamma_3} \ldots 0^{\gamma_3} G(y_n) 0^{n\gamma_4}$$

where the padding gadgets γ_i are constants created as functions of the type of input in each case.

Regarding Edit Distance
Rule out the easy case: $Edit(c_{del-x}, c_{del-y}, c_{match}, c_{subst})$ can be solved in constant time if $c_{subst} = c_{match}$ or $c_{del-x} + c_{del-y} \leq min(c_{match}, c_{subst})$

Now for $c_{del-x} = 1, c_{del-y} = 1, c_{match} = 0, 0 < c_{subst} \leq 2$ *Edit(c_{subst}) admits coordinate values by setting:*

$$1_x = 11100, 0_x = 10011, 1_y = 00111, 0_y = 11001.$$

The alignment gadgets for Edit(c_{subst}) are realized as such:$G(z) := (1^{\gamma_1}0^{\gamma_1})^{\rho}z(0^{\gamma_1}1^{\gamma_1})^{\rho}$ for $\rho = 2\lceil 1/c_{subst} \rceil$

$$x := G(x_1)0^{\gamma_2}G(x_2)0^{\gamma_2}\ldots 0^{\gamma_2}G(x_n)$$
$$y := 0^{n\gamma_3}G(y_1)0^{\gamma_2}G(y_2)0^{\gamma_2}\ldots 0^{\gamma_2}G(y_n)0^{n\gamma_3}$$

In the same publication, similar gadgets are produced for the Dynamic Time Warping, Longest Palindromic subsequence, and Longest Tandem subsequence problems.

3.2.2 More Reductions from SETH and k-OV

Fine-grained reductions are transitive in nature. To elaborate, if problem A is reduced to problem B, and problem B is in turn reduced to problem C, then A is also reduced to problem C via the concatenation of the reductions. In this manner, most of the reductions stemming from SETH go through the Orthogonal Vectors Hypothesis. Because of this, we group problems associated with either of these conjectures together. Note also that even if SETH fails at some point in the future, the OV hypothesis (OVH) could still hold true, as there has only been a reduction showing $K - SAT \leq k - OV$, and not the reverse. Remember that SETH bounds the K-SAT problem to exponential time (2^n), and the OVH bounds the k-OV problem to quadratic time (n^2).

3.2.2.1 Subset Sum, Bicriteria s,t-Path

Subset Sum is one of the most studied problems in complexity theory and cryptography. The problem is to determine, given a set of numbers S and a target T, if there exists a non-empty subset of S such that its elements sum to T.

Problem: SUBSET SUM

Input: a set of numbers S and a target T

Output: a non-empty $S' \subseteq S$ s.t. $sum(S') = T$

A related problem is the Bicriteria s,t-Path problem, which is, given a graph $G = (V, E)$ with positive integer weights w_i and positive integer lengths l_i on its edges, and two distinguished nodes s, t, to determine if there exists a path from s to t with weight $\leq W$ and length $\leq L$ for given W and L.

Problem: BICRITERIA S,T-PATH

Input: a graph $G = (V, E)$, Positive integer weights w_i and lengths l_i of the edges, integer limits W and L, and two nodes s, t

Output: a path from s to t with weight $\leq W$ and length $\leq L$.

In [9] they show that $k-SAT \leq SubsetSum$ by reducing k-SAT to an intermediate problem called StructCSP and in turn reducing that to the Subset Sum problem. The two main results of this publication were a) showing that Subset Sum cannot be solved in time $T^{1-\epsilon}2^{o(n)}$ for any $\epsilon > 0$ unless SETH fails, and the statement separating the Subset Sum problem from the Bicriteria s,t-Path problem which is thought to be linked with Subset Sum, b) on graphs with m edges and edge lengths bound by L, the pseudo-polynomial algorithm cannot be improved to $\tilde{O}(L + m)$, in contrast to recent results on Subset Sum.

3.2.2.2 Subtree Isomorphism

The Subtree Isomorphism problem is to determine if a given tree is a subgraph of another given tree. Some variants and specialized cases of the problem have near-linear time algorithms that solve it, but in the general case a subquadratic algorithm has not been found.

Problem: SUBTREE ISOMORPHISM

Input: Two tree graphs T and T'

Output: "Yes" if T' is a subgraph of T.

[10] reduces the OV problem to the Subtree Isomorphism problem, showing that truly subquadratic algorithms for the latter would refute the k-OV conjecture, and in turn the SETH conjecture. In the same publication, they go on to prove quadratic lower bounds for some specified variants of the problem such as binary rooted trees, even limiting the depth of the tree to $loglogn$, and finally they show that for every constant d, there is a constant $\epsilon_d > 0$ and a randomized, truly subquadratic algorithm for degree-d rooted trees of depth at most $(1 + \epsilon_d)log_d n$.

3.2.2.3 Regular Expression Matching

Regular Expressions are in the foundation of formal languages and have been very impactful in the course of Computer Science and Logic.

The concept arose in the 1950s when the American mathematician Stephen Cole Kleene formalized the description of a regular language. The concept came into common use with Unix text-processing utilities. Since the 1980s, different syntaxes for writing regular expressions exist, one being the POSIX standard and another, widely used, being the Perl syntax.

> Problem: REGULAR EXPRESSION MATCHING
>
> Input: a Regular Expression R and text T.
>
> Output: A string from the input text T that matches R

In [17] Backurs and Indyk showed specifically for the case of expressions with depth 2 that matching and membership testing can be solved in near-linear time, except for the case ”concatenations of stars”, which cannot be solved in strongly sub-quadratic time assuming the Strong Exponential Time Hypothesis (SETH).

[24] was a followup study on the matter that finalized the dichotomy that was presented previously with two results. (1) They present two almost-linear time algorithms that generalize all known almost-linear time algorithms for special cases of regular expression membership testing and (2) classify all types, except for the Word Break problem, into almost-linear time or quadratic time assuming the Strong Exponential Time Hypothesis. The Word Break problem is considered to be the only intermediate problem in this case, and they also give an improved algorithm for the problem.

3.2.2.4 Model Checking Problems

Model Checking Problems consist of checking a model in an automatic way to see if its behavior is the one expected by the creator. These problems are mainly focused in checking logic models, as these combine both the expressiveness to encapsulate problems and the mathematical grasp to be able to check by an algorithm.

> Problem: MODEL CHECKING
>
> Input: a Model and its expected behavior, most commonly a logic model.
>
> Output: ”Yes” if the model has the expected behavior, or in the case of logic models, if it can be satisfied with a solution (e.g. an assignment of variables).

One of the techniques utilized in checking logic models is to formulate them as games in graphs with certain properties. Some of the more important ones were Büchi, Street, and Rabin automata. The authors of [33, 34] made advances on various model checking algorithms, and proved with these techniques the conditional optimality of their algorithms for these games based on SETH.

3.2.2.5 Succinct Stable Matching

The stable matching problem has been studied both for its use in graph theory, economics, and mathematics. It consists of finding a stable matching between two sets of equal size, given an ordering of preferences for the matching of each element. A variant of this problem is when the preference lists are given in a succinct form.

> Problem: SUCCINCT STABLE MATCHING
>
> Input: Two sets A and B of equal size, and a compressed form of the ordering of preferences to match each element of one set to an element of the other.
>
> Output: A set of relations matching elements of A to elements of B

[78] considers the succinct stable matching problem, and provides a dichotomy of complexity in the problem. Specifically, they give some subquadratic algorithms two special cases of the problem, the d-attribute and d-list problem, and show that for $d = \omega(logn)$ both finding and verifying a stable matching in the d-attribute model requires quadratic time assuming the Strong Exponential Time Hypothesis. The d-attribute model is therefore as hard as the general case for large enough values of d.

3.2.2.6 Machine Learning Problems

Empirical risk minimization (ERM) is a principle in statistical learning theory which defines a family of learning algorithms and is used to give theoretical bounds on their performance.In general, the risk cannot be computed because the distribution is unknown to the learning algorithm (this situation is referred to as agnostic learning). However, we can compute an approximation, called empirical risk, by averaging the loss function on the training set: It is one of the areas that are elusive as to their exact complexity. Some of the most popular such problems are kernel SVMs, kernel ridge regression, and training the final layer of a neural network.

> Problem: EMPIRICAL RISK MINIMIZATION
>
> Input: A collection of data of which we don't have any information about their distribution.
>
> Output: A prediction of the distribution of future data that minimizes the risk of faults.

[18] addresses this issue for multiple ERM problems, and give conditional hardness results for these cases. They show that there are no algorithms that solve the aforementioned ERM problems to high accuracy in sub-quadratic time and give similar hardness results for computing the gradient of the empirical loss, which is the main computational burden in many non-convex learning tasks.

3.2.2.7 One-Dimensional Dynamic Problems

The Least-Weight Subsequence (LWS) problem on [a,b] is to determine, given a weight function $W(i,j)$ that is defined for all $a \leq i < j \leq b$, what is the sequence of pointers with $i_0 < i_1 < ... < i_t$ that has the minimum added pairwise weight.

Problem: LEAST-WEIGHT SUBSEQUENCE

Input: an interval [a,b] and a weight function $W(i,j)$ defined for $a \leq i < j \leq b$

Output: a sequence of pointers with $i_0 < i_1 < ... < i_t$ that has the minimum added pairwise weight $\Sigma_{j\in[t]} W(i_{j-1}, i_j)$

Most One-dimensional Dynamic problems can be written in a form that resembles the Least-Weight Subsequence (LWS) problem. In [72], the authors show subquadratic equivalences between some cases of the LWS problem, and various core problems that have been previously studied. Specifically, they show equivalences between:

- a low rank version of LWS $\equiv$ minimum inner product,
- finding the longest chain of nested boxes $\equiv$ vector domination,
- a coin change problem (close to knapsack) $\equiv$ (min,+)-convolution.

They use these equivalences, as well as some SETH-hardness results in literature, to deduce tight conditional lower bounds for the corresponding LWS instantiations.

3.2.2.8 Furthest Pair in $\mathbb{R}^n$

The Furthest Pair problem is to determine, given a set of points and their corresponding space and distance metric, which two points are further away from each other.

Problem: FURTHEST PAIR IN $\mathbb{R}^n$

Input: A set of points and their corresponding space and distance metric

Output: Points a, b such that $d(a,b) > d(i,j)$ for all other points i, j

Point location problems in Euclidean space are inherently dichotomized into problems that have *near-linear* and *barely subquadratic* time complexities respectively. In [89] the author gives a self-reduction of the Orthogonal Vectors problem on n vectors $\{0,1\}^d$ to n vectors in $\mathbb{Z}^{\omega(logd)}$ that runs in $2^o(d)$ time. This reduction suffices to show that *barely subquadratic* problems such as the Euclidean Diameter, Bichromatic closest pair, and incidence detection do not have truly subquadratic algorithms, unless the OV conjecture fails.

3.2.2.9 Grammar Compression and SLP's

In [8] the authors analyze string distance problems in the case where the data is compressed, specifically in cases where the compression can be represented by the notion of Grammar Compression, i.e. Straight Line Programs. The main problems they address are: the Longest Common Subsequence (LCS),Pattern Matching with Wildcards, Context Free Grammar parsing, RNA folding, and the Disjointness problem. The goal is to determine if there can be an algorithm that solves the classic string problems on compressed strings with complexity faster than decompressing the string and solving it with the normal algorithm.

They discuss a $O(nN\sqrt{(logN/n)})$ bound for LCS and a $O(min\{NlogN, nM\})$ bound for Pattern Matching with Wildcards, and show the conditional optimality of these bounds for these cases under SETH. They also show that for the case of Context Free Grammar parsing and RNA folding, the decompress-and-solve technique is optimal under the k-clique conjecture. Finally, they give an algorithm for the Disjointness problem, that runs faster than the decompress-and-solve version.

3.2.2.10 All Pairs Max Flow

The All Pairs Max Flow problem is to compute, given a directed graph with n nodes, m edges, and capacities in the range of [1..n], the maximum flow value between each pair of nodes.

Problem: All Pairs Max Flow

Input: A directed graph $G = (V, E)$, edge capacities $1 \leq c_i \leq n$

Output: the maximum flow value between each pair of nodes in G

[71] provides evidence that the problem cannot be solved significantly faster than $O(n^2m)$ unless **SETH** falls. Additionally, in [75] it is shown that a single maximum $st-flow$ in such graphs can be solved in time $\tilde{O}(m\sqrt{n})$.

These are conterintuitive since it was conjectured in [73] that All-Pairs-Max-Flow in general graphs can be solved faster than $O(n^2)$ computations of maximum $st - flow$.

3.2.2.11 Closest Pair in Hamming Space

Neighbor problems such as the Closest Pair have many applications. In [13] Alman and Williams show (amongst major advancements on algorithms for the problem) that if there exists an $\epsilon > 0$ such that for all constants c, the Bichromatic Hamming Closest Pair problem can be solved in time $2^{o(d)}n^{2-\epsilon}$ time on a set of points in $\{0,1\}^{clogn}$, then the OV hypothesis would fail (and SETH in turn would be falsified)

3.2.3 More Reductions from 3-SUM

The 3SUM conjecture has proven to be a valuable tool for proving conditional lower bounds on dynamic data structures and graph problems. We will discuss here some of the problems in Computational Geometry that have been linked to the 3SUM problem via fine-grained reductions.

3.2.3.1 Problems in Computational Geometry

Computational geometry involves the study of algorithms which can be stated in terms of geometry. The development of computational geometry as a discipline was stimulated by progress in computer graphics and computer-aided design and manufacturing (CAD/CAM).

Other important applications of computational geometry include robotics , geographic information systems (GIS), computer-aided engineering (CAE) , computer vision (3D reconstruction).

In [52, 53] it is proven for a large class of problems that they are all at least as difficult as the base problem 3SUM and are classified as 3SUM-hard problems. Because the base problem can be reduced to all of these problems, none of them can be solved in time $o(n^2)$, unless a subquadratic solution exists for the base problem. Note that this does not mean that all problems are equivalent but it means that they are at least as hard as the base problem. Moreover, almost any lower bound for the base problem will immediately carry over to the 3SUM-hard problems.

Computational Geometry is a field that has proven to be invaluable in many areas including robotics, bioinformatics, data science, and more. In [52, 53] it is shown that the 3SUM problem is computationally equivalent to many of the basic Computational Geometry problems, such as incidence problems, separator problems, covering problems, visibility problems, and motion planning problems.

The list of problems includes among others:

- Given a set of lines in the plane, are there three that pass through the same point?
- Given a set of (non-intersecting, axis-parallel) line segments, is there a line that separates them into two non-empty subsets?

- Given a set of (infinite) strips in the plane, do they fully cover a given rectangle?
- Given a set of triangles in the plane, compute their measure.
- Given a set of horizontal triangles in space, can a particular triangle be seen from a particular viewpoint?
- Given a set of (non-intersecting, axis-parallel) line segment obstacles in the plane, and a rod, can the rod be moved, allowing translation and rotation, from a given source to a given destination without colliding with the obstacles?
- Given a set of (horizontal) triangle obstacles in space, and a vertical rod, can the rod be moved, allowing translation only, from a given source to a given destination without colliding with the obstacles?

Note that these reductions were produced a long time before Fine-Grained complexity was introduced, but still obey the definition. This serves as an indication that these types of equivalences have interest even in areas not only concerned with analyzing the complexity of problems, but to put them into practical use as well.

3.2.4 More Reductions from APSP

Here we consider some central problems in Graph Theory, and their relations with the very important and well-known problem, the All-Pairs-Shortest-Path(APSP) problem. APSP is believed to be truly cubic(i.e. there is no exact algorithm for this problem which runs in time $O(n^{3-\epsilon})$ for a constant $\epsilon > 0$).

3.2.4.1 Graph Problems

In [91] the authors analyze some Graph problems and show subcubic equivalences between the APSP problem and various problems that have been very important for their applications in network and graph algorithms. Namely, they show equivalances between APSP and the following problems:

- Detecting if a weighted graph has a triangle of negative total edge weight.
- Listing up to $n^{2.99}$ negative triangles in an edge-weighted graph.
- Finding a minimum weight cycle in a graph of non-negative edge weights.
- The replacement paths problem in an edge-weighted digraph.
- Finding the second shortest simple path between two nodes in an edge-weighted digraph.

- Checking whether a given matrix defines a metric.
- Verifying the correctness of a matrix product over the (min, +)-semiring

[3] continued this work to show equivalences between APSP and some problems associated with graph centrality. To be specific, they give reductions linking APSP to the computation of the *center* and the *median* of a graph, and computing the *betweenness* or *reach centrality* of even a single node in the graph.

3.2.4.2 Matrix Problems

In [91] the authors show generic equivalences between matrix products over a large class of algebraic structures used in optimization, verifying a matrix product over the same structure, and corresponding triangle detection problems over the structure. As a consequence of their work, they come up with new combinatorial approaches to Boolean matrix multiplication over the (OR, AND)-semiring (abbreviated as BMM). Finaly they show that practical advances in triangle detection would imply practical BMM algorithms, among other results. They also give two new BMM algorithms: a derandomization of the combinatorial BMM algorithm of Bansal and Williams (FOCS'09), and an improved quantum algorithm for BMM.

Specifically, given a tripartite graph G on n vertices, we can detect if there is a triangle in G using a combinatorial algorithm in $O(n^3/log^4n)$ time on a word RAM with word size $\omega \geq \Omega(logn)$. [91] proved that triangle detection and Boolean matrix multiplication are subcubic equivalent. For any constant c, if we can solve triangle detection on n-node graphs in $\hat{O}(n^3/log^cn)$ time, we can also solve Boolean matrix multiplication on $n \times n$ matrices in the same running time. Combining the above, results in a fast combinatorial algorithm for Boolean matrix multiplication, i.e. there is a combinatorial algorithm to multiply two $n \times n$ Boolean matrices in $\hat{O}(n^3/log^4n)$ time.

In [15] the authors study the Maximum Weight Rectangles problem, and provide tight conditional lower bounds based on the APSP conjecture, specifically in the case where the points are aligned in a grid which is a well studied problem called the Max Subarray Problem.

3.2.4.3 Tree Edit Distance

The Tree Edit Distance problem is a generalization of the edit distance problem, where instead of strings, the objects that are measured are rooted Trees with n nodes and symbols on each node

The fastest known algorithm for tree edit distance runs in $O(n^3)$ time and is based on a dynamic programming solution similar to the one for string edit distance.

In [27] the authors show that a subcubic algorithm for this problem is unlikely to be developed. Specifically, they show that for $|\Sigma| = \Omega(n)$, a truly subcubic algorithm for tree edit distance implies a truly subcubic algorithm for the all pairs shortest paths problem. Furthermore, for $|\Sigma| = O(1)$, a truly subcubic algorithm for tree edit distance implies an $O(n^{k-\epsilon})$ algorithm for finding a maximum weight k-clique.

3.2.5 Other Reductions

3.2.5.1 Local Alignment

The Local Alignment problem is, given two strings s and r and a scoring function on pairs of letters, to determine which are the substrings of the s and r that are most similar under the scoring function. It is a special case of the Edit Distance problem, and is very important for its applications in biology and bioinformatics. The best known algorithm for this problem is of quadratic time complexity. In [6] we see that if there exists an algorithm running in $O(n^{2-\epsilon})$ for some $\epsilon > 0$ then there exists a $\delta > 0$ for which 3SUM numbers is in $O(n^{2-\delta})$, CNF-SAT is in $O((2-\delta)^n)$, and Max Weight 4-Clique is in $O(n^{4-\delta})$ time for input size *n* respectively.

3.2.5.2 Triangle Collection, Matching Triangles

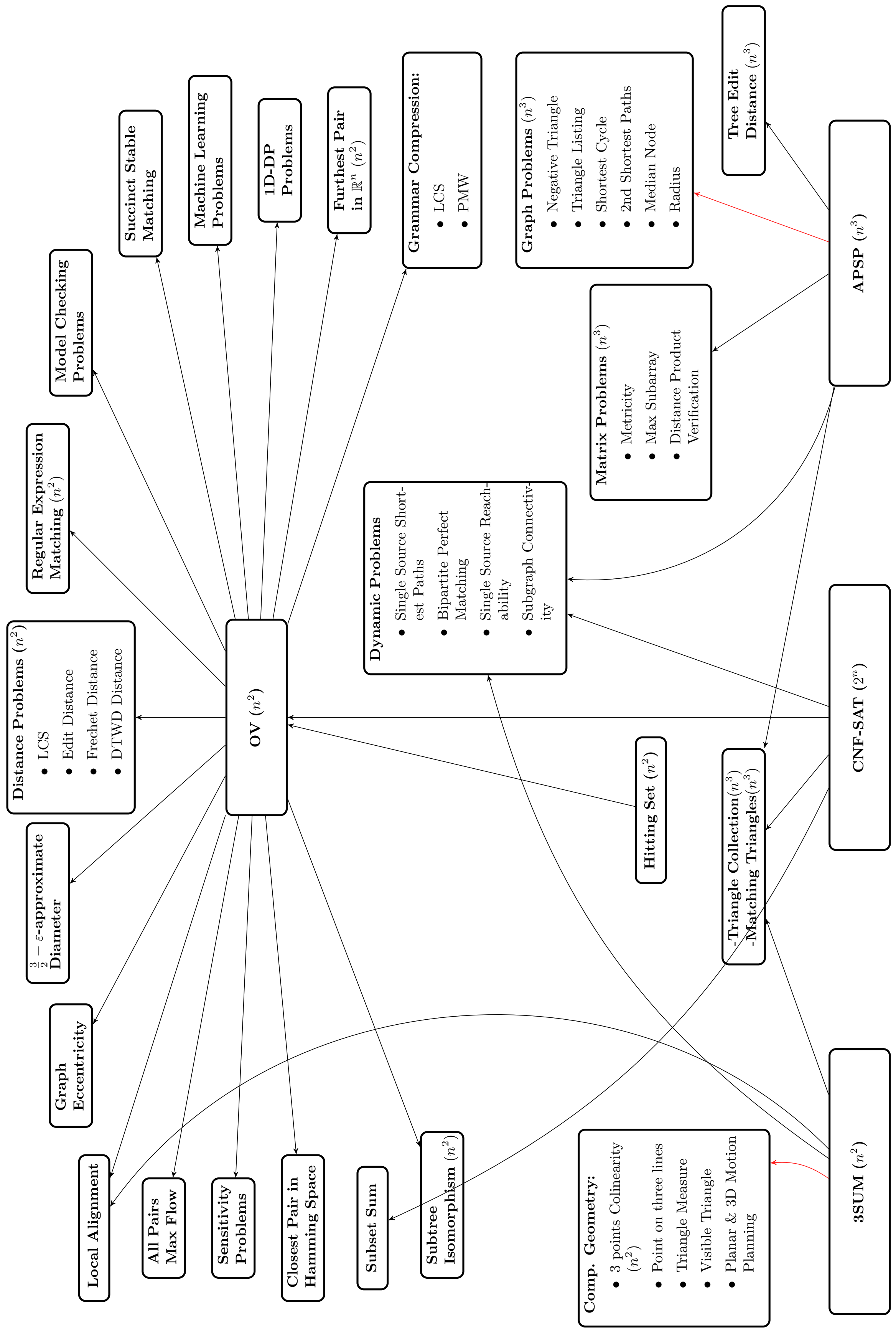
One of the main questions regarding fine-grained reductions is to find links between the three basic conjectures, namely SETH, 3SUM, and APSP. While a direct link remains elusive, [7] shows that there exist problems that have reductions from all three conjectures to them. Specifically, they show reductions from all these conjectures to the Triangle Collection problem, and the Matching Triangles problem. This could potentially indicate that these problems have all the structural properties that bound the complexity of the three basic conjectures.

Among their results are tight n^{3-1} lower bounds for purely-combinatorial problems about the triangles in unweighted graphs. Furthermore, [7] show new $n^{1-o(1)}$ lower bounds for the amortized update and query times of dynamic algorithms for single-source reachability, strongly connected components, and Max-Flow, as well as new $n^{1.5-o(1)}$ lower bound for computing a set of n st-maximum-flow values in a directed graph.

3.2.6 Web of Reductions

Figure 3: Web of Reductions

Fine-Grained Reductions Landscape



4. STRUCTURAL IMPLICATIONS

4.1 Properties and Implications of Fine-Grained Reductions

As with the classic polynomial reductions, fine grained reductions have many implications on all of the problems connected through them.
Firstly, it is easy to see that these reductions have the transitive property, i.e. if problem A can be reduced to problem B, and problem B can in turn be reduced to problem C, then problem A can also be reduced to problem C by pipelining the two reductions. As such, any problem connected via a chain of reductions to another problem share all the implications provided by fine-grained reductions. Nevertheless, extracting these structural properties from the fact that these problems have such a reduction is one of the biggest questions in fine-grained complexity.
It is important to find properties of problems that are preserved via fine-grained reductions, meaning that if problem A has such a property, it can be reduced to problem B in a fine-grained way, then problem B would also have this property. This would mean that all problems that do not have this property, can never take part in a reduction from A to them, which in turn implies that their hardness is possibly based on a different structural property.

4.1.1 NSETH and non-reducibility

In [31] the authors introduce a Nondeterministic version of the Strong Exponential Time Hypothesis. NSETH claims that a ”co-nondeterministic machine” solving the k-SAT problem also requires exponential time to be solved. The hypothesis can be equivalently be seen as stating that the a nondeterministic machine cannot solve the Tautology or the non-SAT problem in less than exponential time. They show that refuting this hypothesis would give some interesting lower bounds for circuit complexity. They also introduce a structural property of problems (we will refer to this as property X) that is preserved via fine-grained reductions.
This property states that either the nondeterministic or the co-nondeterministic complexity of a given problem is at least as hard as the deterministic case. This can alternatively be seen through the lens of certificate length for non-determinism:

If the time required to verify the shortest certificate for either the problem or the complement of the problem (e.g. SAT and non-SAT) is at least the time required to solve the problem in a deterministic way, then the property holds. In this context, NSETH is translated simply as such: ”The shortest certificate for the non-SAT problem is of exponential length”. Since k-SAT can be solved exponentially by a deterministic machine, this conjecture implies property X holds for k-SAT.

The authors proceed to show that if property X does not hold for a problem B, then the fine-grained reduction can be used to create a non-deterministic algorithm for both the problem and its complement. As such, property X is preserved via fine-grained reductions in the sense that, if problem A has this property (w.l.o.g. say the co-nondeterministic complexity of the problem is as much as the deterministic one) and a fine-grained reduction to problem B which in turn doesn't have this property, then that reduction can be used to produce both a nondeterministic and a co-nondeterministic algorithm for problem A, which would mean that problem A doesn't have this property. In other words, if A can be reduced to B, either both of these problems have this property, or none of them do.
Since NSETH claims that the k-SAT has Property X, then all of the problems k-SAT can be reduced to also have this property. It is important to note that the reverse is not shown: if a problem B has property X, that does *not* necessarily mean that k-SAT *can* be reduced to that problem.

4.1.2 Interesting Results and viewpoints

Combining the structure and transitivity of fine-grained reductions with the an analysis of the non-deterministic and co-non-deterministic complexities of problems, one can show that, under NSETH, there can never exist fine-grained reductions between many of the problems studied in the field. To put it formally, the following hold:

Theorem 11 *(NSETH implies no reduction from SAT). If NSETH and $C \in (N \cap coN)TIME[T_C]$ for some problem C, then (SAT, 2^n) $\not\leq_{FGR}$(C, $T_C^{1+\gamma}$) for any $\gamma > 0$.*

Corollary 1 *(NSETH implies no reductions from SETH-hard problems). If NSETH and $C \in (N \cap coN)TIME[T_C]$, then for any B that is SETH-hard under deterministic reductions with time T_B, and $\gamma > 0$, we have $(B, T_B) \not\leq_{FGR} (C, T_C^{1+\gamma})$*

Theorem 12 *Under NSETH, there is no deterministic or zero-error fine-grained reduction from SAT or any SETH-hard problem to the following problems with the following time complexities for any $\gamma > 0$.*

- *MAXFLOW with $T(m) = m^{1+\gamma}$*
- *HITTINGSET with $T(m) = m^{1+\gamma}$*
- *3-SUM with $T(n) = n^{1.5+\gamma}$*
- *ALL-PAIRS SHORTEST PATHS with $T(n) = n^{2+\frac{6+\omega}{9}+\gamma}$*

4.1.3 Some Proof sketches for the above

The above results stems from combining the initial theorem with the fact that these problems belong to the class $(N \cap coN)TIME[T_C]$ for their respective T_C time function. We

will now see parts of the analysis of these problems' non-deterministic complexities.
The technique used to succeed in this is simply to provide non-deterministic algorithms for the specific problems and their respective complements, therefore showing that they belong in both $NTIME[T_C]$ and $coNTIME[T_C]$

4.1.3.1 3-SUM

As mentioned, to prove that 3-SUM cannot take part in a reduction from k-SAT to 3-SUM, it suffices to show a non-deterministic algorithm for the problem it self, as well as a non-deterministic algorithm for the complement of the problem, which is to prove that there are no triplets in the list that sum to 0.
For the initial problem there is an easy algorithm of non-deterministically guessing a triplet and checking if it sums to 0. For the case of showing that no triplet sums to 0, one can do so with complexity $O(n^{1.5})$ using number-theoretic properties to create a proof of such length, that can be then non-deterministically guessed and verified within the specified complexity.
The proof is of the form (p, t, S) such that

- p is a prime number such that $p \leq prime_{n^{1.5}}$, where $prime_{n^{1.5}}$ is the $n^{1.5}$'th prime number.
- t is a nonnegative integer with $t \leq 3cn^{1.5}logn$ such that $t = |\{(i,j,k) : a_i + a_j + a_k = 0modp\}|$ is the number of triplets that sum to 0 modulo p.
- $S = \{(i_1, j_1, k_1), \ldots, (i_t, j_t, k_t)\}$ is a set of t triplets of indices, such that for all $r = 1, \ldots, t$ we have $a_{i_r} + a_{j_r} + a_{k_r} = 0modp$ and $a_{i_r} + a_{j_r} + a_{k_r} \neq 0$.

To show that a proof of this required length and property exists, a counting argument needs to be made: Let R be the number of all pairs $((i,j,k), p)$ s.t. p is a prime $\leq prime_{n^{1.5}}$ and $a_i + a_j + a_k = 0modp$. Then $|R| \leq n^3 log(3n^c) < 3cn^3 logn$, as any integer z can have at most $log(z)$ prime divisors. Then by simple counting argument, there exists a prime $\leq prime_{n^{1.5}}$, such that the number of such pairs is at most $3cn^{1.5}logn$. The verification for such a proof needs to firstly check that for all $r \in [t]$ the following hold: $a_{i_r} + a_{j_r} + a_{k_r} = 0modp$ and $a_{i_r} + a_{j_r} + a_{k_r} \neq 0$, and then compute the number of 3-sums modulo p and compare it with t. In order to do the second step we expand the following expression using Fast Fourier Transformation (FFT) in time $\tilde{O}(t)$:

$$(\Sigma_i x^{(a_i modp)})$$

let b_j be a coefficient before x^j. We need to check that $b_0 + b_p + b_{2p} = t$. If this is true, then the proof is accepted, otherwise it is rejected.

Every check needed, as well as the FFT expansion, can be done in time $\tilde{O}(t) = \tilde{O}(n^{1.5})$. Therefore, the non-deterministic as well as the co-non-deterministic complexity of 3-SUM is faster than its deterministic one (n^2).

4.1.3.2 All-Pairs Shortest Paths

This proof is an indicator of how powerful the transitivity of fine-grained reductions is in producing results. The authors first show that the Zero Weight Triangle problem is in $(N \cap coN)TIME[O(n^{2+\omega/3})]$ using techniques closely related to the above proof for the 3-SUM problem.
Because in [85] it was shown that the Negative Triangle problem can be reduced to the Zero Weight Triangle Problem, it also belongs in $(N \cap coN)TIME[\tilde{O}(n^{2+\omega/3})]$ Additionally, [91] give a deterministic fine-grained reduction from the APSP problem to the Negative Triangle problem with time $\tilde{O}(n^2 T(n^{1/3}))$, where $T(n)$ is the time complexity of the negative weight triangle problem. As such, APSP is in $(N \cap coN)TIME[\tilde{O}(n^{2+\frac{6+\omega}{9}})]$.

4.1.4 Quantifier structure of SETH-hard graph problems

In addition to the aforementioned results, the authors of [31] observed that there is a logic coherence between graph problems that are SETH-hard. That is, graph problems that are SETH-hard seem to have logical forms that are very similar in nature and structure. They specialize in problems concerning first order properties in sparse graphs, meaning that the amount of edges m is not immensely bigger than the amount of nodes n.
The logic representation of the graph is done by representing every relation in the graph as a binary predicate. For instance, a predicate $P(x_1, x_2)$ is true on nodes x_1, x_2 if there is an edge between x_1, x_2 in the graph. To give some more context, we will give some examples of graph properties expressed in this way:
The k-Clique problem can be expressed as:

$$\phi = \exists x_1 \dots \exists x_k \wedge_{i,j \in [k], i \neq j} E(x_i, x_j)$$

Similarly, the k-Dominating Set:

$$\phi = \exists x_1 \dots \exists x_k \forall x_{k+1} (E(x_1, x_{k+1}) \vee \dots \vee E(x_k, x_{k+1}))$$

The maximum deterministic complexity of a k-quantifier formula for $k \geq 2$ is $O(m^{k-1})$. For $k = 2$, this is just linear in the input size, so matching lower bounds follow. So the interesting case is $k \geq 3$. If SETH is true, some formulas require approximately this time. But if NSETH holds, all such formulas that are SETH-hard are of the same logical form. This is made precise as follows:

Theorem 13 *If NSETH is true, then there is a k-quantifier formula whose model checking problem is $O(m^{k-1})$ SETH-hard, but all such formulas have the form $\forall^{k-1}\exists$ or $\exists^{k-1}\forall$.*

This theorem comes directly from the following lemmas:

Lemma 1 *If SETH or NSETH is true, then there are $\forall^{k-1}\exists$ problems that are SETH-hard for time $O(m^{k-1})$.*

By negating ϕ, the $\exists^{k-1}\forall$ are also SETH-hard. The following three lemmas show that if a problem is not of the aforementioned quantifier structure, then these have smaller non-deterministic complexity.

Lemma 2 *If ϕ has exactly one existential quantifier, but it is not on the innermost position, then it can be solved in $O(m^{k-2})$ non-deterministic time*

Lemma 3 *If ϕ has more than one existential quantifier, then it can be solved in $O(m^{k-2})$ non-deterministic time*

These problems can be solved by guessing the existentially quantified variables, and exhaustively searching on universally quantified variables. Because there are at most $k-2$ universal quantifiers, the algorithm runs in time $O(m^{k-2})$.

Lemma 4 *If all quantifiers are universal, then it can be solved in time $O(m^{k-1.5})$.*

Thus, assuming NSETH, only these two types of first-order properties might be SETH-hard for the maximum difficulty of a k-quantifier formula.

5. CURRENT APPROACH

5.1 Current Approach

In this section I will discuss my thoughts and approach on the field.

There are two main goals in my current approach on Fine-Grained Complexity. The first is to continue the search for structural properties of problems that are preserved via fine-grained reductions. The property I am currently studying in this manner is a variation of self-reducibility.
The second goal is to analyze the amount of information that can be produced via a fine-grained reduction that is inherently limited in computational time and possibly produce bounds based on the entropy of each problem

5.1.1 Observations

We will begin this chapter with some observations made while studying the web of fine-grained reductions.
Firstly, we can see that fine-grained reductions create a somehow clustered web. There are some basic conjectures, and the reductions that stem from them create an tree-like directed graph. While there are some intersections between the trees from each conjecture, for the most part these trees form clusters of reducibility where some families of problems seem to be very similar with each other. This implies that there are structural properties of problems that are shared within these clusters, but not globally.
Secondly, these tree-like graphs that link these problems, all stem from some initial conjecture, but there are no known fine-grained relations between these conjectures. One can view this as a sort of independence between these families of problems.
Another observation in the known and well-studied fine-grained reductions, is that hardness is in a way ”transferred” through parameters. That is, the entropy and structure of problem A that is reduced to problem B, is directly linked problem B via the reduction, not only as an information-theoretic construct, but also preserving some of the relations of the parameters. For example, in the $kSAT \leq OV$ reduction [87] the number of variables is explicitly transferred to the number of vectors in the sets $(N = 2^{n/k})$, and the number of clauses m in the formula is transferred to the number of dimensions of the vectors in the OV instance. The study done in [25] also gives some evidence in this direction, as they show hardness for specific parameters in problems.
Another interesting observation regarding the structure of fine-grained reductions, is the fact that these reductions go across complexity classes. In other words, a problem with exponential complexity can be reduced to a problem with polynomial complexity, regardless of the fact that these two problems belong in different complexity classes. In both the cases of classic complexity theory and fine grained complexity, a reduction from problem A to problem B is an indicator that problem B is ”harder than” or ”at least as hard as” problem A, as improvements on the running time of B immediately transfer to the com-

plexity of A. Nevertheless, in the case of fine-grained complexity, the notion of ”hardness” can be viewed via a different lens, as (to my knowledge) there have only been reductions linking problems of higher complexity to problems of lower complexity. This implies that the structural properties that cause a problem to be ”hard” in this manner, are not directly linked to the computational complexity of the problem itself.

These facts, along with the publication of property X and NSETH in [31] motivated us to find other structural properties that differ in the 3 basic conjectures (3SUM, SETH, APSP). The hope in such a study is to extract a structural property that is preserved via fine-grained reductions.

5.1.2 Self Reducibility

After discussing many properties related with locality of computation and the way the input is processed, we decided to study the notion of downward self-reduciblity, which is one of the most prolific and studied structural properties in problems, that is, the ability to solve a problem by having access to an oracle solving instances of smaller input size.

The first problem in the direction of self reducibility (SR), is the fact that the classic definition for downward SR allows the reduction to use poly(n) time, which in many cases results in an algorithm with running time complexity greater than the current optimal algorithm for the problem. We need to define a variant of Self Reducibility that respects the amount of resources needed for the reduction

Definition 10 *We say that a problem is Self-Reducible in a Fine-grained way, if for a (natural) parameter p of the problem we have that $A(p) \leq_{FGR} A(p')$, where $p' \prec p$, for a partial-ordering $\prec$ defined for the parameter p.*

An example of these notions is k-SAT, for which one can see that $kSAT(n) \leq_{FGR} kSAT(n-1)$, so in this regard we say that the k-SAT problem has the Fine-Grained Self Reducibility property.

Note that in this definition and example we used the Fine-Grained reduction from a problem with parameter p to a problem with parameter p’. This notation is not the one usually used in Fine-Grained complexity, as normally fine-grained reductions are not concerned with a certain instance of a problem that would define a parameter such as p, but are abstractly defined on all inputs for both of the problems. In order to properly define self-reducibility, we need a kind of parameterization of the input. This notation can be used freely because in fine-grained reductions, there is always a translation of the inputs, and as such a translation of all the parameters that can be extracted by each instance of the problems.

We can generalize this definition and notation by allowing a parameter to be tuples of other parameters ($P = (p_1p_2 \cdots p_k)$). Then we write $A(P) \leq_{FGR} B(R)$ for $P = (p_1p_2 \cdots p_k)$, $R = (r_1r_2 \cdots r_k)$ This can be viewed as the existence of a mapping family $f_i(p_i) = r_i$ implicit in the reduction process/algorithm.

It is only natural to see that this notation holds this property as an inherent structure of fine-grained reductions: If $A \leq B$, then $\exists(N_1, \cdots, N_k)$ parameters of B, such that $b_i^{1-\varepsilon}(N_i) \Rightarrow a^{1-\delta}(n)$.

This framework could culminate in results such as the following speculation:

Conjecture: *Self Reducibility is preserved through fine-grained reductions. That is, if a problem is Self Reducible on some parameter p in a Fine-Grained way and is FG-reduced (or reduces, TBD) to another problem, then this problem is also Self Reducible in a Fine-Grained way on the parameters that are the translation of p.*

While there is intuitive evidence for this conjecture, the proof for such a result remains elusive at the moment.

5.1.3 Information Theory

One of the main questions in Fine-Grained complexity is to define which problems can be linked via fine-grained reductions, as there have not been many non-reducibility results in the years this field has been active. This idea began when noticing that all of the reductions in literature start with a problem of high complexity, and reduce it to a problem of lower or equal complexity. This is not a completely new idea as it has been discussed by Amir Abboud in the Dagstuhl Seminar 16451 along with various other open problems regarding structural hardness in P.
My approach is as follows: Given a reduction from problem A to problem B, as defined previously, fine-grained reductions are bound to using less running time than the current algorithm for the problem A. The amount and size of the calls that can be made to instances of the second problem are also bound by the complexity of the initial problem:

$$\textstyle\sum_{i=1}^{q}(b(n_i))^{1-\epsilon} \leq (a(n))^{1-\delta}$$

The idea is that if the entropy of problem A is greater than the amount of information that can be gathered by these calls combined with the running time of the reduction, then problem A can never be reduced to problem B, therefore producing massive non-reducibility results, as this property would also carry the transitiveness of fine-grained reductions.

6. OPEN PROBLEMS

Formulating the right question is of paramount importance in science but it is not an easy matter as it requires deep knowledge and hands-on experience in the field. Meetings and conferences may aid in the formulation of open problems in different aspects in a field provided by different researchers.
Dagstuhl Seminar 16451 took place in November of 2016, having as its topic ”Structure Hardness in P”. During the seminar, researchers introduced various interesting open problems in order to motivate research in the field of Fine-Grained Complexity. These problems belong to a big range of research objectives, spanning from lower bounds for problems to structural properties of Fine-Grained reductions (similar to those discussed in this thesis). In [76] the authors gathered all these problems that were discussed there, and as such we will also provide the list in this chapter, as presented by the authors of the aforementioned publication.

6.1 Open Problems Discussed in Dagstuhl Seminar 16451

6.1.0.1 Open Problem 1: Parameterizing problems in P by treewidth

Let t be the treewidth of an input graph. Many NP-hard problems, particularly those expressible in MSOL, are solvable in $f(t)n$ time and there are lower bounds on the (exponential) function f conditioned on the Strong Exponential Time Hypothesis (SETH) [39]. For problems in **P** the picture is less clear. Consider your favorite problem Π in **P** solvable in $T_{\Pi}(n)$ time on a graph with n vertices. Some problems Π admit algorithms running in $poly(t)o(T_{\Pi}(n))$ time whereas others do not. For example, [5] proved that *Diameter* can be solved in $2^{O(tlogt}n^{1+o(1)}$ time, yet a $2^{o}(t)n^{2-\epsilon}$ time algorithm would refute SETH. On the other hand, maximum cardinality matching can be solved in randomized $O(t^3nlogn)$-time [45].

Question: Classify graph problems in P according to their dependence on treewidth. Which problems admit $f(t)n^{t+o(1)}$-time algorithms with polynomial function f, and which require exponential f? A specific goal is to determine whether maximum weight perfect matching has an $\tilde{O}(poly(t)n)$ algorithm, for integer weights from a polynomial range.

[Contributed by Fedor V. Fomin.]

6.1.0.2 Open Problem 2: Approximate all-pairs shortest paths

In unweighted, undirected graphs, we can compute All Pairs Shortest Paths (APSP) in $O(n^3)$ time with a fast ”combinatorial” algorithm, or in $O(n^{\omega})$ time, where $\omega < 2.373$

is the matrix multiplication exponent. It is conjectured that a truly subcubic combinatorial algorithm does not exist, which is equivalent to the combinatorial Boolean matrix multiplication conjecture. What about approximation algorithms? The best kind of approximation is an additive +2, so that for all pairs *u,v* we return a value that is between $d(u, v)$ and $d(u, v) + 2$. [43] presented a combinatorial algorithm with runtime $\tilde{O}(n^{7/3})$. Note that this runtime is currently even better that $O(n^{\omega})$, and has the advantage of being practical.

Question: Is there a conditional lower bound for +2-APSP? Can we show that a combinatorial algorithm must spend $n^{7/3-o(1)}$ time?

[Contributed by Amir Abboud.]

6.1.0.3 Open Problem 3: Approximate diameter

Computing the diameter of a sparse graph in truly subquadratic time refutes SETH: [81] showed that a $(3/2-\epsilon)$-approximation to the diameter requires $n^{2-o(1)}$ time, even on a sparse unweighted undirected graph under SETH. On the other hand, there are algorithms [81, 35] that give a (roughly) $3/2$ approximation in $\tilde{O}(m\sqrt{n})$ time on unweighted graphs, or $\tilde{O}(min\{m^{3/2}, mn^{2/3}\})$ time on weighted graphs. Extending these algorithms further, [29] showed that for all integers $k \geq 1$, there is an $\tilde{O}(mn^{1/k+1})$ time algorithm that approximates the diameter of an undirected unweighted graph within a factor of (roughly) $2 - 1/2^k$.

Question: If we insist on near-linear runtime, what is the best approximation factor we can get? It is easy to see that a 2-approximation can be achieved in linear time, but what about an α-approximation, where $3/2 \leq \alpha < 2$?

[Contributed by Amir Abboud.]

6.1.0.4 Open Problem 4: Finding cycles and approximating the girth

Consider an unweighted undirected graph $G = (V, E)$. The girth of G is the length of the shortest cycle. The problem of detecting 3-cycles (and odd cycles of any length) is reducible to matrix multiplication and there are reductions in the reverse direction; see [91]. Yuster and Zwick [92] showed that detecting 2k-cycles can be computed in $O(f(k)n^2)$ time, where f is exponential.

Question: For any fixed constant k, give a conditional lower bound, showing that there does not exist an algorithm deciding whether G contains a 2k-cycle in time $O(f(k)n^{2-\epsilon})$ for any $\epsilon > 0$, or one running in $O(f(k)m^{2k/(k+1)-\epsilon})$ time, where m is the number of edges.

Question: Prove or disprove the following conjecture: There exists a truly subquadratic algorithm for finding a 4-cycle in a graph if and only if there exists a truly subquadratic algorithm for finding a multiplicative $(2-\epsilon)$-approximation of the girth.

Question: Prove or disprove the following conjecture from [82]: the problem of detecting a 3-cycle in a graph G without 4- and 5-cycles requires $n^{2-o(1)}$ time. Note that if there exists a subquadratic $2-\epsilon)$-approximation for the girth, it must be able to detect 3-cycles in graphs without 4- and 5-cycles.

Main Paper Reference: [82]

[Contributed by Mathias Bæk Tejs Knudsen and Liam Roditty]

6.1.0.5 Open Problem 5: Minimum cycle problem in directed graphs

Given an unweighted directed graph $G=(V,E)$ on n vertices, the problem is to find a shortest cycle in G. The potentially simpler Girth problem asks to compute just the length of the shortest cycle.
The girth and the minimum cycle can be computed in $O(n^{\omega})$ time exactly, as shown by [63] , where $\omega < 2.373$. It is easy to see that the minimum cycle problem is at least as hard as finding a triangle in a graph. In fact, even obtaining a $(2-\delta)$-approximation for the girth for any constant $\delta > 0$ is at least as hard as triangle detection. The fastest algorithm for the Triangle problem in n node graphs runs in $O(n^{\omega})$ time.

Question: Is there any $O(1)$-approximation algorithm for the girth that runs faster than $O(n^{\omega})$ time? In recent work, [79] showed that for any integer k, there is an $\tilde{O}(mn^{1/k})$ time $O(klogn)$ approximation algorithm for the Minimum Cycle problem. Thus, in nearly linear time, one can obtain an $O(log^2 n)$-approximation. Can one improve the approximation factor further? Can one even obtain a constant factor approximation in linear time?

[Contributed by Virginia Vassilevska Williams.]

6.1.0.6 Open Problem 6: Linear Programming

Consider a linear program of the following form: minimize $c^T x$ subject to $Ax \geq b$, where A is a *d-by-n* constraint matrix. Suppose that we could solve any such LP in time

$$\tilde{O}((nnz(A)+d^2)d^{\delta}logL),$$

where $nnz(A)$ is the number of non-zero entries of A, L is the bound on the bit complexity of the input entries, and δ is a positive constant.

Question: Is there some value of δ for which the above (hypothetical) running time bound would disprove any of the popular hardness conjectures?
In [74], it is shown that one can achieve the above running time bound for $\delta = 1/2$.

[Contributed by Aleksander Madry.]

6.1.0.7 Open Problem 7: Fully dynamic APSP

In the fully dynamic all-pairs shortest paths (APSP) problem we are interested in maintaining the distance matrix of a graph under insertions and deletions of nodes. [41] showed that the distance matrix can be updated in *amortized* time $\tilde{O}(n^2)$ after each node update. The current fastest worst case algorithms have update times of $O(n^{2+2/3})$ (randomized Monte Carlo [11]) and $\tilde{O}(n^{2+3/4})$(deterministic [84]).

Question: Can the worst case update time $\tilde{O}(n^2)$ be achieved? A barrier for current algorithmic approaches is $n^2.5$. Is there a conditional lower bound showing this to be a true barrier?

[Contributed by Sebastian Krinninger.]

6.1.0.8 Open Problem 8: Dynamic reachability in planar graphs

Dynamic reachability in a planar graph G is the problem of maintaining a data structure supporting the following operations: (i) Insert a directed edge (u,v) into G, (ii) delete an edge from G, and (iii) query whether v is reachable from u in G.
An algorithm with update and query time $\tilde{O}(\sqrt{n})$ is known ([42] for dynamic plane graphs-that is, the graph is dynamic but the plane embedding is fixed.

Question: Does an $n^{1/2-\Omega(1)}$ algorithm exist or is there a conditional $n^{1/2-o(1)}$ hardness result? Any polynomial hardness result would be interesting. A good place to start for the latter part would be [2] about hardness for dynamic problems in planar graphs.

[Contributed by Søren Dahlgaard.]

6.1.0.9 Open Problem 9: Static hardness for planar graphs

An important direction is to show conditional hardness for important problems, even on restricted (easier) classes of graphs, e.g., planar graphs. [2] showed hardness for several dynamic problems in planar graphs, but nothing is known for static problems.

Question: On planar graphs, many problems (such as shortest paths, multi-source multi-sink max-flow, etc.) run in near-linear time. Can we show that some problem does not? No hardness results are known for any static problem in **P** on planar graphs. Two candidate problems to consider are diameter and sum of distances. Both require subquadratic time ([28]), but it may still be possible to show a hardness result, e.g., $n^{3/2-o(1)}$ hardness.

[Contributed by Søren Dahlgaard.]

6.1.0.10 Open Problem 10: Sparse reductions for graph problems

Many graph problems are known to be as hard as APSP on dense graphs [3, 83, 91], in the sense that a subcubic algorithm for any of them implies a subcubic algorithm for all of them. When the graph sparsity is taken into account, these problems currently are no longer in a single class: many have $\tilde{O}(mn)$-time algorithms whereas finding minimum weight triangle and related problems have $\tilde{O}(m^{3/2})$-time algorithms. Most known fine-grained reductions between graph problems do not preserve the graph sparsity. Until recently, the only examples of sparseness-preserving truly subcubic reductions appeared in [3].[12] presented several more such reductions, strengthening the connections between problems with $\tilde{O}(mn)$-time algorithms. A reduction from CNF-SAT to Diameter was presented in [81] to give SETH-hardness results for Diameter and Eccentricities. The notion of a sub-mn time bound was formalized later, in [12], where is was observed that the reduction in [81] gives SETH-hardness for any sub-mn time bound for these problems.

Question: Is there a sparseness-preserving, $\tilde{O}(n^2)$ time reduction from undirected weighted All Nodes Shortest Cycles (ANSC) to APSP? Is there a sparseness-preserving, $\tilde{O}(m+n)$ time reduction from undirected Min-Wt-Cycle to either Radius or Eccentricities? Is it SETH-hard to find a sub-mn bound for Min-Wt-Cycle or an $O(n^2+sub-mn)$ bound on APSP?

[Contributed by Vijaya Ramachandran.]

6.1.0.11 Open Problem 11: Hardness for partially dynamic graph problems

Many results show hardness for fully-dynamic problems in graphs, but the techniques do not seem to extend well to amortized lower bounds in the incremental and decremental cases. [4, 60, 40, 70] give some initial results on incremental/decremental problems.

Question: Develop general techniques for showing amortized hardness of partially dynamic problems in graphs. One candidate problem is decremental single-source reachability. [36] shows that $\tilde{O}(m\sqrt{n})$ total time is sufficient. Is it necessary?

[Contributed by Søren Dahlgaard.]

6.1.0.12 Open Problem 12: Hardness of vertex connectivity

A connected undirected graph is k-vertex (resp. edge) connected if it remains connected after any set of at most k-1 vertices (edges) is removed from the graph. A strongly connected directed graph is k-vertex (edge) connected if it remains strongly connected after any set of at most k-1 vertices (edges) is removed from the graph. The vertex (edge) connectivity of a graph is the maximum value of k such that the graph is k-vertex (edge) connected.
The edge-connectivity λ of an undirected graph can be determined in time $O(mlog^2nlog^2logn)$ [59, 68], and for the specific case of directed graphs in time $O(\lambda mlog(n^2/m))$ [49]. In contrast, the vertex-connectivity κ can only be computed in time $O((n+min\{\kappa^{5/2}, \kappa n^{3/4}\})m)$ [50], where for undirected graphs m can be replaced by kn.

Question: To check k-vertex connectivity means to either confirm that $\kappa \geq k$ or to find a set of k-1 vertices that disconnects the graph. Even when k is constant, no $o(n^2)$ time (or $o(nm)$ time for directed graphs) algorithms are known for checking k-connectivity. Is there a conditional superlinear lower bound?

[Contributed by Veronika Loitzenbauer.]

6.1.0.13 Open Problem 13: Parity and mean-payoff games

Parity games, and their generalization mean-payoff games, are among the rare ”natural” problems in $NP \cap co-NP$(and in $UP \cap co-UP$[66]) for which no polynomial-time algorithm is known. Both parity games and mean-payoff games are 2-player games played by taking an infinite walk on a directed graph; one of the vertices is designated the start vertex. In parity games each vertex is labeled by an integer in $[0, c]$; in mean payoff games each edge is labeled by an integer in $[-W, W]$(in [69] you can see a description of the game). The algorithmic question is to decide, for each start vertex, which of the two players wins the game and to construct a corresponding winning strategy. Parity games can be reduced to mean-payoff games with $W = n^c$. Quasipolynomial $O(n^{logc})$ time algorithms for parity games were discovered [30, 67]. The best known algorithms for mean-payoff games run in pseudo-polynomial time $O(mnW)$[22] and randomized sub-exponential time $O(2^{\sqrt{nlogn}}logW)$[21].

Question: Is there a polynomial-time algorithm for parity or mean-payoff games? Are there conditional superlinear lower bounds on these problems?

[Contributed by Veronika Loitzenbauer.]

6.1.0.14 Open Problem 14: Unknotting

A knot is a closed, non-self-intersecting polygonal chain in $\mathbb{R}^3$. Two knots are equivalent if one can be continuously deformed into the other without self-intersection. The unknot problem is to decide if a knot is equivalent to one that is embeddable in the plane.
Knots can be represented combinatorially, by projecting the polygonal chain onto $\mathbb{R}^2$, placing a vertex wherever two edges intersect. The result is a 4-regular planar graph (possibly with loops and parallel edges) where each vertex carries a bit indicating which pair of edges is ”over” and which pair is ”under”. *Reidemeister* moves (a small set of transformations on the knot diagram) suffice to transform any knot diagram to one of its equivalent representations.
The complexity of unknot and related problems (e.g., are two knots equivalent?, can two knots simultaneously embedded in $\mathbb{R}^3$ be untangled?) are known to be in **NP** [58] and solvable in $2^O(n)$ time [58, 64].

Question: Given a plane knot diagram with n intersections, can unknot or knot-equivalence be solved in time near-linear in n? If not, are there conditional lower bounds that show even some polynomial hardness?

[Contributed by Seth Pettie.]

6.1.0.15 Open Problem 15: 3-Collinearity (general position testing)

A set S of n points in $\mathbb{R}^2$ is said to be in general position if there do not exist three points in S that lie on a line. The 3-Collinearity problem is to test whether S is in general position. The 3-Collinearity problem is known to be as hard as 3SUM, and an algorithm that runs in $O(n^2)$ time is known.

Question: The question is whether the $O(n^2)$ algorithm is optimal or whether it can be solved in $o(n^2)$. Recent subquadratic algorithms for 3SUM [[19, 44, 55, 65] indicate that polylogarithmic improvements should be possible. A related question is whether there is an $O(n^{2-\epsilon})$-depth decision tree for 3-Collinearity; see [65, 20].

[Contributed by Omer Gold.]

6.1.0.16 Open Problem 16: Element uniqueness in X + Y

Given two sets X and Y , each of n real numbers, determine whether all the elements of $X + Y = \{x + y|x \in X, y \in Y\}$ are distinct. A somewhat stronger variant of this problem is to sort X+Y
The decision tree complexity of sorting X+Y and Element Uniqueness in X + Y was shown to be $O(n^2)$ by [46].

Question: Can these problems can be solved in $o(n^2 logn)$ time, even for the special case X=Y?

[Contributed by Omer Gold.]

6.1.0.17 Open Problem 17: Histogram indexing

The *histogram* $\psi(T)$ of a string $T \in \Sigma^*$ is a $|\Sigma|$-length vector containing the number of occurrences of each letter in T. The histogram indexing problem (aka jumbled indexing) is to preprocess a string T to support the following query: given a histogram vector ψ, decide whether there is a substring T' of T such that $\psi(T') = \psi$.
The state-of-the-art algorithm for histogram indexing [32] preprocesses a binary text T in $O(n^{1.859})$ time and answers queries in $O(1)$ time. Over a d-letter alphabet the preprocessing and query times are $\tilde{O}(n^{2-\delta})$ and $\tilde{O}(n^{2/3+\delta(d+13)/6})$, for any $\delta \geq 0$. On the lower bound side [14, 56], the 3SUM conjecture implies that it is impossible to simultaneously improve $n^{2-\delta}$ preprocessing and $n^{\delta(d/2-1)}$ query time by polynomial factors, where $\delta \leq 2/(d-1)$ and $d \geq 3$.

Question: Are there any non-trivial lower bounds on histogram indexing when $d = 2$? Is it possible to close the gap between the lower and upper bounds in general, or to base the hardness off of a different conjecture than 3SUM?

[Contributed by Isaac Goldstein.]

6.1.0.18 Open Problem 18: Integer programming

The objective of Integer Programming (IP) is to decide, for a given $m \times n$ matrix A and an m-vector $b = (b_1, \ldots, b_m)$, whether there is a non-negative integer n-vector x such that $Ax = b$. In 1981, Papadimitriou [80] showed that (IP) is solvable in pseudo-polynomial time on instances for which the number of constraints m is constant. The rough estimation of the running time of Papadimitriou's algorithm is $n^{O(m)} d^{O(m^2)}$, where d bounds the magnitude of any entry in A and b. The best known lower bound is $n^{o(\frac{m}{logm})} d^{o(m)}$[44], assuming the Exponential Time Hypothesis (ETH).

Question: Is it possible to narrow the gap between algorithms for IP and the ETH-hardness of IP?

[Contributed by Fedor V. Fomin.]

6.1.0.19 Open Problem 19: All-pairs min-cut and generalizations

The all-pairs min-cut problem is, given an edge-capacitated undirected graph $G = (V, E, c)$ to compute the minimum s-t cut over all pairs $s, t \in V$. Gomory and Hu [86] showed the problem is reducible to $n-1$ s-t min-cut instances, and moreover, all $\binom{n}{2}$ min-cuts can be represented by a capacitated tree T on the vertex set V. On unweighted graphs, the construction of T takes time $\tilde{O}(mn)$[14, 60].
Generalizations of this problem include finding the min-cut separating every triple $r, s, t \in V^3$, which is NP-hard, and

finding the min-cuts separating all pairs of k-sets $\{s_1, \ldots, s_k\}$ from $\{t_1, \ldots, t_k\}$.[57, 37]

Question: Are there superlinear conditional lower bounds for all-pairs min-cut/Gomory-Hu tree construction? (Refer to [7] for conditional lower bounds for variants of the problem on directed graphs.) Are there non-trivial conditional lower bounds for all-triplets approximate min-cut, or all-k-sets min-cut?

[Contributed by Robert Krauthgamer.]

6.1.0.20 Open Problem 20: Parameterizing string algorithms by compressibility

The broad idea can be illustrated with a lower bound for string edit distance: Given two strings of length N whose compressed length (say, using Lempel-Ziv compression) is n, it is known that their edit distance can be computed in $O(nN)$ time. Is it possible to prove an $\Omega(nN)$ conditional lower bound? The known conditional lower bound [16, 1, 26] reduces CNF-SAT (with n variables) to string edit distance by creating two strings each consisting of $O(2^{n/2})$ blocks. To make such a reduction suitable for proving $\Omega(nN)$ lower bound, one needs to generate instead two strings whose length is much more than $2^{n/2}$ but that compress to much less than $2^{n/2}$.

[Contributed by Oren Weimann.]

6.1.0.21 Open Problem 21: Reductions from low complexity to high complexity

We know that improving the runtime of our 10-Clique algorithms improves the runtime of our 100-Clique algorithms. E.g., if 10-Clique can be solved in $O(n^5)$, then 100-Clique can be solved in $O(n^{50})$. In general, we have many examples of reductions showing that a faster algorithm for a problem with best known runtime $O(n^a)$, implies a faster algorithm for a problem with runtime $O(n^b)$, where $a \leq b$.
However, we have no interesting reductions in the other way, showing that improvements over $O(n^b)$ imply improvements over $O(n^a)$, where $a < b$. In particular, we do not know how to use an algorithm that solved 100-Clique in $O(n^{50})$ or even $O(n^{11})$

time, to speed up the known algorithms for 10-Clique.
Could it be that such reductions, from low complexity to high complexity, do not exist? It is not hard to construct artificial problems where this can be done, but what about the natural problems we typically study: Clique, Orthogonal Vectors, k-SUM, APSP, LCS, etc. Can we show that a fine-grained reduction from 10-Clique to 100-Clique is unlikely due to some surprising consequences? Another candidate is 3SUM (for which the complexity is n^2) vs. APSP (for which the complexity is $N^{1.5}$, where N is the input size). We repeatedly ask if faster 3SUM implies faster APSP, but maybe proving such a result (via fine-grained reductions) has unexpected consequences? On the other hand, it would be of great interest to find examples of such reductions between interesting and natural problems.

[Contributed by Amir Abboud.]

Note: This problem is one that interests me greatly, and as such I mention it in the respective chapter regarding Information Theoretic Bounds. In short, I believe these questions can be answered via Kolmogorov Complexity

6.1.0.22 Open Problem 22: Stable matching in the two-list model

Gale and Shapley's stable matching [54] algorithm runs in $O(n^2)$ time and it is known that $\Omega(n^2)$ is optimal if the preference lists are arbitrary. Moeller, Paturi, and Schneider [78] studied the complexity of stable matching when the preference lists are constrained, and encoded in some succinct manner. Many succinct input models nonetheless require $n^{2-o(1)}$ time, conditioned on SETH.

Question: A problem left open by [78] is two-list stable matching. A matching market in the two-list model consists of two sets M and W, both of size n, and permutations π_1, π_2 on M and σ_1, σ_2 on W. The preference list of each agent $m \in M$ is either σ_1 or σ_2 and the preference list of each agent $w \in W$ is either π_1 or π_2. The input size is $O(n)$.The goal is to find a stable matching in the resulting matching market. Can this problem be solved in linear time, or is there a superlinear conditional lower bound?

[Contributed by Stefan Schneider.]

6.1.0.23 Open Problem 23: Boolean vs. real maximum inner product

In the maximum inner product problem we are given two sets of d-dimensional vectors U and V of size n as well as a threshold l. The problem is to decide if there is a pair $u \in U, v \in V$ such that their inner product $u \cdot v$ is at least l. If the vectors are Boolean, then a randomized algorithm by Alman and Williams [13] solves the problem in time $n^{2-1/\Theta(clog^2c)}$ where $d = clogn$. In contrast, if the vectors are real or integer, then using ray-shooting techniques [77] we can solve the problem in time $n^{2-1/\Theta(d)}$. This leaves a large gap between the two problems. In particular, the Boolean case is strongly subquadratic if $d = O(logn)$, while the real case is only strongly subquadratic for constant d. The conditional lower bounds of [13] show that any $n^{2-\epsilon}$ algorithm when $d = \omega(logn)$ refutes SETH.

Question: Can the gap between the boolean and integer/real case be closed, with a better maximum inner product algorithm? If the gap is natural, can it be explained with a stronger conditional lower bound on (real or integer) maximum inner product?

[Contributed by Stefan Schneider.]

6.1.0.24 Open Problem 24: Hardness of Approximating NP-hard Problems

Many approximation algorithms for NP-hard problems run in polynomial time, but not linear time. This is often due to the use of general LP or SDP solvers, but not always. To take two examples, the chromatic index (edge coloring) and minimum degree spanning tree problems are NP-hard, but can both be approximated to within 1 of optimal in $\tilde{O}(m\sqrt{n})$ time [51] and $\tilde{O}(mn)$ time [48], respectively.

Question: Prove superlinear conditional lower bounds on the time complexity of any approximation problem, whose exact version is NP-hard.

[Contributed by Seth Pettie.]

6.1.0.25 Open Problem 25: Chromatic index/edge coloring

The chromatic index of a graph is the least number of colors needed for a proper edge-coloring. Vizing's theorem implies that the chromatic index is either Δ or $\Delta + 1$ (where Δ is the maximum degree), but determining which one is NP-hard. The NP-hardness reduction of Holyer [61] reduces 3SAT to a 3-regular graph on $O(n)$ vertices, so the ETH implies a $2^{\Omega(n)}$ lower bound. There is an $O^*(2^m)$ algorithm for chromatic index, by reduction to vertex coloring, so the hardness is well understood when $m = O(n)$.

Question: Does the ETH rule out a $2^o(m)$ algorithm for chromatic index on dense graphs? Is there, for example, an $n^{O(n)}$ or $2^{n^{2-\epsilon}}$-time algorithm?

[Contributed by Marek Cygan.]

6.1.0.26 Open Problem 26: Communication Complexity of Approximate Hamming Distance

Consider strings P of length n and T of length $2n$. Alice has the whole of P and the first half of T. That is she has P and $T[0,\ldots,n-1]$. Bob has the second half of T, that is $T[n,\ldots,2n-1]$. Alice sends one message to Bob and Bob has to output a $(1+\epsilon)$ multiplicative approximation of $HD(P,T[i,\ldots,i+n])$ for all $i \in [n]$ where HD is the Hamming Distance.
In [38] a $O(\sqrt{n}logn/\epsilon^2)$ bit communication protocol was given.

Question: Is there a matching lower bound for the randomized one-way communication complexity of this problem?

[Contributed by Raphaël Clifford]

7. CONCLUSION

One of the main goals of complexity theory is to determine the worst case time complexity of fundamental computational problems. When considering a computational problem, as a first step we decide on a computational model, such as a Random Access Machine (RAM) or a Turing machine (TM). Following that, the goal is to develop an efficient algorithm that solves the problem and to prove that for a function f(n),the algorithm solves the problem on instances of size n in O(f(n)) time in the selected computational model. To solve most problems, one needs to at least read the input, which implies linear time. Over the years, the theory of algorithms has developed a wide variety of techniques and near-linear time algorithms for many diverse problems. Nevertheless, for most problems of interest, the fastest known algorithms run much slower than linear time. Furthermore, lower bounds are very challenging to derive and as a result the computer science community has resorted to lower bounds that are conditioned on plausible, but so far unproven hypotheses. The notion of reducibility is very important allowing inferences about problems being equivalent in complexity to each other, even if we cannot pinpoint what that complexity is. It is well known that in mathematics, as in everyday life, a typical way to solve a new problem is to reduce it to a previously solved problem.
In this manner, on the basis of a widely believed hypothesis about the time complexity of a key problem, fine-grained reductions are used to reduce this key problem to other important problems, giving conditional lower bounds on how fast these problems can be solved.

Despite the fact that its a relatively new field, Fine-Grained Complexity is already showing its importance and potential for giving answers to problems unsolved by other techniques. The general idea is by no means a new concept, as it mimics to a high extent the notion of NP-completeness and the way reductions were used traditionally. However, the added condition of respecting the resources used in the reduction, and being precise with the computations, has given the field a new power that was unexpected. The initial goal of the field was to show hardness within polynomial bounds, which has historically been one of the less discovered areas of Computer Science. Many problems have become stuck in undiscovered deadlocks, where algorithmic design cannot give a better algorithm for a problem, but there is also no proof that they can't be solved faster than the current best algorithm. The main concept is to use these reductions to bind problems with each other into families of problems, and in a way ”transfer” the confidence about the hardness of each problem into the whole family of problems that are reduced to one another. Using some problems that have been very widely studied as ”heads” of these families gives even further confidence as to whether these problems will find improvements or not. Various studied and interesting implications of falsifying a conjecture associated with these families are carried through the reductions into the whole family, magnifying the impact of results in any of the problems involved. Having SETH, OVH,3SUMH and APSPH as the cornerstone of these reductions, these reductions are continuously flourishing, to the point that they produce a whole web of reductions clustering many important algorithmic problems into such families.

However, my viewpoint is that Fine-Grained complexity can also be used as a tool to unveil structural properties of problems. The reductions have some properties inherent in their definition that have been undiscovered. One can speculate about limitless properties and links in the structure of the problems that are reduced to one another, as well as the structure of the web of reductions itself. It is evident to me that such a study would doubtlessly resolve into unlocking even more structural properties of problems, and possibly even create mathematical tools that will help solve even the greatest mysteries of Algorithms, Complexity and Computer Science as a whole. Furthermore, it is not unreasonable to say that concepts such as entropy and chaos theory will also see the effect of these results.

Finally, the main appeal of Fine-Grained Complexity is that in encapsulates all of the fields of Algorithmic design and Mathematics, ranging from Computational Complexity to Graph Theory, Logic, and Dynamic Programming. Usually the way in which a field can encapsulate such diversity is by magnifying the abstract features, which results into almost philosophical statements that are incredibly hard to prove and/or put to practical use. Fine-Grained Complexity does this in a way that involves the whole spectrum of practice to theory, stating in this way the argument that there exist universal truths that affect every specialization of Computer Science. I believe that this field will grow to encapsulate even more aspects of Computer Science in the future, in a way that will change the viewpoint of every sub-field involved.

REFERENCES

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. “Tight Hardness Results for LCS and Other Sequence Similarity Measures”. In: *IEEE 56th Annual Symposium on Foundations of Computer Science FOCS, 2015, Berkeley, CA, USA, 17-20 October, 2015*. 2015, pp. 59–78. DOI: 10.1109/FOCS.2015.14. URL: https://doi.org/10.1109/FOCS.2015.14.

[2] Amir Abboud and Søren Dahlgaard. “Popular Conjectures as a Barrier for Dynamic Planar Graph Algorithms”. In: *CoRR* abs/1605.03797 (2016). arXiv: 1605.03797. URL: http://arxiv.org/abs/1605.03797.

[3] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. “Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter”. In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*. 2015, pp. 1681–1697. DOI: 10.1137/1.9781611973730.112. URL: https://doi.org/10.1137/1.9781611973730.112.

[4] Amir Abboud and Virginia Vassilevska Williams. “Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems”. In: *55th IEEE Annual Symposium on Foundations of Computer Science FOCS, 2014, Philadelphia, PA, USA, October 18-21, 2014*. 2014, pp. 434–443. DOI: 10.1109/FOCS.2014.53. URL: https://doi.org/10.1109/FOCS.2014.53.

[5] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. “Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs”. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 2016, pp. 377–391. DOI: 10.1137/1.9781611974331.ch28. URL: https://doi.org/10.1137/1.9781611974331.ch28.

[6] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. “Consequences of Faster Alignment of Sequences”. In: *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*. 2014, pp. 39–51. DOI: 10.1007/978-3-662-43948-7_4. URL: https://doi.org/10.1007/978-3-662-43948-7_4.

[7] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. “Matching Triangles and Basing Hardness on an Extremely Popular Conjecture”. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 41–50. DOI: 10.1145/2746539.2746594. URL: http://doi.acm.org/10.1145/2746539.2746594.

[8] Amir Abboud et al. “Fine-Grained Complexity of Analyzing Compressed Data: Quantifying Improvements over Decompress-and-Solve”. In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 192–203. DOI: 10.1109/FOCS.2017.26. URL: https://doi.org/10.1109/FOCS.2017.26.

[9] Amir Abboud et al. “SETH-Based Lower Bounds for Subset Sum and Bicriteria Path”. In: *CoRR* abs/1704.04546 (2017). arXiv: 1704.04546. URL: http://arxiv.org/abs/1704.04546.

[10] Amir Abboud et al. “Subtree Isomorphism Revisited”. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 2016, pp. 1256–1271. DOI: 10.1137/1.9781611974331.ch88. URL: https://doi.org/10.1137/1.9781611974331.ch88.

[11] Ittai Abraham, Shiri Chechik, and Sebastian Krinninger. “Fully Dynamic All-pairs Shortest Paths with Worst-case Update-time Revisited”. In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’17. Barcelona, Spain: Society for Industrial and Applied Mathematics, 2017, pp. 440–452. URL: http://dl.acm.org/citation.cfm?id=3039686.3039714.

[12] Udit Agarwal and Vijaya Ramachandran. “Fine-Grained Complexity and Conditional Hardness for Sparse Graphs”. In: *CoRR* abs/1611.07008 (2016). arXiv: 1611.07008. URL: http://arxiv.org/abs/1611.07008.

[13] Josh Alman and Ryan Williams. “Probabilistic Polynomials and Hamming Nearest Neighbors”. In: *IEEE 56th Annual Symposium on Foundations of Computer Science FOCS, 2015, Berkeley, CA, USA, 17-20 October, 2015*. 2015, pp. 136–150. DOI: 10.1109/FOCS.2015.18. URL: https://doi.org/10.1109/FOCS.2015.18.

[14] Amihood Amir et al. “On Hardness of Jumbled Indexing”. In: *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*. 2014, pp. 114–125. DOI: 10.1007/978-3-662-43948-7_10. URL: https://doi.org/10.1007/978-3-662-43948-7_10.

[15] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. “Tight Hardness Results for Maximum Weight Rectangles”. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016, 81:1–81:13. DOI: 10.4230/LIPIcs.ICALP.2016.81. URL: https://doi.org/10.4230/LIPIcs.ICALP.2016.81.

[16] Arturs Backurs and Piotr Indyk. “Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)”. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 51–58. DOI: 10.1145/2746539.2746612. URL: http://doi.acm.org/10.1145/2746539.2746612.

[17] Arturs Backurs and Piotr Indyk. “Which Regular Expression Patterns Are Hard to Match?” In: *IEEE 57th Annual Symposium on Foundations of Computer Science FOCS, 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. 2016, pp. 457–466. DOI: 10.1109/FOCS.2016.56. URL: https://doi.org/10.1109/FOCS.2016.56.

[18] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. “On the Fine-Grained Complexity of Empirical Risk Minimization: Kernel Methods and Neural Networks”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 4311–4321. URL: http://papers.nips.cc/paper/7018-on-the-fine-grained-complexity-of-empirical-risk-minimization-kernel-methods-and-neural-networks.

[19] Ilya Baran, Erik D. Demaine, and Mihai Patrascu. “Subquadratic Algorithms for 3SUM”. In: *Algorithmica* 50.4 (2008), pp. 584–596. DOI: 10.1007/s00453-007-9036-3. URL: https://doi.org/10.1007/s00453-007-9036-3.

[20] Luis Barba et al. “Subquadratic Algorithms for Algebraic Generalizations of 3SUM”. In: *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*. 2017, 13:1–13:15. DOI: 10.4230/LIPIcs.SoCG.2017.13. URL: https://doi.org/10.4230/LIPIcs.SoCG.2017.13.

[21] Henrik Björklund and Sergei G. Vorobyov. “A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games”. In: *Discrete Applied Mathematics* 155.2 (2007), pp. 210–229. DOI: 10.1016/j.dam.2006.04.029. URL: https://doi.org/10.1016/j.dam.2006.04.029.

[22] Lubos Brim et al. “Faster algorithms for mean-payoff games”. In: *Formal Methods in System Design* 38.2 (2011), pp. 97–118. DOI: 10.1007/s10703-010-0105-x. URL: https://doi.org/10.1007/s10703-010-0105-x.

[23] Karl Bringmann. “Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails”. In: *55th IEEE Annual Symposium on Foundations of Computer Science FOCS, 2014, Philadelphia, PA, USA, October 18-21, 2014*. 2014, pp. 661–670. DOI: 10.1109/FOCS.2014.76. URL: https://doi.org/10.1109/FOCS.2014.76.

[24] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. “A Dichotomy for Regular Expression Membership Testing”. In: *58th IEEE Annual Symposium on Foundations of Computer Science FOCS, 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 307–318. DOI: 10.1109/FOCS.2017.36. URL: https://doi.org/10.1109/FOCS.2017.36.

[25] Karl Bringmann and Marvin Künnemann. “Multivariate Fine-Grained Complexity of Longest Common Subsequence”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 2018, pp. 1216–1235. DOI: 10.1137/1.9781611975031.79. URL: https://doi.org/10.1137/1.9781611975031.79.

[26] Karl Bringmann and Marvin Künnemann. “Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping”. In: *IEEE 56th Annual Symposium on Foundations of Computer Science FOCS, 2015, Berkeley, CA, USA, 17-20 October, 2015*. 2015, pp. 79–97. DOI: 10.1109/FOCS.2015.15. URL: https://doi.org/10.1109/FOCS.2015.15.

[27] Karl Bringmann et al. “Tree Edit Distance Cannot be Computed in Strongly Subcubic Time (unless APSP can)”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 2018, pp. 1190–1206. DOI: 10.1137/1.9781611975031.77. URL: https://doi.org/10.1137/1.9781611975031.77.

[28] Sergio Cabello. “Subquadratic Algorithms for the Diameter and the Sum of Pairwise Distances in Planar Graphs”. In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’17. Barcelona, Spain: Society for Industrial and Applied Mathematics, 2017, pp. 2143–2152. URL: http://dl.acm.org/citation.cfm?id=3039686.3039825.

[29] Massimo Cairo, Roberto Grossi, and Romeo Rizzi. “New Bounds for Approximating Extremal Distances in Undirected Graphs”. In: *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’16. Arlington, Virginia: Society for Industrial and Applied Mathematics, 2016, pp. 363–376. ISBN: 978-1-611974-33-1. URL: http://dl.acm.org/citation.cfm?id=2884435.2884462.

[30] Cristian S. Calude et al. “Deciding parity games in quasipolynomial time”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 2017, pp. 252–263. DOI: 10.1145/3055399.3055409. URL: http://doi.acm.org/10.1145/3055399.3055409.

[31] Marco Carmosino et al. “Nondeterministic extensions of the Strong Exponential Time Hypothesis and consequences for non-reducibility”. In: *Electronic Colloquium on Computational Complexity (ECCC)* 22 (2015), p. 148. URL: http://eccc.hpi-web.de/report/2015/148.

[32] Timothy M. Chan and Moshe Lewenstein. “Clustered Integer 3SUM via Additive Combinatorics”. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 31–40. DOI: 10.1145/2746539.2746568. URL: http://doi.acm.org/10.1145/2746539.2746568.

[33] Krishnendu Chatterjee et al. “Conditionally Optimal Algorithms for Generalized Büchi Games”. In: *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*. 2016, 25:1–25:15. DOI: 10.4230/LIPIcs.MFCS.2016.25. URL: https://doi.org/10.4230/LIPIcs.MFCS.2016.25.

[34] Krishnendu Chatterjee et al. “Model and Objective Separation with Conditional Lower Bounds: Disjunction is Harder than Conjunction”. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS ’16, New York, NY, USA, July 5-8, 2016*. 2016, pp. 197–206. DOI: 10.1145/2933575.2935304. URL: http://doi.acm.org/10.1145/2933575.2935304.

[35] Shiri Chechik et al. “Better Approximation Algorithms for the Graph Diameter”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 2014, pp. 1041–1052. DOI: 10.1137/1.9781611973402.78. URL: https://doi.org/10.1137/1.9781611973402.78.

[36] Shiri Chechik et al. “Decremental Single-Source Reachability and Strongly Connected Components in $\tilde{O}(m\sqrt{n})$ Total Update Time”. In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. 2016, pp. 315–324. DOI: 10.1109/FOCS.2016.42. URL: https://doi.org/10.1109/FOCS.2016.42.

[37] Rajesh Chitnis, Lior Kamma, and Robert Krauthgamer. “Tight Bounds for Gomory-Hu-like Cut Counting”. In: *Graph-Theoretic Concepts in Computer Science - 42nd International Workshop, WG 2016, Istanbul, Turkey, June 22-24, 2016, Revised Selected Papers*. 2016, pp. 133–144. DOI: 10.1007/978-3-662-53536-3_12. URL: https://doi.org/10.1007/978-3-662-53536-3_12.

[38] Raphaël Clifford and Tatiana A. Starikovskaya. “Approximate Hamming Distance in a Stream”. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016, 20:1–20:14. DOI: 10.4230/LIPIcs.ICALP.2016.20. URL: https://doi.org/10.4230/LIPIcs.ICALP.2016.20.

[39] Marek Cygan et al. *Parameterized Algorithms*. 1st. Springer Publishing Company, Incorporated, 2015. ISBN: 3319212745, 9783319212746.

[40] Søren Dahlgaard. “On the Hardness of Partially Dynamic Graph Problems and Connections to Diameter”. In: *CoRR* abs/1602.06705 (2016). arXiv: 1602.06705. URL: http://arxiv.org/abs/1602.06705.

[41] Camil Demetrescu and Giuseppe F. Italiano. “A New Approach to Dynamic All Pairs Shortest Paths”. In: *J. ACM* 51.6 (Nov. 2004), pp. 968–992. ISSN: 0004-5411. DOI: 10.1145/1039488.1039492. URL: http://doi.acm.org/10.1145/1039488.1039492.

[42] Krzysztof Diks and Piotr Sankowski. “Dynamic Plane Transitive Closure”. In: *Proceedings of the 15th Annual European Conference on Algorithms*. ESA’07. Eilat, Israel: Springer-Verlag, 2007, pp. 594–604. ISBN: 3-540-75519-5, 978-3-540-75519-7. URL: http://dl.acm.org/citation.cfm?id=1778580.1778635.

[43] Dorit Dor, Shay Halperin, and Uri Zwick. “All-Pairs Almost Shortest Paths”. In: *SIAM J. Comput.* 29.5 (Mar. 2000), pp. 1740–1759. ISSN: 0097-5397. DOI: 10.1137/S0097539797327908. URL: http://dx.doi.org/10.1137/S0097539797327908.

[44] Fedor V. Fomin et al. “Fine-grained complexity of integer programming: The case of bounded branch-width and rank”. In: *CoRR* abs/1607.05342 (2016). arXiv: 1607.05342. URL: http://arxiv.org/abs/1607.05342.

[45] Fedor V. Fomin et al. “Fully polynomial-time parameterized computations for graphs and matrices of low treewidth”. In: *CoRR* abs/1511.01379 (2015). arXiv: 1511.01379. URL: http://arxiv.org/abs/1511.01379.

[46] Michael L. Fredman. “How Good is the Information Theory Bound in Sorting?” In: *Theor. Comput. Sci.* 1.4 (1976), pp. 355–361. DOI: 10.1016/0304-3975(76)90078-5. URL: https://doi.org/10.1016/0304-3975(76)90078-5.

[47] Ari Freund. “Improved Subquadratic 3SUM”. In: *Algorithmica* 77.2 (Feb. 2017), pp. 440–458. ISSN: 0178-4617. DOI: 10.1007/s00453-015-0079-6. URL: https://doi.org/10.1007/s00453-015-0079-6.

[48] Martin Fürer and Balaji Raghavachari. “Approximating the Minimum-Degree Steiner Tree to within One of Optimal”. In: *J. Algorithms* 17.3 (1994), pp. 409–423. DOI: 10.1006/jagm.1994.1042. URL: https://doi.org/10.1006/jagm.1994.1042.

[49] Harold N. Gabow. “A Matroid Approach to Finding Edge Connectivity and Packing Arborescences”. In: *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*. STOC ’91. New Orleans, Louisiana, USA: ACM, 1991, pp. 112–122. ISBN: 0-89791-397-3. DOI: 10.1145/103418.103436. URL: http://doi.acm.org/10.1145/103418.103436.

[50] Harold N. Gabow. “Using Expander Graphs to Find Vertex Connectivity”. In: *J. ACM* 53.5 (Sept. 2006), pp. 800–844. ISSN: 0004-5411. DOI: 10.1145/1183907.1183912. URL: http://doi.acm.org/10.1145/1183907.1183912.

[51] Harold N. Gabow and Oded Kariv. “Algorithms for Edge Coloring Bipartite Graphs”. In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*. 1978, pp. 184–192. DOI: 10.1145/800133.804346. URL: http://doi.acm.org/10.1145/800133.804346.

[52] Anka Gajentaan and Mark H. Overmars. “On a class of O(n^2) problems in computational geometry”. In: *Comput. Geom.* 45.4 (2012), pp. 140–152. DOI: 10.1016/j.comgeo.2011.11.006. URL: https://doi.org/10.1016/j.comgeo.2011.11.006.

[53] Anka Gajentaan and Mark H. Overmars. “On a Class of O(n2) Problems in Computational Geometry”. In: *Comput. Geom.* 5 (1995), pp. 165–185. DOI: 10.1016/0925-7721(95)00022-2. URL: https://doi.org/10.1016/0925-7721(95)00022-2.

[54] D. Gale and L. S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 120.5 (2013), pp. 386–391. DOI: 10.4169/amer.math.monthly.120.05.386. URL: https://doi.org/10.4169/amer.math.monthly.120.05.386.

[55] Omer Gold and Micha Sharir. “Improved Bounds for 3SUM, K-SUM, and Linear Degeneracy”. In: *CoRR* abs/1512.05279 (2015). arXiv: 1512.05279. URL: http://arxiv.org/abs/1512.05279.

[56] Isaac Goldstein et al. “How Hard is it to Find (Honest) Witnesses?” In: *CoRR* abs/1706.05815 (2017). arXiv: 1706.05815. URL: http://arxiv.org/abs/1706.05815.

[57] Ramesh Hariharan et al. “An Õ(mn) Gomory-Hu tree construction algorithm for unweighted graphs”. In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*. 2007, pp. 605–614. DOI: 10.1145/1250790.1250879. URL: http://doi.acm.org/10.1145/1250790.1250879.

[58] Joel Hass, J. C. Lagarias, and Nicholas Pippenger. “The Computational Complexity of Knot and Link Problems”. In: *J. ACM* 46.2 (1999), pp. 185–211. DOI: 10.1145/301970.301971. URL: http://doi.acm.org/10.1145/301970.301971.

[59] Monika Henzinger, Satish Rao, and Di Wang. “Local Flow Partitioning for Faster Edge Connectivity”. In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’17. Barcelona, Spain: Society for Industrial and Applied Mathematics, 2017, pp. 1919–1938. URL: http://dl.acm.org/citation.cfm?id=3039686.3039811.

[60] Monika Henzinger et al. “Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture”. In: *CoRR* abs/1511.06773 (2015). arXiv: 1511.06773. URL: http://arxiv.org/abs/1511.06773.

[61] Ian Holyer. “The NP-Completeness of Edge-Coloring”. In: *SIAM J. Comput.* 10.4 (1981), pp. 718–720. DOI: 10.1137/0210055. URL: https://doi.org/10.1137/0210055.

[62] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. “Which Problems Have Strongly Exponential Complexity?” In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 512–530. DOI: 10.1006/jcss.2001.1774. URL: https://doi.org/10.1006/jcss.2001.1774.

[63] Alon Itai and Michael Rodeh. “Finding a Minimum Circuit in a Graph”. In: *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*. STOC ’77. Boulder, Colorado, USA: ACM, 1977, pp. 1–10. DOI: 10.1145/800105.803390. URL: http://doi.acm.org/10.1145/800105.803390.

[64] William Jaco and Jeffrey L. Tollefson. “Algorithms for the complete decomposition of a closed 3-manifold”. In: *Illinois J. Math.* 39.3 (Sept. 1995), pp. 358–406. URL: https://projecteuclid.org:443/euclid.ijm/1255986385.

[65] Allan Grønlund Jørgensen and Seth Pettie. “Threesomes, Degenerates, and Love Triangles”. In: *CoRR* abs/1404.0799 (2014). arXiv: 1404.0799. URL: http://arxiv.org/abs/1404.0799.

[66] Marcin Jurdzinski. “Deciding the Winner in Parity Games is in UP \cap co-Up”. In: *Inf. Process. Lett.* 68.3 (1998), pp. 119–124. DOI: 10.1016/S0020-0190(98)00150-1. URL: https://doi.org/10.1016/S0020-0190(98)00150-1.

[67] Marcin Jurdzinski and Ranko Lazic. “Succinct progress measures for solving parity games”. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. 2017, pp. 1–9. DOI: 10.1109/LICS.2017.8005092. URL: https://doi.org/10.1109/LICS.2017.8005092.

[68] Ken-ichi Kawarabayashi and Mikkel Thorup. “Deterministic Global Minimum Cut of a Simple Graph in Near-Linear Time”. In: *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*. STOC ’15. Portland, Oregon, USA: ACM, 2015, pp. 665–674. ISBN: 978-1-4503-3536-2. DOI: 10.1145/2746539.2746588. URL: http://doi.acm.org/10.1145/2746539.2746588.

[69] Ken-ichi Kawarabayashi and Mikkel Thorup. “Deterministic Global Minimum Cut of a Simple Graph in Near-Linear Time”. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 665–674. DOI: 10.1145/2746539.2746588. URL: http://doi.acm.org/10.1145/2746539.2746588.

[70] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. “Higher Lower Bounds from the 3SUM Conjecture”. In: *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’16. Arlington, Virginia: Society for Industrial and Applied Mathematics, 2016, pp. 1272–1287. ISBN: 978-1-611974-33-1. URL: http://dl.acm.org/citation.cfm?id=2884435.2884524.

[71] Robert Krauthgamer and Ohad Trabelsi. “Conditional Lower Bounds for All-Pairs Max-Flow”. In: *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 2017, 20:1–20:13. DOI: 10.4230/LIPIcs.ICALP.2017.20. URL: https://doi.org/10.4230/LIPIcs.ICALP.2017.20.

[72] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. “On the Fine-Grained Complexity of One-Dimensional Dynamic Programming”. In: *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 2017, 21:1–21:15. DOI: 10.4230/LIPIcs.ICALP.2017.21. URL: https://doi.org/10.4230/LIPIcs.ICALP.2017.21.

[73] Jakub Lacki et al. “Single Source - All Sinks Max Flows in Planar Digraphs”. In: *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*. 2012, pp. 599–608. DOI: 10.1109/FOCS.2012.66. URL: https://doi.org/10.1109/FOCS.2012.66.

[74] Yin Tat Lee and Aaron Sidford. “Efficient Inverse Maintenance and Faster Algorithms for Linear Programming”. In: *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*. FOCS ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 230–249. ISBN: 978-1-4673-8191-8. DOI: 10.1109/FOCS.2015.23. URL: http://dx.doi.org/10.1109/FOCS.2015.23.

[75] Yin Tat Lee and Aaron Sidford. “Path Finding Methods for Linear Programming: Solving Linear Programs in Õ(vrank) Iterations and Faster Algorithms for Maximum Flow”. In: *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. 2014, pp. 424–433. DOI: 10.1109/FOCS.2014.52. URL: https://doi.org/10.1109/FOCS.2014.52.

[76] Moshe Lewenstein, Seth Pettie, and Virginia Vassilevska Williams. “Structure and Hardness in P (Dagstuhl Seminar 16451)”. In: *Dagstuhl Reports* 6.11 (2017). Ed. by Moshe Lewenstein, Seth Pettie, and Virginia Vassilevska Williams, pp. 1–34. ISSN: 2192-5283. DOI: 10.4230/DagRep.6.11.1. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7037.

[77] Jiří Matoušek. “Efficient partition trees”. In: *Discrete & Computational Geometry* 8.3 (Sept. 1992), pp. 315–334. ISSN: 1432-0444. DOI: 10.1007/BF02293051. URL: https://doi.org/10.1007/BF02293051.

[78] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. “Subquadratic Algorithms for Succinct Stable Matching”. In: *Computer Science - Theory and Applications - 11th International Computer Science Symposium in Russia, CSR 2016, St. Petersburg, Russia, June 9-13, 2016, Proceedings*. 2016, pp. 294–308. DOI: 10.1007/978-3-319-34171-2_21. URL: https://doi.org/10.1007/978-3-319-34171-2_21.

[79] Jakub Pachocki et al. “Approximating Cycles in Directed Graphs: Fast Algorithms for Girth and Roundtrip Spanners”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, pp. 1374–1392. ISBN: 978-1-6119-7503-1. URL: http://dl.acm.org/citation.cfm?id=3174304.3175396.

[80] Christos H. Papadimitriou. “On the complexity of integer programming”. In: *J. ACM* 28.4 (1981), pp. 765–768. DOI: 10.1145/322276.322287. URL: http://doi.acm.org/10.1145/322276.322287.

[81] Liam Roditty and Virginia Vassilevska Williams. “Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs”. In: *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*. STOC ’13. Palo Alto, California, USA: ACM, 2013, pp. 515–524. ISBN: 978-1-4503-2029-0. DOI: 10.1145/2488608.2488673. URL: http://doi.acm.org/10.1145/2488608.2488673.

[82] Liam Roditty and Virginia Vassilevska Williams. “Subquadratic Time Approximation Algorithms for the Girth”. In: *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’12. Kyoto, Japan: Society for Industrial and Applied Mathematics, 2012, pp. 833–845. URL: http://dl.acm.org/citation.cfm?id=2095116.2095183.

[83] Barna Saha. “Language Edit Distance and Maximum Likelihood Parsing of Stochastic Grammars: Faster Algorithms and Connection to Fundamental Graph Problems”. In: *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*. FOCS ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 118–135. ISBN: 978-1-4673-8191-8. DOI: 10.1109/FOCS.2015.17. URL: http://dx.doi.org/10.1109/FOCS.2015.17.

[84] Mikkel Thorup. “Worst-case Update Times for Fully-dynamic All-pairs Shortest Paths”. In: *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*. STOC ’05. Baltimore, MD, USA: ACM, 2005, pp. 112–119. ISBN: 1-58113-960-8. DOI: 10.1145/1060590.1060607. URL: http://doi.acm.org/10.1145/1060590.1060607.

[85] Virginia Vassilevska and Ryan Williams. “Finding, minimizing, and counting weighted subgraphs”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. 2009, pp. 455–464. DOI: 10.1145/1536414.1536477. URL: http://doi.acm.org/10.1145/1536414.1536477.

[86] “Volume Information”. In: *Journal of the Society for Industrial and Applied Mathematics* 9.4 (1961). ISSN: 03684245. URL: http://www.jstor.org/stable/2098875.

[87] Ryan Williams. “A new algorithm for optimal 2-constraint satisfaction and its implications”. In: *Theor. Comput. Sci.* 348.2-3 (2005), pp. 357–365. DOI: 10.1016/j.tcs.2005.09.023. URL: https://doi.org/10.1016/j.tcs.2005.09.023.

[88] Ryan Williams. “Faster all-pairs shortest paths via circuit complexity”. In: *CoRR* abs/1312.6680 (2013). arXiv: 1312.6680. URL: http://arxiv.org/abs/1312.6680.

[89] Ryan Williams. “On the Difference Between Closest, Furthest, and Orthogonal Pairs: Nearly-Linear vs Barely-Subquadratic Complexity”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 2018, pp. 1207–1215. DOI: 10.1137/1.9781611975031.78. URL: https://doi.org/10.1137/1.9781611975031.78.

[90] Virginia Vassilevska Williams. “Fine-grained Algorithms and Complexity”. In: *21st International Conference on Database Theory (ICDT 2018)*. Ed. by Benny Kimelfeld and Yael Amsterdamer. Vol. 98. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 1:1–1:1. ISBN: 978-3-95977-063-7. DOI: 10.4230/LIPIcs.ICDT.2018.1. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8613.

[91] Virginia Vassilevska Williams and Ryan Williams. “Subcubic Equivalences between Path, Matrix and Triangle Problems”. In: *51th Annual IEEE Symposium on Foundations of Computer Science FOCS, 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. 2010, pp. 645–654. DOI: 10.1109/FOCS.2010.67. URL: https://doi.org/10.1109/FOCS.2010.67.

[92] Raphael Yuster and Uri Zwick. “Finding Even Cycles Even Faster”. In: *SIAM J. Discret. Math.* 10.2 (May 1997), pp. 209–222. ISSN: 0895-4801. DOI: 10.1137/S0895480194274133. URL: https://doi.org/10.1137/S0895480194274133.