Санкт-Петербургский государственный университет

ИСТОМИНА Александра Андреевна

Выпускная квалификационная работа

Условные нижние оценки для задачи поиска путей с контекстно-свободными ограничениями

Уровень образования: магистратура
Направление: 01.04.01 «Математика»
Основная образовательная программа: ВМ.5832.2021 «Современная математика»

Научный руководитель:
доцент кафедры
информатики СПбГУ,
кандидат физ.-мат. наук,
Григорьев Семён Вячеславович

Рецензент:
Associate Professor,
Computer Science Department,
University of Warwick
PhD of Mathematics
Dmitry Chistikov

Санкт-Петербург
2023

Saint-Petersburg State University

ISTOMINA Aleksandra

Graduation qualification thesis

Fine-grained reductions around CFL-reachability

Master's program
Specialization and code: 01.04.01 «Mathematics»
Cipher of the EP: BM.5832.2021 «Advanced mathematics»

Thesis supervisor:
Associate Professor,
Informatics Department, SPbU
PhD of Phys.-Math. Sc.
Semyon Grigorev

Reviewer:
Associate Professor,
Computer Science Department,
University of Warwick
PhD of Mathematics
Dmitry Chistikov

Saint-Petersburg
2023

Contents

1 Introduction

Context-free language (CFL) reachability is a framework for graph analysis which was introduced by Thomas Reps [31] and Mihalis Yannakakis [48] and allows one to specify path constraints in terms of context-free languages. CFL reachability finds application in such fields of research as static code analysis (e.g. type-based flow analysis [30] or points-to analysis [40, 39]), graph databases [48], bioinformatics [36].

CFL reachability problem is formulated as follows. One is given context-free language $\mathcal{L}$ over finite alphabet Σ and a directed graph $D = (V, E, L)$ which edges are labeled with the symbols from Σ. It is needed to decide whether there exist a path between a pair of vertices in D on which labels on the edges form a word from $\mathcal{L}$. There exist two possible variations of a problem, where the question of defined above reachability is asked for a selected pair (s, t) of vertices — s-t reachability, or for all pairs of them – all-pairs reachability. CFL reachability problem complexity is often measured by the number of vertices in the graph: $n = |V|$, while the grammar is supposed to be fixed. There are several cubic [48, 32] and slightly subcubic [13] (with time $\mathcal{O}(n^3/poly(\log n))$[1]) algorithms for all-pairs CFL reachability, so the problem clearly lies in complexity class **P**.

Fine-grained complexity theory studies the exact degree of the polynomial expressing the complexity of the problems. In respect to CFL reachability the big open question is whether a truly subcubic (with time $\tilde{\mathcal{O}}(n^{3-\epsilon})$[2]) algorithm exists. Another, equally interesting, question is whether the fine-grained complexity of s-t and all-pairs CFL reachability are the same.

One of the ways to answer the question about an exact complexity of a problem is to find a lower bound. It is hard to create unconditional lower bound at most of the times. At the moment for the most of the problems there are known only trivial lower bounds based on the necessity to read all the input and produce all the output. As an example of a non-trivial lower bound one can see $\mathcal{O}(n \log n)$ lower bound for sorting algorithms based on comparisons, but this case is the exception rather than the rule.

In fine-grained complexity the usual way to answer a question about an exact complexity of a problem is to create a conditional lower bound: the lower bound is true if some widely believed hypothesis is true. The most popular hypotheses in the field are stated for the SAT, 3SUM and APSP problems.

Conditional lower bounds are achieved via a fine-grained reduction which shows how to convert an algorithm for problem A into an algorithm for problem B. In this case fast enough algorithm for A can lead to breakthrough algorithm for B which is not believed to exist.

2 Problem statement

The aim of this work is to analyze CFL reachability problem from the fine-grained complexity point of view. In order to achieve the aim, the following objectives were set.

[1] Every logarithm in the paper is a logarithm with base 2.

[2] Here and throughout the paper $\tilde{\mathcal{O}}$ notation hides polylogarithmic factors, e.g. $\tilde{\mathcal{O}}(n^{3-\epsilon}) = \mathcal{O}(n^{3-\epsilon} \cdot poly(\log(n))$.

1. Study existing algorithms for CFL reachability and explore possible ways of their enhancement.
2. Study existing conditional lower bounds on the problem and their techniques.
3. Find new conditional lower bounds on CFL reachability (preferably based on SAT, 3SUM and APSP hypotheses) or present reasons of their non-existence.

The rest of the paper is organised as follows. We introduce the necessary definitions and basic algorithms in the field in Sec. 3. After that in Sec. 4 we structure the results in the area and present a map with connections between the problems. In Sec. 5 we present some lower bounds and argue on the possibility of some reductions. Sec. 6 is devoted to the faster algorithms for CFL reachability if path lengths are bounded by some value. In Sec. 7 we discuss a technique that may be useful in creating new reductions and its limitations.

3 Preliminaries

In this section we introduce the necessary definitions, discuss basic algorithms for the CFL reachability and recognition problems and give an example of a fine-grained reduction.

3.1 Definitions

CFL Reachability *Context-free grammar* (CFG) is a four-tuple $G = (N, \Sigma, P, S)$, where:

- N is a set of nonterminals,
- Σ is a set of terminals,
- P is a set of productions of the followings form: $A \to \alpha$, $\alpha \in (N \cup \Sigma)^*$
- and $S \in N$ is a starting nonterminal.

We call $|G| = |N| + |\Sigma| + |P|$ the *size of the grammar.*

We write $\alpha A \beta \Rightarrow \alpha\gamma\beta$ where $\alpha, \gamma, \beta \in (\Sigma \cup N)^*$ if $A \to \gamma$ is a production, we call it a *derivation step. Derivation* is a composition of $i \geq 0$ derivation steps and is represented as $\Rightarrow^*$. Denote a context-free language of words derived from the starting nonterminal as $\mathcal{L}(G)$, i.e. $\mathcal{L}(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$.

Grammar is said to be in *Chomsky-Normal Form (CNF)* if every production is in one of the following forms:

1. $A \to BC, B, C \in N$
2. $A \to a, a \in \Sigma$
3. $S \to \epsilon$, if S is not used in right-hand side of the productions

Every CFG can be transformed to CFG in CNF in polynomial time of its size [15]. Further in some algorithms we will need the input grammar be in CNF and we assume that it can be done in $poly(|G|)$ time as preprocessing without affecting the time complexity of the algorithm.

CFG recognition problem is to decide whether $w \in \mathcal{L}(G)$ given a CFG G and a string $w \in \Sigma^*$. This problem is closely related to *CFG parsing problem* where we want a possible

derivation sequence, if $w \in \mathcal{L}(G)$. It is known [33] that CFG recognition is as hard as CFG parsing up to logarithmic factors.

Let $D = (V, E, L)$ be a directed graph with n vertices which edges are labeled with symbols from $L \subseteq \Sigma$. Denote by $v \xrightarrow{a} u$ and edge from vertex v to vertex u in D labeled with symbol a. We call a path from vertex v to vertex u an *A-path* if concatenation of labels on that path is a word that can be derived from the nonterminal $A \in N$. We say that v is *A-reachable* from u (or that (u, v) is an A-reachable pairs of vertices) if there exists an A-path from u to v. In case if $A = S$ is the starting nonterminal we may also say that v is *$\mathcal{L}$-reachable* from u.

Context-free language (CFL) reachability problem [31] is to determine if there exists an S-path between some vertices. In *single source/single target (s-t)* CFL reachability there is one pair of vertices $s, t \in V(D)$ for which we want to determine the existence of an S-path from s to t. This variation of a problem is also called *on-demand problem*, because it can be useful in case of many reachability queries. On the other hand in *all-pairs* CFL reachability we are asked about S-path existence between all possible pairs of vertices in a graph, all at once.

Dyck-k reachability problem is a CFL reachability problem where G defines a Dyck language on k types of parentheses. The corresponding grammar is $G = (N, \Sigma, P, S)$, where:

- $N = \{S\}$
- $\Sigma = \{(_i,)_i\}, \forall i = 1, \ldots, k$
- productions rules are $S \to \epsilon | SS | (_1 S)_1 | \ldots | (_k S)_k$, where ϵ is the empty string.

Dyck language is known [26] to be the hardest among the context-free languages in a way that every other language can be seen as a combination of some Dyck-k with a regular language. Moreover s-t reachability problem can be transformed [35] into Dyck-2 s-t reachability problem in $\mathcal{O}(|E|)$ time if we suppose the size of the given grammar to be fixed (which is a traditional assumption).

The PDA Emptiness problem is closely connected to CFL reachability. We begin with the definitions used in it.

Pushdown Automaton (PDA) [35] is six $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Q_f)$ where:

- Q is a finite set of states
- Σ is a finite string alphabet
- Γ is a finite stack alphabet
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to Q \times \Gamma^*$ is the finite transition function
- q_0 is a start state
- $Q_f \subseteq Q$ is a set of final states

For some string w the PDA reads it starting in state q_0 and traverses through the states using transition function. String w is said to be accepted by A if after reading w PDA $\mathcal{A}$ stops in state q that lies in Q_f. $\mathcal{L}(\mathcal{A})$ is the language of words accepted by PDA $\mathcal{A}$. *PDA Emptiness* problem is the problem of determining by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A})$ empty.

Reachability in push-down systems is the problem analogous to CFL reachability problem in model checking. *A push-down system (PDS) [17]* $\mathcal{P}$ is a triple (Q, Γ, δ), where:

- $Q, |Q| = n$ is a finite set of control states
- Γ is a finite stack alphabet
- $\delta \subseteq Q \times \Gamma \times Q \times \Gamma^*$ is a transition relation.

A configuration of $\mathcal{P}$ is a pair $(q, w) \in Q \times \Gamma^*$, where q is a control state and w is the stack word. *Configuration graph* $\mathcal{G}_\mathcal{P}$ if a graph on configurations as vertices with edge relation defined as follows: edge $(q_1, w_1) \to (q_2, w_2)$ exists when $(q_1, \gamma, q_2, w) \in \delta$, where $w \in \Gamma^*$ and $\gamma \in \Gamma \cup \{\epsilon\}$ are such that either:

1. $w_1 = \gamma = \epsilon$ and $w_2 = w$, or
2. $\gamma \neq \epsilon$ and there exists $w' \in \Gamma^*$ such that $w_1 = \gamma w'$ and $w_2 = ww'$.

The (state) reachability problem for a PDS $\mathcal{P}$ asks, given two states $q_s, q_t \in Q$, to decide whether there exists a path $P : (q_s, \epsilon) \to^* (q_t, \epsilon)$ in $\mathcal{G}_\mathcal{P}$, where $\to^*$ denotes a path consisting of non-negative number of edges between the corresponding vertices. We analogously define *all-pairs PDS reachability*. The problem is called *sparse* if $|\delta| = \mathcal{O}(|Q|)$. We call *the stack depth* the maximum size of the stack that the system $\mathcal{P}$ can have. Below we will always work with systems that have stack depth no more than some k, we call k the *upper bound on a stack depth*.

Fine-grained reductions For problems P, Q and time bounds t_P, t_Q, a *fine-grained reduction* [9] from (P, t_P) to (Q, t_Q) is an algorithm that, given an instance I of P, computes an instance J of Q such that:

- I is a YES-instance of P if and only if J is a YES-instance of Q,
- for any $\epsilon > 0$ there is a $\delta > 0$ such that $t_Q(|J|)^{1-\epsilon} = \mathcal{O}(t_P(|I|)^{1-\delta})$,
- the running time of the reduction is $\mathcal{O}(t_P(|I|)^{1-\gamma})$ for some $\gamma > 0$.

3.2 Existing problems and hypotheses

Here we give definitions for several problems that are connected with CFL recognition and reachability.

Popular problems *Boolean satisfiability problem (SAT, k-SAT)* is to determine if there exists an interpretation of variables that satisfies a given Boolean formula on n variables written in k-conjunctive normal form, $k \geq 2$ (k-CNF). The hypotheses about SAT, that we are interested about, are:

SETH [21] which proposes that there is no $\epsilon > 0$ such that k-SAT can be solved in time $\mathcal{O}(2^{(1-\epsilon)n})$ for any k.

NSETH [11] which proposes that there is no $\epsilon > 0$ such that k-SAT can be solved co-nondeterministically in time $\mathcal{O}(2^{(1-\epsilon)n})$ for any k.

The *3SUM* problem [45] is as follows: given a set S of n integers from $\{-n^c, \ldots, n^c\}, c > 0$, determine whether there are a triple that sums in zero, i.e. $a, b, c \in S$ such that $a + b + c = 0$. 3SUM hypothesis states that there exists no algorithm that runs in time $\mathcal{O}(n^{2-\epsilon}), \epsilon > 0$ for this problem.

In *All-Pairs Shortest Path* (APSP) problem [45] one is given an n node edge-weighted graph $U = (V, E)$. As in the previous problem all weights are integers from $\{-n^c, \ldots, n^c\}, c > 0$. It is supposed that there are no cycles of total negative weight in the graph. For each pair of vertices v, u one needs to find the weighted distance between them, i.e. the minimum over all paths from v to u of the total weight sum of the edges of the path. Hypothesis states that no algorithm with time $\mathcal{O}(n^{3-\epsilon}), \epsilon > 0$ can solve APSP problem.

In *Boolean Matrix Multiplication (BMM)* problem [45] it is needed to calculate matrix product of the two given $n \times n$ matrices over boolean semiring. The fastest algorithm currently known [4] solves the task in $\mathcal{O}(n^{2.372\ldots})$ using algebraic methods. The minimal constant c such that the BMM problem can be solved in $\mathcal{O}(n^c)$ is known as ω, matrix multiplication exponent. Combinatorial BMM hypothesis [23] states that there is no $\mathcal{O}(n^{3-\epsilon}), \epsilon > 0$ combinatorial algorithm for BMM. Combinatorial algorithms are not explicitly defined but the idea behind the term is that the algorithm does not use algebraic tricks to get smaller degree in polynomial; also combinatorial algorithms are more often used in practice because algebraic tricks lead to big constants behind $\mathcal{O}$-notation.

Orthogonal Vectors (OV) problem decides whether two sets X, Y of n boolean $d = \omega(\log n)$-dimensional vectors contain a pair $x \in X, y \in Y$ which dot product equals zero. Conjecture states that OV problem cannot be solved in $\mathcal{O}(n^{2-\epsilon} \cdot poly(d)), \forall \epsilon > 0$ time.

Given an undirected graph U on n vertices the *k-Clique* problem [1] seeks the clique on k vertices in U. If $0 \leq F \leq \omega$ and $0 \leq C \leq 3$ are the smallest numbers such that k-Clique can be solved combinatorially in $\mathcal{O}(n^{\frac{Ck}{3}})$ time and in $\mathcal{O}(n^{\frac{Fk}{3}})$ time by any algorithm, for any constant $k \geq 1$, a conjecture [45] is that $C = 3$ and $F = \omega$.

Triangle detection problem [17] is a case of k-Clique problem with $k = 3$, it asks whether an undirected graph $G = (V, E)$ contains three nodes $i, j, k \in V$ with $(i, j), (j, k), (k, i) \in E$. It can be solved in $\mathcal{O}(n^3)$ time by combinatorial algorithms, and in $O(n^\omega)$ time in general.

Language Editing Distance (LED) problem is about determining the minimum number of corrections: insertions, deletions, substitutions, that are needed to transform the given string $w \in \Sigma^*$ into the string w' that lies in the given context-free language $\mathcal{L}(G)$ over the same alphabet Σ. This problem is known [3] to be solvable in $\mathcal{O}(n^3)$ time, where n is the length of the string and the grammar is considered to be fixed.

Dynamic problems The following problems are connecting CFL reachability with dynamic problems, where for one input there can be multiple queries or updates.

In the *Online boolean Matrix-Vector multiplication (OMV)* problem [19] we are given an $n \times n$ boolean matrix M, we receive n boolean vectors $v_1, \ldots, v_n$ one at a time, and are required to output Mv_i (over the boolean semiring) before seeing the vector v_{i+1}, for all i. It is conjectured that there is no algorithm with total time $\mathcal{O}(n^{3-\epsilon})$ for this problem, even with polynomial time to preprocess M.

The incremental *Dynamic Transitive Closure (DTC)* [16] problem asks to maintain reachability information in a directed graph $D = (V, E)$ between arbitrary pairs of vertices under insertions of edges. Conditional lower bound on DTC follows from OMV hypothesis and reduction from it [19]: there is no algorithm with total update time $\mathcal{O}((mn)^{1-\epsilon})$ ($n = |V|, m = |E|$) even with $poly(n)$ time preprocessing of the input graph and $m^{\delta-\epsilon}$ time per query for any $\delta \in (0, 1/2]$ such that $m = \Theta(n^{1/(1-\delta)})$ under OMV hypothesis.

Problems with lower bound based on SAT, 3SUM or APSP Below are listed two problems that have conditional lower bounds under SAT, 3SUM or APSP hypotheses and thus reductions from them to CFL reachability problem will create conditional lower bounds based on widely believed conjecture.

Our first problem of interest is *AE-Mono*Δ problem where we are presented with a graph $G = (V, E)$ on n vertices in which every edge e_{uv} is colored with color $c(e_{uv})$, where c is color function. AE-MonoΔ problem asks for every edge e_{uv} whether there exists a monochromatic triangle that includes this edge, i.e. there is vertex w that $e_{uw}, e_{vw} \in E(G)$ and $c(e_{uv}) = c(e_{uw}) = c(e_{vw})$. This problem is known [43, 42] to be solvable in $n^{(3+\omega)/2}$ time and has [24] $n^{2.5}$ conditional lower bound under 3SUM and APSP hypotheses (which is tight if $\omega = 2$).

Second problem is really a pair of problems, both of them have [2] n^3 conditional lower bound under SETH, 3SUM and APSP hypotheses, where n is the number of vertices in the given graph. In the *Triangle collection* problem one is given a node-colored graph, and the question is whether for all triples of distinct colors there exists a triangle in a graph which vertices have these colors. Δ *Matching Triangles* problem have the same input but in this problem one is asked about existence of a triple of colors for which there are at least Δ triangles in a graph which vertices are colored in this triple.

3.3 Basic algorithms and examples

In this section we present the basic algorithms for the defined CFL problems and a classic example of the fine-grained reduction.

CFG recognition

Cocke–Younger–Kasami (CYK) algorithm [38] is a cubic algorithm for the CFL recognition problem. Recall that in CFL recognition problem it is needed to decide whether the given string $w = a_1 \dots a_n$ lies in the given context-free language $\mathcal{L}(G)$. CYK algorithm for each substring $a_{i+1} \dots a_j, 0 \le i < j \le n$ of w builds set $T_{i,j}$:

$$T_{i,j} = \{A \in N | A \Rightarrow^* a_{i+1} \dots a_j\}$$

After all the sets are built what is left is only to check if $S \in T_{0,n}$, which by construction of the sets is equivalent to $S \Rightarrow^* w$.

CYK algorithm needs as an input a string and a grammar G in CNF. Recall from Sec. 3.1 that every grammar can be translated into CNF in $poly(|G|)$ time. Algorithm is based on the idea that if $A \Rightarrow^* w'$ then there exists a production $A \to BC$ and a partition of the string $w' = w'' \cdot w'''$ such that $B \Rightarrow^* w''$ and $C \Rightarrow^* w'''$. For the pseudocode see Algorithm 1.

Algorithm 1 CYK algorithm

Input: Grammar $G = (N, \Sigma, P, S)$ in CNF, string $w = a_1 \dots a_n$, $a_i \in \Sigma$

Output: True if $w \in \mathcal{L}(G)$, False otherwise

```
1: if w = ε then
2:     return S → ε ∈ P
3: for i = 1 to n do
4:     T_{i-1,i} = {A | A → a_i ∈ P}
5: for l = 2 to n do
6:     for j = 0 to n − l do
7:         j = i + l
8:         T_{i,j} = ∅
9:         for A → BC ∈ P do
10:            for k = i + 1 to j − 1 do
11:                if B ∈ T_{i,k} and C ∈ T_{k,j} then
12:                    T_{i,j} = T_{i,j} ∪ {A}
13: return S ∈ T_{0,n}
```

Time complexity of the algorithm is $\mathcal{O}(|P| + n \cdot |P| + n^3 \cdot |P| \cdot |N|^2)$ which equals to $\mathcal{O}(n^3)$ as we suppose the grammar fixed.

CFL reachability

Classic algorithm for (all-pairs) CFL reachability problem [27, 20] is a generalisation of the CYK algorithm. It is needed to find all S-reachable pairs of vertices in the given graph $D = (V, E, L)$, $|V| = n$ for the given grammar $G = (N, \Sigma, P, S)$. The idea of the algorithm is based on the transitive closure, it iteratively inserts edges $v \xrightarrow{A} u$ in D for all A-reachable pairs of vertices (v, u) for every nonterminal $A \in N$. For the pseudocode see Algorithm 2.

Algorithm 2 CFL reachability algorithm

Input: Grammar $G = (N, \Sigma, P, S)$ in CNF; directed labeled graph $D = (V, E, L), L \subset \Sigma$

Output: Pairs of S-reachable vertices in D

1: $W = \emptyset$
2: **if** $S \to \epsilon \in P$ **then**
3: **for** $v \in V$ **do**
4: Insert edge $v \xrightarrow{S} v$ in D and triple (v, v, S) in W
5: **for** $A \to a \in P$ **do**
6: **for** $v \xrightarrow{a} u \in E$ **do**
7: Insert edge $v \xrightarrow{A} u$ in D and triple (v, u, A) in W
8: **while** $W \neq \emptyset$ **do**
9: Extract triple (v, u, A) from W
10: **for** $u \xrightarrow{B} w \in E$ **do**
11: **for** $C \to AB \in P$ **do**
12: **if** $v \xrightarrow{C} w \notin D$ **then**
13: Insert edge $v \xrightarrow{C} w$ in D and triple (v, w, C) in W
14: **for** $w \xrightarrow{B} v \in E$ **do**
15: **for** $C \to BA \in P$ **do**
16: **if** $w \xrightarrow{C} u \notin D$ **then**
17: Insert edge $w \xrightarrow{C} u$ in D and triple (w, u, C) in W
18: **return** $\{(u, v) | u \xrightarrow{S} v \in E\}$

The algorithm correctness proof is based on the induction over the derivation length of the word on S-path between each pair of vertices. For establishing the time complexity of the algorithm it is enough to notice the following. Between every two vertices it can be no more than $|\Sigma| + |N|$ labeled edges, every triple in W corresponds to the inserted edge and is added and deleted from W at most once. The time complexity of the algorithm is $\mathcal{O}(|P| \cdot n + |P| \cdot n^2 + (|\Sigma| + |N|) \cdot n^2 \cdot n \cdot |P|) = \mathcal{O}(n^3)$ as in processing triple (v, u, A) we iterate only over neighbours of v and u.

SAT to OV fine-grained reduction [44]

Below we build a (SAT, 2^n)→(OV, n^2) fine-grained reduction. This is one of the basic and easy to understand reductions, moreover it implies that the OV conjecture is stronger than SETH. This makes OV problem a good candidate to make a reduction from.

Reduction Let F be a SAT formula in k-conjunctive normal form with n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$. Without loss of generality $n \vdots 2$, otherwise we can add one more variable x_{n+1} and a clause $c_{m+1} = (x_{n+1} \vee \ldots \vee x_{n+1})$, this operation does not affect the satisfiability of F. We build two sets of vectors X, Y of the equal size $|X| = |Y| = 2^{\frac{n}{2}}$, where each vector has dimension m. Each vector x^α from X corresponds to the assignment α of values of $x_1, \ldots, x_{\frac{n}{2}}$. Then

$$x_i^\alpha = \begin{cases} 1 & \text{if } x^\alpha \text{ satisfies clause } c_i \\ 0 & \text{otherwise} \end{cases}$$

Each vector y^β from Y corresponds to the assignment β of values of $x_{\frac{n}{2}+1}, \ldots, x_n$. Values of y^β are defined in the same way as for x:

$$y_i^\beta = \begin{cases} 1 & \text{if } y^\beta \text{ satisfies clause } c_i \\ 0 & \text{otherwise} \end{cases}$$

Correctness Correctness of the reduction follows from the series of equivalent statements below.

F is satisfiable if and only if there exists an assignment γ which satisfied every clause $c_i, i \in \{1, \ldots, m\}$. Let $\gamma = (\alpha', \beta')$, where α' assigns the values of $x_1, \ldots, x_{\frac{n}{2}}$, β' assigns the values of $x_{\frac{n}{2}+1}, \ldots, x_n$. For each clause c_i it is true that c_i is satisfied by γ if and only if it is satisfied by at least one of α' or β'. The last observation is equivalent to the fact that $\forall i \in \{1, \ldots, m\}\ x_i^{\alpha'} \cdot y_i^{\beta'} = 0$, which exactly means orthogonality of vectors $x^{\alpha'}$ and $y^{\beta'}$.

The following theorem concludes the proof of correctness.

Theorem 3.1. *SETH implies OV hypothesis.*

Proof. Suppose OV problem can be solved in time $\mathcal{O}(n^{2-\epsilon} \cdot poly(d))$ for some $\epsilon > 0$.

Let F be arbitrary k-CNF formula. From F we create an OV instance as described above with $|X| = |Y| = 2^{\frac{n}{2}}$ vectors in each set with dimension $m \leq n^k$. Thus OV instance can be created in time $\mathcal{O}(2^{\frac{n}{2}} \cdot n^k)$. After that we solve OV problem with the existing algorithm in $\mathcal{O}((2^{\frac{n}{2}})^{2-\epsilon} \cdot m^{\mathcal{O}(1)}) = \mathcal{O}(2^{n-\frac{\epsilon}{2}} \cdot n^{\mathcal{O}(k)}) = \mathcal{O}(2^{n-\epsilon'})$ time for arbitrary k. The answer to the OV instance is the answer to the original SAT instance. That implies that k-SAT problem can be solved in $\mathcal{O}(2^{\frac{n}{2}} \cdot n^k + 2^{n-\epsilon'}) = \mathcal{O}(2^{n-\epsilon''}), \epsilon'' > 0$ time for any k which contradicts SETH. □

4 Overview of existing reductions

Before looking onto the reductions we need to mention the interconvertibility of CFL reachability problems and a class of set-constraint problems [27] as this result allows us to reformulate our problem if we wish so.

The CFL reachability problem has been shown to be complete for the class of two-way nondeterministic pushdown automata (2NPDA) [18]. It means that subcubic algorithm for CFL reachability would lead to subcubic algorithms for the whole 2NPDA class and cubic upper bound has not been improved since the discovery of the class in 1968. Now we are ready to proceed to the fine-grained reductions.

First of all we want to separate as subcase of CFL reachability — Dyck-1 reachability problem. This subcase, when the grammar G is fixed as the grammar of Dyck language on one type of parentheses, is known to be solvable in truly subcubic time and thus could be easier than the general CFL reachability problem.

Concerning the upper bounds on this subcase one of the important reductions is a reduction from all-pairs Dyck-1 reachability problem to BMM problem. It was firstly

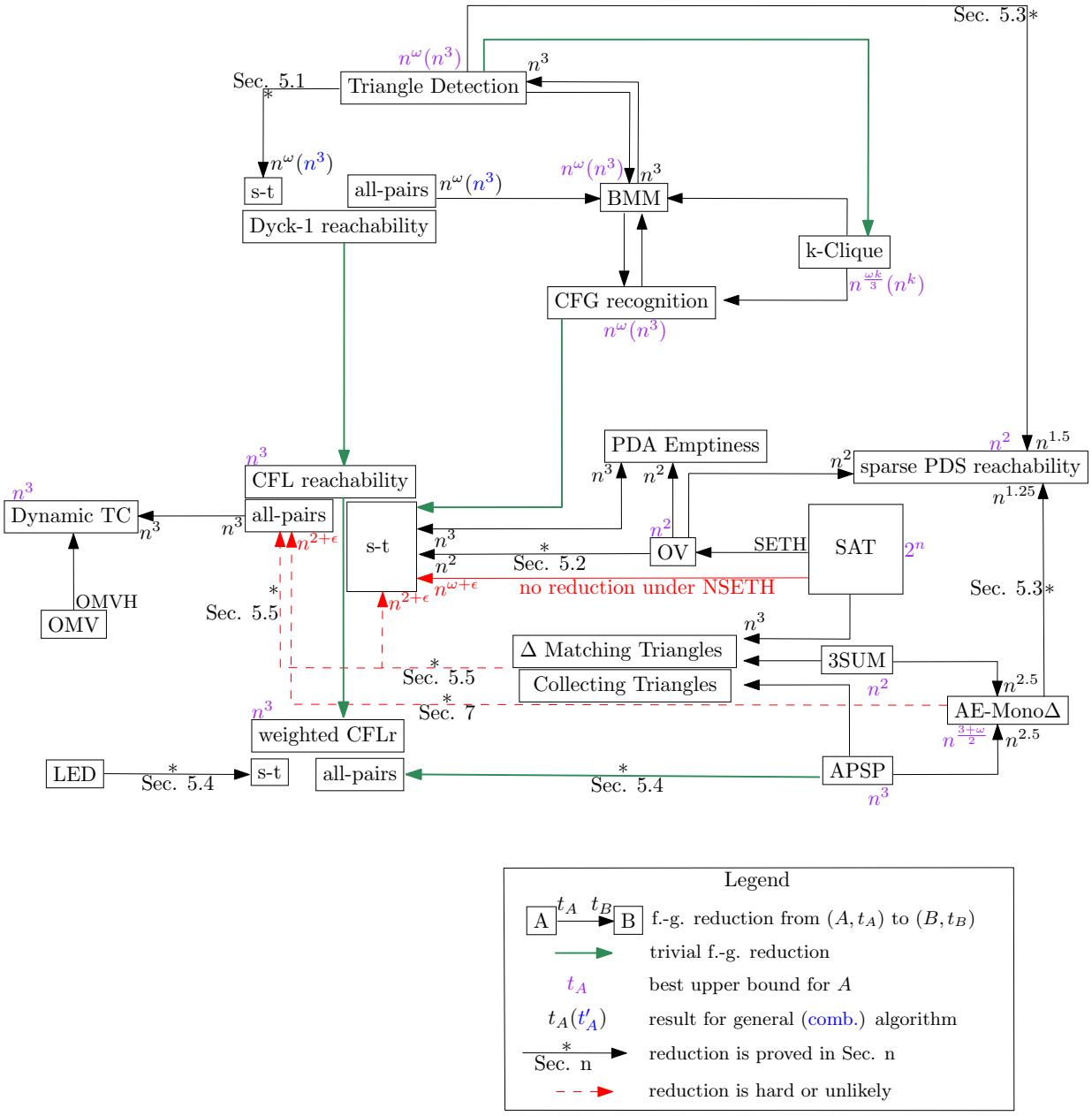

Figure 1: Existing reductions concerning CFL reachability and CFG recognition. Arrow $a \to b$ represents a fine-grained reduction from a to b. Note that asymptotics in upper bounds are written up to polylogarithmic (or polynomial in case of SAT) factors. When the asymptotic is not written on some side of the arrow it is equal to the upper bound of the problem (written in violet).

proved by Bradford [7] via algebraic matrix encoding and then combinatorially by Mathiasen and Pavlogiannis [25] by combining Dyck-1 path from bell-shaped paths. Via this reduction all-pairs (and s-t) Dyck-1 reachability can be solved in n^3 time by combinatorial algorithm and in n^ω time by general one. For this and the following reductions see Fig. 1.

Through the subcubic combinatorial reductions between BMM and Triangle detection [47] and reduction from the latter to s-t Dyck-1 reachability (see Sec. 5.1 and [17]) we get that the above cubic upper bound is tight for combinatorial algorithms under BMM hypothesis. That implies that in the world of combinatorial algorithms Dyck-1 case with high probability is not easier than the general one. Next we discuss the results for CFG recognition and general CFL reachability.

$\mathcal{O}(n^\omega)$ upper bound on CFG recognition was shown by Valiant in his famous paper [41]. He reduced finding sets from CYK algorithm to several calls of BMM. The other way reduction created by Lee [23] was combinatorial and thus allows us to get BMM based lower bounds on CFG recognition problem in both combinatorial and general case. Combining the latter reduction with trivial reduction from CFG recognition to s-t CFL reachability we get another cubic combinatorial lower bound on our problem under BMM hypothesis, but this time we also achieve $\mathcal{O}(n^\omega)$ conditional lower bound in general case.

Another $\mathcal{O}(n^\omega)$ lower bound on s-t CFL reachability was achieved via combination of reduction of Abboud et al. [1] from k-Clique to CFG recognition and the reduction from CFG recognition to CFL reachability. This lower bound is based on k-Clique hypothesis. There are some reductions that are unlikely to help improve upper or lower bounds on the problem but they give insight on the essence of CFL reachability.

In the recent paper of Shemetova et al. [37] the reduction from all-pairs CFL reachability to incremental DTC have been proven. Still this reduction cannot give truly subcubic algorithm for CFL reachability without refuting OMV conjecture [8, 19].

Subcubic equivalence between s-t CFL reachability and PDA Emptiness problem was proven by Schepper [35]. This equivalence allows us to reformulate the problem when we are proving cubic conditional lower bound on CFL reachability.

Natural question that rises considering the lower bounds on CFL reachability is whether there exist one based on 3SUM, APSP or SAT hypotheses. Partially this question has been answered. Recently it was discovered by Chistikov et al. [14] that there exist subcubic certificates for s-t CFL reachability (for existence and non-existence of the valid paths). From this fact it follows that there are no reductions under NSETH from SAT problem (SETH) to CFL reachability problem that give lower bound stronger than $\mathcal{O}(n^\omega)$.

5 Lower bounds

This section is devoted to fine-grained reductions that produce lower bounds on CFL reachability problem or the problems connected to it.

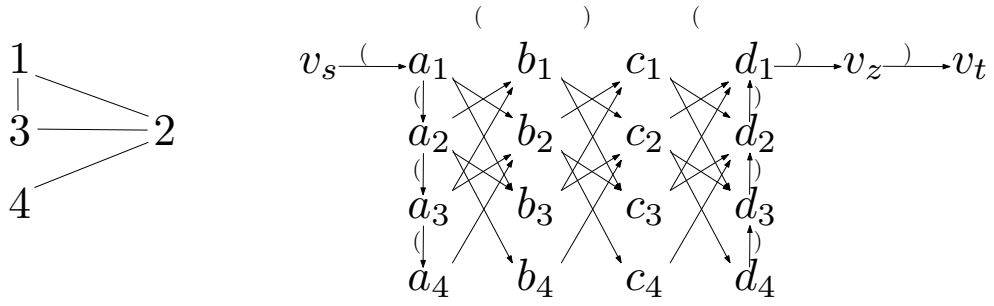

Figure 2: The given graph in the triangle detection problem on the left and the corresponding graph with (v_s, v_t)-CFL Reachability problem on the right. Every vertex i corresponds to vertices a_i, b_i, c_i, d_i; edges $(a_i, b_j), (b_i, c_j), (c_i, d_j)$ are marked with $(,), ($ respectively.

5.1 Lower bound based on BMM hypothesis

There exists a $\mathcal{O}(n^3)$ conditional combinatorial lower bound on s-t Dyck-1 reachability problem following from the fine-grained reduction from Triangle Detection problem. The reduction and the proof of its correctness is the same as the reduction from Triangle Detection to One-counter Net reachability problem (see Theorem 2 in [17]). One-counter Net is a model in program verification isomorphic to Dyck-1 reachability. As formally the models are different, below we reformulate the result from [17] in terms of Dyck-1 reachability.

Reduction Let $U = (V, E), V = \{x_1, \ldots, x_n\}$ be a graph in Triangle Detection problem. We build directed labeled graph $D = (V', E', L)$ for the Dyck-1 reachability problem as follows:

- $V' = \{v_s\} \cup \{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_n\} \cup \{c_1, \ldots, c_n\} \cup \{d_1, \ldots, d_n\} \cup \{v_z\} \cup \{v_t\}$
- (v_s, v_t) is a pair of vertices, for which we want to check whether it is Dyck-1-reachable
- $E' = \{(v_s, '(', a_1)\} \cup \{(a_i, '(', a_{i+1}) | i \in \{1, \ldots n-1\}\} \cup$
  $\{(a_i, '(', b_j) | (x_i, x_j) \in E\} \cup \{(b_i, ')', c_j) | (x_i, x_j) \in E\} \cup$
  $\{(c_i, '(', d_j) | (x_i, x_j) \in E\} \cup \{(d_i, ')', d_{i-1}) | i \in \{2, \ldots n\}\} \cup$
  $\{(d_1, ')', v_z)\} \cup \{(v_z, ')', v_t)\}$, where (v, l, u) denotes a directed edge from v to u with label l on it. Note that each undirected edge (x_i, x_j) transforms into 6 directed edges, 2 between a- and b-vertices, b- and c-vertices, c- and d-vertices
- $L = \{'(', ')'\}$ that is exactly the Dyck-1 language alphabet

For visual representation of the reduction see Fig. 2.
Correctness Let us first estimate the size of D:

- $|V'| = 3 + 4 \cdot |V| = \mathcal{O}(n)$
- $|E'| = 3 + 2 \cdot (n-1) + 6 \cdot |E| = \mathcal{O}(|E|) = \mathcal{O}(n^2)$

Thus the graph D can be constructed in $\mathcal{O}(n^2)$ time.

Lemma 5.1. *v_t is Dyck-1-reachable from v_s in D if and only if U contains a triangle.*

Proof. We first prove the "if"-part of the claim.

Suppose vertices x_i, x_j, x_k form a triangle in U. Then in D there exists a path

$$v_s \to a_1 \to \ldots \to a_i \to b_j \to c_k \to d_i \to \ldots \to d_1 \to v_z \to v_t$$

The corresponding string of concatenated labels is

$$(\overbrace{(\ldots(}^{i-1 \text{ times}}\ ()(\ \overbrace{)\ldots)}^{i-1 \text{ times}}\ )) = \overbrace{(\ldots(}^{i \text{ times}}()()\overbrace{)\ldots)}^{i \text{ times}}$$

which is clearly a Dyck-1 word.

Now we proceed to the "only if"-part of the claim.

Suppose there exists a Dyck-1 path from v_s to v_t. Note that D is acyclic. Then from the construction of a graph we can see that the path should look like:

$$v_s \to a_1 \to \ldots \to a_l \to b_j \to c_k \to d_i \to \ldots \to d_1 \to v_z \to v_t$$

The corresponding word w, that we know is Dyck-1, is:

$$(\overbrace{(\ldots(}^{l-1 \text{ times}}\ ()(\ \overbrace{)\ldots)}^{i-1 \text{ times}}\ )) = \overbrace{(\ldots(}^{l \text{ times}}()()\overbrace{)\ldots)}^{i \text{ times}}$$

Then w is Dyck-1 only if $l = i$. In that case by the construction of E' we know, that in U existed edges $(x_i, x_j), (x_j, x_k), (x_k, x_i)$. It implies that x_i, x_j, x_k is a triangle in U. □

Theorem 5.1. *There exists a (Triangle Detection, n^3) to (Dyck-1 reachability, n^3) combinatorial fine-grained reduction.*

Proof. We can construct D in time $\mathcal{O}(n^2)$. If Dyck-1 reachability problem is solvable in $\mathcal{O}(n^{3-\epsilon})$ time, then Triangle Detection is solvable in $\mathcal{O}(n^{3-\epsilon} + n^2) = \mathcal{O}(n^{3-\epsilon'})$ time. □

For the general algorithms we can state a similar theorem. It is proved in an analogous way.

Theorem 5.2. *There exists a (Triangle Detection, n^ω) to (Dyck-1 reachability, n^ω) fine-grained reduction.*

Combining the Theorem 5.1 with the combinatorial reduction from BMM to Triangle Detection [47] we get the following corollary.

Corollary 5.1. *If combinatorial BMM hypothesis is true, then there exist no combinatorial algorithm for Dyck-1 reachability problem with time $\mathcal{O}(n^{3-\epsilon}), \forall \epsilon > 0$.*

5.2 Lower bound based on OV conjecture

This section is devoted to the (OV, n^2) to (s-t CFL reachability, n^2) fine-grained reduction. As the previous one this reduction was firstly presented in [17] but as the reduction from (OV, n^2) to (sparse PDS reachability, n^2) reduction. PDS reachability is a problem isomorphic to CFL reachability, still as formally the problems are different we reformulate the reduction in terms of s-t CFL reachability. In addition we change the reduction from [17] a bit so it translates to CFL reachability problem without ϵ-edges.

We need to mention that this reduction gives nontrivial lower bound on s-t CFL reachability only in case of graphs with truly subquadratic number of edges, otherwise in graph with n vertices and $\mathcal{O}(n^2)$ edges $\mathcal{O}(n^2)$ time is needed only to read the input graph.

We proceed with the construction of the reduction.

Reduction Suppose we are given two sets X, Y of n d-dimensional vectors as an instance of OV problem, $X = \{x_1, \ldots, x_n\}, \forall i \; x_i = (x_1^i, \ldots, x_d^i)$, $Y = \{y_1, \ldots, y_n\}, \forall i \; y_i = (y_1^i, \ldots, y_d^i)$. We construct an instance of weighted CFL reachability problem $D = (V, E, L, \Omega)$, $G = (N, \Sigma, P, S)$ as follows:

Let denote by h_x the following function:

$$h_x(b) = \begin{cases} (& \text{if } b = 0 \\ [& \text{if } b = 1 \end{cases} \quad \forall i \in \{1, \ldots, n\} \; \forall j \in \{1, \ldots, d\}$$

Let denote by h_y the following function:

$$h_y(b) = \begin{cases}) & \text{if } b = 0 \\ ] & \text{if } b = 1 \end{cases} \quad \forall i \in \{1, \ldots, n\} \; \forall j \in \{1, \ldots, d\}$$

- G is the grammar of Dyck-2 language
- $V = \{v_s, v_l, v_t\} \cup (\cup_{i=1}^{n} \{u_j^i | j \in \{1, \ldots, d-1\}\}) \cup (\cup_{i=1}^{n} \{w_j^i | j \in \{1, \ldots, d-1\}\})$
- (v_s, v_t) is a pair of vertices for which we want to know if it is Dyck-2-reachable
- $E = E_{sl} \cup E_{lt}$, where
  $E_{sl} = (\cup_{i=1}^{n} (\{(v_s, h_x(x_1^i), u_1^i)\} \cup$
  $\{(u_j^i, h_x(x_{j+1}^i), u_{j+1}^i) | j \in \{1, \ldots, d-2\})\} \cup$
  $\{(u_{d-1}^i, h_x(x_d^i), v_l)\}))$
  $E_{lt} = (\cup_{i=1}^{n} (\{(v_l, h_y(0), w_1^i)\} \cup \{(v_l, h_y(1), w_1^i | y_d^i = 0)\} \cup$
  $\{(w_j^i, h_y(0), w_{j+1}^i) | j \in \{1, \ldots, d-2\}\} \cup$
  $\{(w_j^i, h_y(1), w_{j+1}^i) | y_d^j = 0 \; j \in \{1, \ldots, d-2\}\} \cup$
  $\{(w_{d-1}^i, h_y(0), v_t)\} \cup \{(w_{d-1}^i, h_y(1), v_t) | y_1^i = 0\})),$
  where (v, l, u) denotes a directed edge from vertex v to vertex u labeled with symbol l.

For visual representation of the reduction see Fig. 3.

Correctness Let us first estimate the size of D:

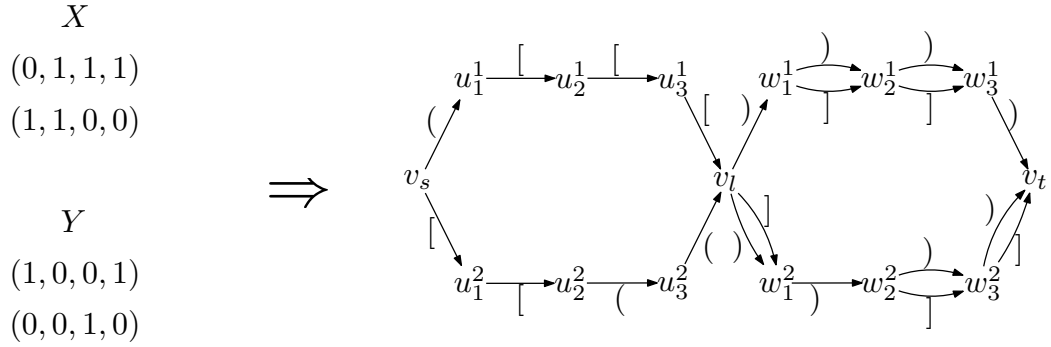

Figure 3: The given sets X, Y in the OV problem on the left and directed graph D on the right.

- $V = 3 + 2 \cdot (d-1) \cdot n = \mathcal{O}(nd)$
- As each vertex of type u_j^i, w_j^i, v_t has outgoing degree at most 2 and vertices v_s, v_l have outgoing degree at most $2n$ we have $|E| = \mathcal{O}(nd)$.

Therefore the size of D is $\mathcal{O}(nd)$ and it can be constructed in this time from OV instance.

Lemma 5.2. *There exist two orthogonal vectors $x_i \in X, y_j \in Y$ if and only if (v_s, v_t) is a Dyck-2-reachable pair of vertices in D.*

Proof. We first prove the "only if"-part of the claim.

Suppose vectors $x_i \in X, y_j \in Y$ are orthogonal. We claim that there exists in D a Dyck-2 path of the following construction (for now we do not choose one edge from multiple edges between consecutive vertices in the path, if there exist any):

$$v_s \to u_1^i \to \ldots \to u_{d-1}^i \to v_l \to w_1^j \to \ldots \to w_{d-1}^j \to v_t \tag{1}$$

Note that the parenthesis on the outgoing edge from vertex u_l^i should match with the parenthesis on the edge ingoing to vertex w_{d-i}^j. Case of vertices v_s, v_t is similar and can be analyzed in the same way. Now we can build the Dyck-2 path iteratively:

- If $x_l^i = 1$ we know that $y_l^j = 0$ as x_i, y_j are orthogonal. By the construction an edge from u_l^i is marked with "[" and one of the two edges to vertex w_{d-i}^j is marked with "]". We choose these two edges with matching parentheses to our path.

- If $x_l^i = 0$ we know that y_l^j may be 0 and may be 1 as x_i, y_j are orthogonal. By the construction an edge from u_l^i is marked with "(" and in both cases ($y_l^j = 0$ and $y_l^j = 1$) there exists an edge to vertex w_{d-i}^j that is marked with ")". We choose these pair edges with matching parentheses to our path.

By the construction of the path it is a Dyck-2 path and (v_s, v_t) is a Dyck-2-reachable pair of vertices in D.

Now we proceed to the "if"-part of the claim.

If v_t is Dyck-2-reachable from v_s in D then there exists a Dyck-2 path between them. By the construction of D the path is of the way described in (1) for some i and j. By the same observations as in the previous case we note that all pairs of parentheses on the edges should match. We prove for every pair of bits x^i_m, y^i_m that they are orthogonal:

- If the corresponding pair of parentheses is a ()-pair. Then $x^i_m = 0$ and m-th pair of bits of x_i, y_j is orthogonal.
- If the corresponding pair of parentheses is a []-pair, then $y^i_m = 0$, because only in this case we create an edge labeled with "]" ingoing to vertex w^j_{d-m}. Thus m-th pair of bits of x_i, y_j is orthogonal.

We have proved that $x_i \in X, y_j \in Y$ are orthogonal. □

Theorem 5.3. *There exists an (OV, n^2) to (s-t CFL reachability, n^2) fine-grained reduction.*

Proof. Suppose we are given an instance of OV problem: two sets X, Y of n d-dimensional vectors. We construct an instance D, G of s-t CFL reachability problem in time $\mathcal{O}(nd)$ as described above.

If there exists an $\mathcal{O}(n^{2-\epsilon})$-time algorithm for CFL reachability, then we can get an answer to an instance D, G in time $\mathcal{O}((nd)^{2-\epsilon} = \mathcal{O}(n^{2-\epsilon} \cdot poly(d))$ and it is the answer to original OV instance. Thus we can solve OV problem in time $\mathcal{O}(n^{2-\epsilon} \cdot poly(d))$. □

As in the reduction we have build a sparse graph D we can get the following corollary.

Corollary 5.2. *There in no $n^{2-\epsilon}, \epsilon > 0$ algorithm for s-t CFL reachability on sparse graphs unless OV hypothesis is false.*

5.3 Lower bound on push-down systems

Next we present a fine-grained combinatorial reduction (Triangle Detection, n^3) to (sparse PDS reachability with stack depth $b = \lceil \log n \rceil$, $n^{1.5}$). CFL reachability problem can be restated in terms of PDS reachability. In this section we want to add a restriction on stack depth which is a natural restriction in terms of PDS, therefore we use this problem.

Reduction

Let $U = (V, E), V = \{x_1, \ldots, x_n\}$ be a graph in Triangle Detection problem. We build PDS $\mathcal{P} = (Q, \Gamma, \delta)$ as follows:

- Set of states Q consist of ordinary states Q' and auxiliary states Q''.
  - $Q' = \{q_s\} \cup \{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_n\} \cup \{c_1, \ldots, c_n\} \cup \{d_1, \ldots, d_n\} \cup \{q_t\}$.
  - Q'' are auxiliary states, they appear in transitions in which we want to read or write more than one symbol from the stack. If in transition t we want to read k_1 symbols and write k_2 symbols, it is separated into $k_1 + k_2$ transitions. In each of the new transitions we either read one symbol from the stack without writing or write one symbol to the stack without reading. This procedure creates $k_1 + k_2 - 1$ auxiliary states.

    Another auxiliary states can appear when we want to split a transition in two transitions. This type of auxiliary states will help to maintain the system sparse.

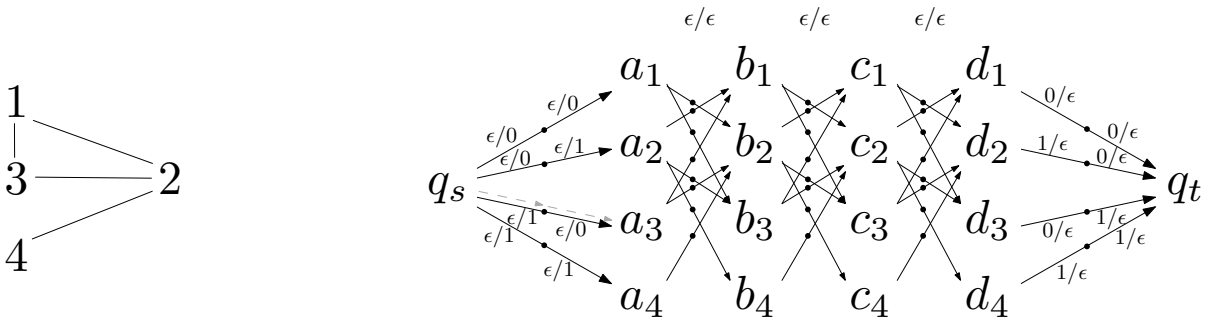

Figure 4: The given graph in the triangle detection problem on the left and PDS $\mathcal{P}$ on the right. An edge from v to u labeled x/y corresponds to a transition $(v, x, u, y) \in \sigma$. Each dot symbolizes auxiliary state, they split a transition into many transitions, e.g. see edge (q_s, a_3).

- q_s, q_t are the states, for which we want to check reachability
- $\Gamma = \{0, 1\}$
- $\delta = \{(q_s, \epsilon, [i]_2, a_i) | i \in \{1, \dots n\}\} \cup$
  $\{(a_i, \epsilon, \epsilon, b_j) | (x_i, x_j) \in E\} \cup \{(b_i, \epsilon, \epsilon, c_j) | (x_i, x_j) \in E\} \cup$
  $\{(c_i, \epsilon, \epsilon, d_j) | (x_i, x_j) \in E\} \cup \{(d_i, \overline{[i]_2}, \epsilon, q_t) | i \in \{1, \dots n\}\}$, where $[i]_2$ denotes a binary representation of i (padded with leading 0-s if necessary) of length $\lceil \log n \rceil$, $\overline{[i]_2}$ denotes a reversed binary representation of i. Each transition which reads or writes $[i]_2$ is split in $\lceil \log n \rceil$ transitions with auxiliary states. Each transition which reads and writes ϵ, nothing, is split in two transitions with one auxiliary state.

For visual representation of the reduction see Fig. 4.
Correctness Let us first estimate the size of $\mathcal{P}$:

- $|Q| = |Q'| + |Q''| \leq (2 + 3 \cdot n) + (2 \cdot \log n + 3 \cdot n^2) = \mathcal{O}(n^2)$
- Each auxiliary state has exactly one transition from it. For each ordinary state transitions from it do not exceed n. As $|Q'| = \mathcal{O}(n)$, we get $|\delta| = \mathcal{O}(n^2) = \mathcal{O}(|Q|)$

Thus the system $\mathcal{P}$ can be constructed in $\mathcal{O}(n^2)$ time.

Lemma 5.3. *q_t is reachable from q_s in $\mathcal{P}$ if and only if U contains a triangle.*

Proof. We first prove the "if"-part of the claim.

Suppose vertices x_i, x_j, x_k form a triangle in U. Then in $\mathcal{G}_\mathcal{P}$ there exists a path

$$(q_s, \epsilon) \to \dots \to (a_i, [i]_2) \to \dots \to (b_j, [i]_2) \to \dots$$

$$\dots \to (c_k, [i]_2) \to \dots \to (d_i, [i]_2) \to \dots \to (q_t, \epsilon) \tag{2}$$

Now we proceed to the "only if"-part of the claim.

Graph $\mathcal{G}_\mathcal{P}$ is acyclic and by its construction path from (q_s, ϵ) to (q_t, ϵ) must be of the way described in equation (2). It implies that x_i, x_j, x_k form a triangle in U. □

Theorem 5.4. *There exists a (Triangle Detection, n^3) to (sparse PDS reachability with stack depth $b = \lceil \log n \rceil$, $n^{1.5}$) combinatorial fine-grained reduction.*

Proof. By the observations above constructed PDS $\mathcal{P}$ is sparse and of size $|Q| = \mathcal{O}(n^2)$. stack depth is bounded by $[n]_2 = \lceil \log n \rceil$: we can only write on stack the number of one vertex in binary encoding.

$\mathcal{P}$ can be constructed in $\mathcal{O}(n^2)$ time. If there was $\mathcal{O}(n^{1.5-\epsilon})$ algorithm for PDS reachability it would lead to $\mathcal{O}(n^2 + (n^2)^{1.5-\epsilon}) = \mathcal{O}(n^{3-\epsilon'})$ algorithm for Triangle Detection. □

We must mention that in [12] it was proven conditional $\mathcal{O}(n^3)$ lower bound on PDS reachability based on BMM hypothesis, and by the similar technique of splitting edges it can be further translated into the conditional lower bound the was described above. Nevertheless the presented reduction is easier, has additional restriction on a stack depth and helps us in construction of the reduction below. The following conditional lower bound is based on 3SUM and APSP hypotheses.

We proceed with describing (AE-MonoΔ, $n^{2.5}$) to (sparse all-pairs PDS reachability with stack depth $b = 4\lceil \log n \rceil$, $n^{1.25}$) fine-grained reduction.

Reduction Suppose we are given graph $U = (V, E), V = \{v_1, \ldots, v_n\}$ as an input in AE-MonoΔ problem. The construction of PDS is very similar to the one described above and based on the construction described in [22] (see Theorem 3.2 in [22]):

- Set of states Q analogously consists of ordinary states Q' and auxiliary states Q''. As rules for the auxiliary states remain the same we describe only set of ordinary states.

  $Q' = \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\} \cup \{z_1, \ldots, z_n\} \cup \{x'_1, \ldots, x'_n\} \cup \{y'_1, \ldots, y'_n\}$.

- $(x_i, y'_j), i, j \in \{1, \ldots, n\}$ are the pairs of states, for which we want to check reachability, for that we will run all-pairs reachability algorithm

- $\Gamma = \{0, 1\}$

- $\delta = \{(x_i, \epsilon, [i]_2\,[j]_2\,[c(e_{ij})]_2, y_j) | (x_i, x_j) \in E\} \cup$
  $\{(y_i, \overline{[c(e_{ij})]_2}, [c(e_{ij})]_2, z_j) | (x_i, x_j) \in E\} \cup$
  $\{(z_i, \overline{[c(e_{ij})]_2}, \epsilon, x'_j) | (x_i, x_j) \in E\} \cup$
  $\{(x'_i, \overline{[j]_2}\,\overline{[i]_2}, \epsilon, y'_j) | (x_i, x_j) \in E\}$, where $[i]_2$ denotes a binary representation of i (padded with 0-s if necessary) of length $\lceil \log n \rceil$, $[c(e_{ij})]_2$ denotes a binary representation of number of color of the edge $e_ij = (v_i, v_j)$ in U and is of length $2\lceil \log n \rceil$ (padded with 0-s if necessary).

For visual representation of the reduction see Fig. 7.
Correctness Let us first estimate the size of $\mathcal{P}$:

- $|Q| = |Q'| + |Q''| \leq (5 \cdot n) + (4 \cdot (4 \log n + 1) \cdot n^2) = \mathcal{O}(n^2 \log n)$

- Each auxiliary state has exactly one transition from it. For each ordinary state transitions from it do not exceed n. As $|Q'| = \mathcal{O}(n)$, we get $|\delta| = \mathcal{O}(n^2) = \mathcal{O}(|Q|)$

Thus the system $\mathcal{P}$ can be constructed in $\mathcal{O}(n^2 \log n)$ time.

Lemma 5.4. *For every pair of vertices x_i, x_j there is a monochromatic triangle in U containing an edge e_{ij} if and only if x_i is reachable from y'_j in PDS $\mathcal{P}$.*

Proof. We first proof the "only if"-part of the claim.

Suppose there is a monochromatic triangle x_i, x_j, x_k in U. Then there is a path in $\mathcal{G}_\mathcal{P}$:

$$(x_i, \epsilon) \to \ldots \to (y_j, [i]_2\,[j]_2\,[c(e_{ij})]_2) \xrightarrow{(1)} \ldots \to (z_k, [i]_2\,[j]_2\,[c(e_{jk})]_2) \xrightarrow{(2)}$$

$$\ldots \to (x'_i, [i]_2\,[j]_2) \xrightarrow{(3)} \ldots \to (y'_j, \epsilon)$$

Note that as $c(e_{ij}) = c(e_{jk})$ transition (1) is valid, as $c(e_{jk}) = c(e_{ki})$ transition (2) is valid.

Now we proceed to the "if"-part of the claim.

Suppose there is a path from (x_i, ϵ) to (y'_j, ϵ) in $\mathcal{G}_\mathcal{P}$. By the construction of $\mathcal{P}$ and $\mathcal{G}_\mathcal{P}$ the path should be of the form:

$$(x_i, \epsilon) \to \ldots \to (y_j, [i]_2\,[j]_2\,[c(e_{ij})]_2) \xrightarrow{(1)} \ldots \to (z_k, [i]_2\,[j]_2\,[c(e_{jk})]_2) \xrightarrow{(2)}$$

$$(x'_{i'}, [i]_2\,[j]_2) \xrightarrow{(3)} \ldots \to (y'_{j'}, \epsilon)$$

As transition (1) is valid we get that $c(e_{ij}) = c(e_{jk})$, as transition (2) is valid we get that $c(e_{jk}) = c(e_{ki})$, as transition (3) is valid we get that $i' = i, j' = j$. Thus x_i, x_j, x_k is a monochromatic triangle in U. □

Theorem 5.5. *There exists an (AE-MonoΔ, $n^{2.5}$) to (sparse all-pairs PDS reachability with stack depth $b = 4\lceil \log n \rceil$, $n^{1.25}$) fine-grained reduction.*

Proof. By the observations above constructed PDS $\mathcal{P}$ is sparse and of size $|Q| = \mathcal{O}(n^2 \log n)$. Stack depth is bounded by $[n]_2 = 4\lceil \log n \rceil$: we write two numbers of vertices and a color number (which does not exceed n^2 as there at most n^2 edges in the graph) on the path from x-state to y-state, this gives $4\lceil \log n \rceil$-upper bound on the depth of the stack. Transitions between y- and z-vertices first read $2\lceil \log n \rceil$ symbols from the stack and then write the back, transitions between z- and x'-vertices and between x'- and y'-vertices only read symbols from the stack. This implies that our upper bound is correct.

$\mathcal{P}$ can be constructed in $\mathcal{O}(n^2 \log n)$ time. If there was $\mathcal{O}(n^{1.25-\epsilon})$ algorithm for PDS reachability it would lead to $\mathcal{O}(n^2 + (n^2)^{1.25-\epsilon}) = \mathcal{O}(n^{2.5-\epsilon'})$ algorithm for AE-MonoΔ. □

As AE-MonoΔ has no $\mathcal{O}(n^{2.5-\epsilon}), \epsilon > 0$ algorithm unless both 3SUM and APSP hypotheses are false, we get the following corollary.

Corollary 5.3. *The is no algorithm for sparse all-pairs PDS reachability with stack depth $b = 4\lceil \log n \rceil$ with time complexity $\mathcal{O}(n^{1.25-\epsilon}), \epsilon > 0$ if at least one of 3SUM and APSP hypotheses is true.*

5.4 Reductions from LED variations

Recall that LED problem is devoted to finding the minimum number of operations needed to transform the given word $w = w_1 \dots w_n \in \Sigma^*$ into the word from the given CFL $\mathcal{L}(G), G = (N, \Sigma, P, S)$. It can be reformulated in a way, that every operation on a word has a cost: each symbol replacement costs ω_{repl}, symbol deletion costs ω_{del}, symbol insertion costs ω_{ins}. The goal is to find transformation of w into some $w' \in \mathcal{L}(G)$ of minimum total weight. In the canonical formulation of a problem $\omega_{repl} = \omega_{del} = \omega_{ins} = 1$.

LED problem is close to CFL reachability problem as it combines in itself CFG recognition with Edit Distance problem (for which we want to find distance from string w to string w'). There are several cubic algorithms for LED [3, 28]. It is known of operations are restricted only to the insertions then a sub-cubic algorithm is unlikely to exists, this fact is proved via a reduction from APSP [34, 46]. On the other hand, if insertions and deletions (in both cases: with or without replacement) are allowed, LED is solvable in truly subcubic time: there is $\tilde{\mathcal{O}}(n^{2.8603})$ algorithm for that [10].

Next we present a reduction from LED to weighted s-t CFL reachability problem. The reduction works with every variation of LED: each operation may be allowed or restricted. Moreover different kind of operations may have different costs, if necessary.

Let *weighted CFL reachability problem* [6] be a generalisation of CFL reachability problem, where each edge in the given graph $D = (V, E, L, \Omega)$ has its integer weight ω, let Ω be the set of weights of the edges. We suppose that there are no cycles in D of total negative edge weight. Aside from the information about existence of $\mathcal{L}(G)$-path between some pairs of vertices in weighted CFL reachability problem we want to know the minimal possible weight of such path. There exists [6] an $\mathcal{O}(n^3)$ algorithm for weighted CFL reachability.

We allow edges to be labeled with ϵ, the empty string. This assumption does not affect the complexity of a problem: we can change the grammar $G = (N, \Sigma, P, S)$ into the new grammar $G' = (N', \Sigma', P', S)$ in such a way that it operates with ϵ as with a new symbol of an alphabet. For that we suppose that G is in CNF (otherwise we transform G to it) and add a new nonterminal E and a symbol ϵ' corresponding to ϵ ($\Sigma' = \Sigma \cup \{\epsilon'\}$), the empty string, and $2 \cdot |N| + 1$ new rules to productions P to get production set P':

- $E \to \epsilon'$
- $A \to EA|AE \ \ \forall A \in N$

In this case new language $\mathcal{L}(G')$ contains exactly the words which, without symbol ϵ' in them, lie in $\mathcal{L}(G)$. In the given graph D we change all edges with label ϵ to the same edges with label ϵ' and obtain a new graph D'. As ϵ' corresponds to the empty string the answer for CFL reachability problem for grammar G' and graph D' will be the same as the answer for CFL reachability problem for grammar G and graph D.

Is it easy to see that CFL reachability problem is a subcase of its weighted version: we can assign each edge weight 1 (that way no negative cycles can appear) and look only on information about reachability in the answer for weighted CFL reachability.

It is also easy to see that APSP problem is a subcase of all-pairs weighted CFL reachability as we can define G in such way that $\mathcal{L}(G) = \Sigma^*$, duplicate each edge and orient them in both ways and label the edges randomly. In that case every path will be

an $\mathcal{L}(G)$-path and we look for the path with minimal total edge weight between a pair of vertices.

Now we proceed to (LED, n^c) to (weighted CFL reachability, n^c), $c > 1$ fine-grained reduction.

Reduction Suppose we are given string $w = w_1 \dots w_n \in \Sigma^*$ and CFG $G = (N, \Sigma, P, S)$ as an instance of LED problem. We construct an instance of weighted CFL reachability problem $D = (V, E, L, \Omega)$, G' as follows:

- $G' = G$
- $V = \{s = v_1, \dots, v_{n+1} = t\}$
- $s = v_1, v_{n+1} = t$ are vertices for which we want to know the minimal weight of $\mathcal{S}$-path between them
- $E = \{(v_i, s(\omega_{ins}), v_i) | s \in \Sigma, i \in \{1, \dots, n+1\}\} \cup$
  $\{(v_i, \epsilon(\omega_{del}), v_{i+1}) | i \in \{1, \dots, n\}\} \cup$
  $\{(v_i, s'(\omega_{repl}), v_{i+1}) | s' \in \Sigma \setminus \{w_i\}, i \in \{1, \dots, n\}\}$, where $(v, s(\omega), u)$ denotes a directed edge from vertex v to vertex u labeled with s and having weight ω. Edges corresponding to the insertion procedure have weight ω_{ins} (*insertion edges*), *replacement edges* (have weight ω_{repl}) and *deletion edges* (have weight ω_{del}) are defined analogously.

For visual representation of the reduction see Fig. 5.

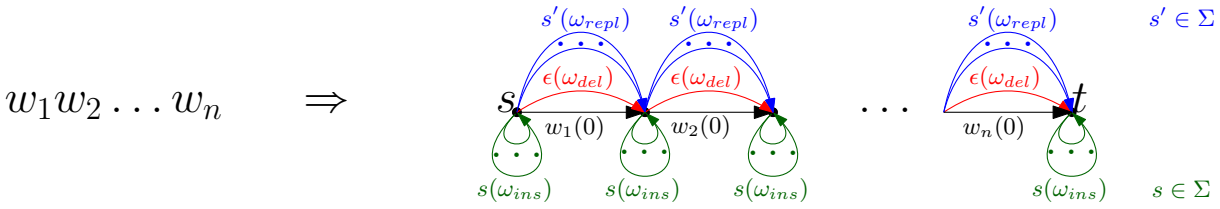

Figure 5: Reduction from LED to CFL reachability with weights. On every edge mark $x(\omega)$ means that the edge is labeled with symbol $x \in \Sigma$ and is assigned weight ω.

Correctness The size of the created directed graph is linear:

- $|V| = n + 1$
- $|E| = n \cdot (|\Sigma| + 1) + (n + 1) \cdot |\Sigma| = \mathcal{O}(n)$

Thus D can be constructed in linear time.

Note that is some operations are forbidden, we can exclude the corresponding (insertion, deletion, replacement) edges from the graph, the upper bound on size of D will remain the same.

Lemma 5.5. *Let k be the answer to instance w, G of LED problem, k' be the answer to constructed from w above instance D, G' of s-t weighted CFL reachability problem, then $k = k'$.*

Proof. We first prove that $k' \leq k$.

Let w' be the closest string to w and T be the list of transformations needed to transform w to w'. We take as an initial S-path from s to t a zero-weight path P over edges labeled with symbols of w. After that we apply every transformation from T to our path:

- If transformation inserts a symbol s after w_j we add an edge $(v_{j+1}, s(\omega_{ins}), v_{j+1})$ to our path. If a symbol is inserted in the beginning of the string we add an edge $(v_0, s(\omega_{ins}), v_0)$.
- If transformation deletes a symbol w_j we replace an edge $(v_{j-1}, w_j(0), v_j)$ with an edge $(v_{j-1}, \epsilon(\omega_{del}), v_j)$ in our path.
- If transformation replaces a symbol w_j with s' we replace an edge $(v_{j-1}, w_j(0), v_j)$ with an edge $(v_{j-1}, s'(\omega_{repl}), v_j)$ in our path.

If several symbols are inserted between w_i and w_{i+1} we suppose that the loops that were added in vertex v_{i+1} in path P are traversed by P in the same order as they are in w'.

By construction the path P now is an S-path between s and t, which labels form the word w'. P has weight k. That means that $k' \leq k$.

We proceed to the second part of the proof: $k \leq k'$.

Suppose there is an optimal S-path of weight k' — P between s and t, which labels form the word $w' \in \mathcal{L}(G)$. We translate it to the list of transformations of w to obtain w'.

- Edge $(v_i, s(\omega_{ins}), v_i)$ translates into insertion of s after w_{i-1} or before w_1 if $i = 0$.
- Edge $(v_{j-1}, \epsilon(\omega_{del}), v_j)$ in P translates into deletion of symbol w_j in w.
- Edge $(v_{j-1}, s'(\omega_{repl}), v_j)$ in P translates into replacement of w_j with s' in w.

If there were several loops through v_i in path P they are processed in the inversed order from the order in path. With this construction after application of all transformations to w we will get w' and the cost of the transformation is exactly k'. It implies that $k \leq k'$. □

Once again note, that if some operation are restricted Lemma 5.5 works.

We get the following results.

Theorem 5.6. *There is a (LED, n^c) to (weighted s-t CFL reachability, n^c), $c > 1$ fine-grained reduction.*

Proof. The construction of D takes $\mathcal{O}(n)$ time. If weighted s-t CFL reachability is solvable in $\mathcal{O}(n^{c-\epsilon}), 0 < \epsilon < c - 1$ time, then LED is solvable in $\mathcal{O}(n^{c-\epsilon'}), 0 < \epsilon' < c - 1$ time. □

In case when insertions are the only allowed operations, combining the conditional lower bound on LED based on APSP hypothesis [34, 46] with the existence of subcubic reduction from LED to weighted s-t CFL reachability we get the following corollary.

Corollary 5.4. *There is no $\mathcal{O}(n^{3-\epsilon}), \epsilon > 0$ algorithm for weighted s-t CFL reachability problem under APSP hypothesis.*

It is worth mentioning that this result applies to the subcase when the weights in weighted CFL reachability problem are restricted to only 0 and 1, i.e. there is no truly subcubic algorithm for small weights (integer weights with absolute value bounded by some constant M) CFL reachability problem. For the analogous small weight version of APSP problem (in both directed and undirected case) there are known [5] truly subcubic algorithms with time complexity $\mathcal{O}(n^{\frac{3+\omega}{2}})$. Together these two results imply that small weights CFL reachability problem in terms of fine-grained complexity is strictly harder than small weights APSP problem if APSP hypothesis it true.

5.5 Other possible reductions

The SAT, APSP and 3SUM problems are ones the most popular to get a reduction from. We have considered three problems which lower bounds are based on the SAT, APSP and 3SUM hypotheses as a possible candidates for creating a reduction.

Collecting Triangles and Δ Matching Triangles are a pair of problems, both of them have no subcubic lower bound unless all three of SETH, 3SUM and APSP hypotheses are false [2].

These problems are good in a way that a reduction from one of them would create the most believable lower bound for CFL reachability. Nevertheless creating a reduction from them seems unlikely.

Theorem 5.7. *There is no reduction from (Collecting Triangles/Δ Matching Triangles, n^3) to (CFL reachability, $n^c, c > 2$) unless at least one of the following is false:*

- *$\omega = 2$, where ω is the matrix multiplication exponent*
- *NSETH*
- *There exists an (all-pairs CFL reachability, n^c) to (s-t CFL reachability, $n^{c'}, c' > 2$) fine-grained reduction.*

Proof. First of all note that if there exists a reduction to the s-t CFL reachability, then we get an $n^c, c > 2$ lower bound via a reduction from SAT. It is known [14] that no lower bound on s-t CFL reachability better than n^ω can be achieved via a reduction from SAT unless NSETH is false. This implies that either $\omega > 2$ or NSETH is false.

Let us proceed to the case when a reduction was to the all-pairs CFL reachability. If there exists (all-pairs CFL reachability, n^c) to (s-t CFL reachability, $n^{c'}, c' > 2$) fine-grained reduction then we are in the previous case. Combining that reduction with the given reduction from Collecting Triangles/Δ Matching Triangles and with the reduction to the latters from SAT we get the desired reduction from SAT to s-t CFL reachability. □

Another problem that we have looked onto was AE-MonoΔ. As its $\mathcal{O}(n^{2.5})$ conditional lower bound [24] is based on 3SUM and APSP hypotheses, reduction from it cannot refute NSETH. We show in the end of Sec. 7 that the reduction from this problem to all-pairs CFL reachability is either nontrivial or highly unlikely.

6 CFL reachability with bounded paths length

Is is known that s-t CFL reachability on directed acyclic graph (DAG) can be solved faster that in general case:

Theorem 6.1 ([35], Th.5.9)**.** *The s-t CFL reachability on a DAG $H = (V, E)$ can be solved in $\mathcal{O}(|H|^\omega)$ time for a fixed grammar.*

The proof of this theorem is based on generalisation of Valiant's parser for CFG recognition problem. Valiant parser works as follows. It builds an $n \times n$ upper triangular matrix M where the cell (i, j) contains the set $T_{i,j}$ from CYK algorithm. Matrix is build through computation of the transitive closure of the matrix of CYK sets in the beginning of the algorithm.

It was noticed in [35] that the vertices in DAG have topological sorting and if we number the vertices according to it, the analogous M matrix is upper triangular. Thus Valiant's algorithm applies to DAGs too with minor corrections. Recall that topological sorting can be built in $\mathcal{O}(|V| + |E|) = \mathcal{O}(n^2)$ and sorting vertices according to it does not affect the time complexity of the algorithm.

With some further observations we get that after computing transitive closure M contains information about reachability of all pairs of vertices and we can formulate the following theorem.

Theorem 6.2. *The all-pairs CFL reachability problem on a DAG H can be solved in $\mathcal{O}(|H|^\omega)$ time for a fixed grammar.*

Next we use Theorem 6.2 to obtain faster all-pairs CFL reachability algorithm for finding paths of bounded length.

Theorem 6.3. *There exists an algorithm for all-edges CFL reachability with time complexity $\tilde{\mathcal{O}}(n^\omega)$ that finds all S-reachable pairs of vertices that are reachable with paths of length no more than $k = \tilde{\mathcal{O}}(1)$.*

Proof. Let $D = (V, E, L), V = \{v_1, \ldots, v_n\}$ be the given graph in all-pairs CFL reachability problem. We build a new labeled DAG $H = (V', E', L')$ of size $k = \tilde{\mathcal{O}}(n)$ as follows.

- $V' = V_1 \cup \ldots \cup V_k$, where $V_i = \{v_1^i, \ldots, v_n^i\}, i \in \{1, \ldots, k\}$ is a copy of set of vertices V.
- $E' = \cup_{i=1}^{k-1}\{(v_p^i, m_{pl}, v_l^{i+1}) | (v_p, m_{pl}, v_l) \in E)\}$, where (v_p, m_{pl}, v_l) denotes directed edge from v_p to v_l with label m_{pl}.

For visual representation of the reduction see Fig. 6.

The number of vertices in graph H is $|V'| = k \cdot n = \tilde{\mathcal{O}}(n)$. Notice that H is acyclic. Now we can apply Theorem 6.2 to H.

We extract the reachability information for each pair of vertices v_p, v_l as follows. v_l is S-reachable from v_p in G via a path of length at most k if at least one pair of vertices $(v_p^1, v_l^i), i \in \{1, \cdots, k\}$ is S-reachable in H. We need $\mathcal{O}(k)$ time to extract bounded-reachability information for every pair of vertices in G from reachability in H. Thus we

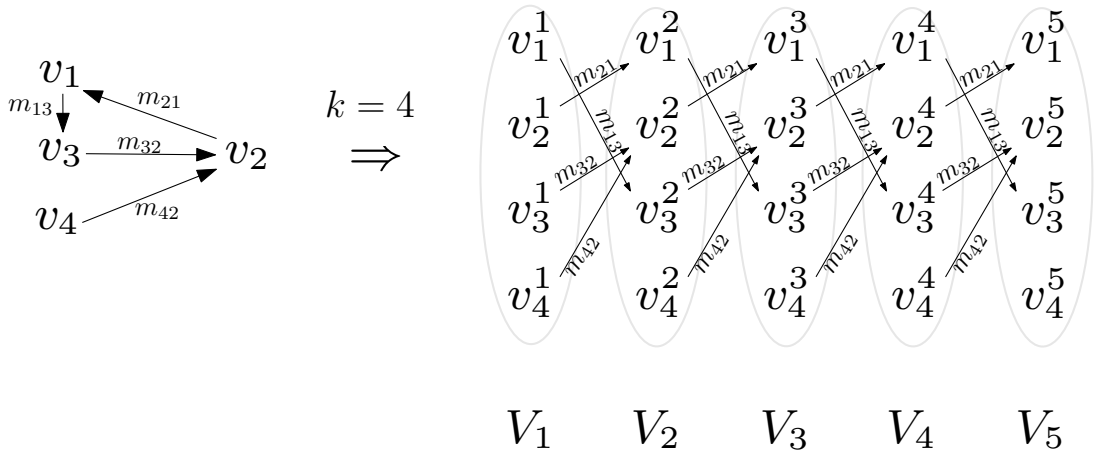

Figure 6: Example of the reduction on graph with 4 vertices and searching for CFL paths of length no more than 4.

can get all-pairs reachability information in $\tilde{\mathcal{O}}(n^2)$ and it does not affect time complexity of our algorithm.

Time complexity of the algorithm is equal to the time complexity of application Theorem 6.2 to H, which is $\mathcal{O}((n \cdot poly(\log n))^\omega) = \tilde{\mathcal{O}}(n^\omega)$. □

We can extend the theorem above to the paths of greater lengths to get an algorithm with worse but still subcubic time complexity.

Corollary 6.1. *There exists an algorithm for all-pairs CFL reachability with time complexity $\tilde{\mathcal{O}}(n^c), c < 3$ that finds all S-reachable pairs that are reachable with paths of length no more than $k = \tilde{\mathcal{O}}(n^{c'})$, where $c' < \frac{3}{\omega} - 1$.*

Proof. We build graph H in the same way as in Theorem 6.3. Graph H has $k \cdot n$ vertices.

Time complexity of application Theorem 6.2 to H is: $\mathcal{O}((k \cdot n)^\omega) = \tilde{\mathcal{O}}((n^{1+c'})^\omega) = \tilde{\mathcal{O}}((n^{(1+c')\cdot\omega})$. As $(1 + c') \cdot \omega < 3$ it is subcubic.

Time complexity of extraction the reachability information from H is: $\mathcal{O}(n^2 \cdot (k-1)) = \tilde{\mathcal{O}}(n^{2+c'})$. As $2 + c' < (1 + c') \cdot \omega < 3$ it is subcubic.

Total time complexity of the algorithm is $\tilde{\mathcal{O}}(n^c)$ for $c = (1 + c') \cdot \omega < 3$. □

Note that in CFL reachability problem some paths that we look for may be of exponential length [29]. It implies that Theorem 6.3 and Corollary 6.1 can give us a faster algorithm in a very restricted case.

7 CFL reachability on graphs with line-edges

Suppose we want to solve all-pairs CFL reachability problem on a graph D that is homeomorphic to graph $D', |V(D')| = n$ (i.e. D is a subdivision of D') but our vertices of interest in all-pairs reachability are exactly the vertices of corresponding to the vertices of D'. An example of such was in Section 5.3, see Fig. 4: we have split the transition edges

to make the system sparse, but the reachability information between auxiliary states was irrelevant to us.

In general graphs even if the number of subdivisions that we can make on each edge is bounded by constant, number of vertices can increase from $\mathcal{O}(n)$ up to $\mathcal{O}(n^2)$. In that case the application of the standard algorithms for CFL reachability will take $\tilde{\mathcal{O}}(n^6)$ amount of time. We show how to preprocess D' and the grammar if the number of subdivisions on each edge was small enough so that the time complexity of the algorithm will remain the same with respect to the polylogarithmic factors.

We say that D is a *line-graph* if D is a subdivision of graph D'. If every edge of D' was subdivided less than k times, D is called *k-line-graph*

In D we call vertices of D' *ordinary* vertices and vertices that were created through subdivision *additional* vertices. Edge e of D' after subdivision is a line-graph and is called *line-edge* of D.

All-pairs CFL reachability on line-graph D is a variation of all-pairs CFL reachability problem on D where we ask only about reachability between pairs of ordinary vertices.

k-line-graph can rise up if the mark that we want to put on each edge does not consist of one symbol, but rather a word (see Fig. 4 and edges from q_s for an example). In that case, a logical step would be to replace an edge with a word on it with a line-graph, which symbols form this word. In this case k equals the biggest length of the word on the edges of G.

Theorem 7.1. *Suppose there exists an all-pairs CFL reachability algorithm with time complexity $\mathcal{O}(n^c \cdot poly(|G|)) = \mathcal{O}(n^c)$. Then there exist all-pairs CFL reachability algorithm on k-line-graphs, $k = o(\log n)$ with time complexity $\mathcal{O}(n^c)$.*

Proof. Let $D = (V, E, L)$ be an k-line-graph graph, $k = o(\log n)$, on n ordinary vertices and $G = (N, \Sigma, P, S)$ be the grammar in the given instance of all-pairs CFL reachability problem. We transform this instance to new instance D', G' of the same problem. We suppose that G is in CNF, otherwise, it can be converted to CNF in $\mathcal{O}(poly(|G|))$ time.

The proof consists of four steps that transform the words on the line-edges so that they are the terminals in the new grammar G'. D' in the new instance is a graph, which was transformed to D by subdivisions of the edges, i.e. D' is a graph on ordinary vertices of D, vertices in D' are connected by an edge if the corresponding vertices in D are connected by an edge or a line-edge. Labels on the edges of D' are defined below together with the grammar G'.

The steps of the transformation from an instance D, G to an instance D', G' are the following:

1. First of all we make all line-edges in D of the same length.
   As D is a k-line-graph all line-edges of D consist of no more than k usual edges. We want to insert ϵ-edges in every line-edge so that the length of every line-edge is exactly $k = o(\log n)$, the resulting graph is denoted by D''. We transform the grammar G to G'' accordingly, see Sec. 5.4, note about adding ϵ-edges in the graph for the full description.
   Note that all words from $\mathcal{L}(G'')$ that we look for in all-pairs CFL reachability problem in D now have length $k \cdot i$ for some $i \geq 0$.
2. We define the new alphabet Σ' as an alphabet of k-tuples of symbols over $\Sigma \cup \{\epsilon'\}$: $\Sigma' = \{(x_1, \ldots, x_k) | x_i \in \Sigma \cup \{\epsilon'\}\ \forall i \in \{1, \ldots, k\}\}$. We mark an edge in D' by the k-

tuple $(x_1, \ldots, x_k)$ from Σ' if the marks on the corresponding transformed line-edge in D'' had marks $x_1, \ldots, x_k$ on the edges.

3. We convert CFG G'' to PDA $\mathcal{A}$ with $l = poly(|G''|)$ number of states by a standard algorithm (see [38], Lemma 2.21)
4. After that we transform PDA $\mathcal{A}$ to PDA $\mathcal{A}'$ with $l \cdot |\Sigma|^k \cdot k = o(n)$ states that recognises all words from $\mathcal{L}(G'')$ that have length $k \cdot i$ for some i and were split into k-tuples. That means that $\mathcal{A}'$ accepts the words over k-tuple language of the form $(w_1, \ldots, w_k) \ldots (w_{ik+1}, \ldots, w_{(i+1)k})$ if and only if $\mathcal{A}$ accepted the word $w_1 \ldots w_{(i+1)k}$. We transform each state q except q_0 of $\mathcal{A}$ into $|\Sigma|^k \cdot k$ states. We suppose that $\mathcal{A}$ can be in state q_0 only in the beginning of its work, otherwise we can make a duplicate state q_0' with the same transitions as q_0 where $\mathcal{A}$ will come in the middle of its work. State $q^i_{(w_1,\ldots,w_k)}$ corresponds to the moment of the system when the automaton $\mathcal{A}$ has read the i-th symbol of the tuple (if they were written as a part of usual string $\ldots w_1 \ldots w_k \ldots$) and is in state q.
   For each transition $(q, s, g, q', g') \in \delta$ we model the work of $\mathcal{A}$ and create the following transitions:

$$(q^i_{(w_1,\ldots,w_k)}, \epsilon, g, q'^{i+1}_{(w_1,\ldots,w_k)}, g') \;\; \forall 0 \le i < k \;\; \forall (w_1, \ldots, w_{i+1} = s, \ldots, w_k) \in \Sigma'$$

$$(q^k_{(w_1,\ldots,w_k)}, (w'_1, \ldots, w'_k), g, q'^0_{(w'_1,\ldots,w'_k)}, g') \;\; \forall (w_1, \ldots, w_k), (w'_1 = s, \ldots, w'_k) \in \Sigma'$$

   The set of final states consists of states:

$$q^k_{(w_1,\ldots,w_k)} \;\; \forall (w_1, \ldots, w_k) \in \Sigma' \;\; \forall q \in Q_f$$

   where Q_f is a set of final states of $\mathcal{A}$.
   Transitions in the cases concerning the starting state q_0 are similar. For transition $(q_0, s, g, q', g') \in \delta$ in $\mathcal{A}$ we create in $\mathcal{A}'$ transitions:

$$(q_0, (w_1, \ldots, w_k), g, q'^0_{(w_1,\ldots,w_k)}, g') \;\; \forall (w_1 = s, \ldots, w_k) \in \Sigma'$$

   By the construction $\mathcal{A}'$ makes the same transitions as $\mathcal{A}$, but "saves" a tuple that is read now in a memory.
5. Finally we convert PDA $\mathcal{A}'$ to CFG G' of size $|G'| = o(n)$ by a standard algorithm (see [38], Lemma 2.22)

Now one can apply algorithm for CFL reachability with grammar G' for graph D' and get the answer to the usual all-pairs CFL reachability problem in time $\mathcal{O}(n^c \cdot poly(|G'|)) = \mathcal{O}(n^c \cdot poly(o(n))) = \mathcal{O}(n^c)$. Note that $\mathcal{L}(G')$-reachable pairs of D' correspond exactly to the $\mathcal{L}(G)$-reachable pairs of D.

□

Note that Theorem 7.1 can be seen as a reformulation of a fact that if a grammar has size $o(n)$ then it does not affect the time complexity of an algorithm if it's time complexity depends polynomially on the size of the grammar, which is typical for CFL reachability algorithms. The result is useful when the words on the edges cannot be easily expressed as nonterminals of some grammar.

One can ask a question whether Theorem 7.1 can be generalized for the line-edges of the bigger length. Below we show that this kind of generalisations will be either hard to prove or highly unlikely for bigger lengths.

Let A be the algorithm for all-pairs CFL reachability with time complexity $\mathcal{O}(T(n) \cdot poly(|G|))$. As generalisation should be correct for any time complexity $T(n)$ of algorithm A we suppose that in the proof of generalisation of the theorem above, one makes several calls to algorithm A on the graphs of size $\Theta(n)$. We suppose that these calls are made on directed graphs $D_1, \ldots, D_k, k \geq 1$. By the statement of the theorem we get that $k = \mathcal{O}(poly(\log n))$. With this assumption we can formulate the following theorem.

Theorem 7.2. *Suppose claim of Theorem 7.1 holds for the k-line-graphs, $k = \Omega(\log n)$. Then either:*

- *at least one of $D_1, \ldots, D_k, k \geq 1$ has structure different from D'*
- *both APSP and 3SUM hypotheses are false*

Before the proof of the theorem we present a reduction which was proposed in [22].

Lemma 7.1. *[22] There exists an (AE-MonoΔ, n^3) to (all-pairs CFL reachability on k-line-graph, $k = \Theta(\log n)$, n^3) fine-grained reduction.*

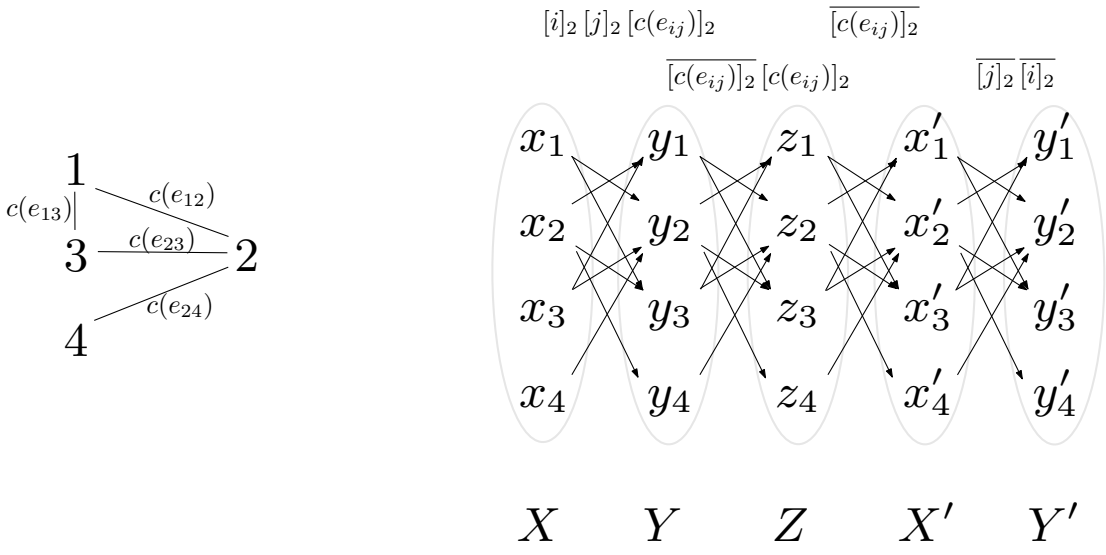

Figure 7: The reduction from an instance of a AE-MonoΔ problem with 4 vertices to the instance of all-pairs CFL reachability problem on line-graph.

Proof. The reduction and its correctness is totally the same as in the reduction from (AE-MonoΔ, $n^{2.5}$) to (sparse all-pairs PDS reachability with stack depth $b = 4\lceil \log n \rceil$, $n^{1.25}$) presented in Sec. 5.3.

Note that on every edge we get $4\lceil \log n \rceil - 1$ additional vertices and thus the constructed graph is a k-line-graph, $k = \Theta(\log n)$, with $5n = \Theta(n)$ ordinary vertices. □

Now we are ready to prove Theorem 7.2.

Proof of Theorem 7.2. Suppose claim of Theorem 7.1 works for k-line-graphs, $k = \Omega(\log n)$. We want to apply it to the graph from the reduction in Lemma 7.1 — graph D. This is a k-line-graph, $k = \Omega(\log n)$, moreover the corresponding graph D' is an acyclic graph where the length of the paths between vertices do not exceed 4.

We have seen in Sec 6 that CFL reachability on DAGs can be solved in (n^ω) time. That means that at least one of $D_1, \ldots, D_k, k \geq 1$ should be not an acyclic graph. In other case AE-MonoΔ can be solved in $\tilde{\mathcal{O}}(n^\omega)$ time by Theorem 6.2. Recall that $AE - Mono\Delta$ problem has $\mathcal{O}(n^{2.5})$ lower bound under 3SUM and APSP hypotheses. Together these two facts imply that both 3SUM and APSP hypotheses are false.

Moreover at least in one of $D_1, \ldots, D_k, k \geq 1$ the paths that we look for should be of length at least $k = \tilde{\Omega}(n^{c'}), c' \geq \frac{2.5}{\omega} - 1$ otherwise by Corollary 6.1 AE-MonoΔ can be solved in $\tilde{\mathcal{O}}(n^{2.5-\epsilon}), \epsilon > 0$ time.

In both of the cases in at least one of 3SUM and APSP hypotheses is true we need to considerably change the underlying graph D' at least once before calling algorithm A on it. □

Note that the proven theorem also helps us to understand the structure of a graph that we should get in a reduction from AE-MonoΔ to CFL reachability if there exists one. Let U be the graph on n vertices in the instance of AE-MonoΔ problem. Then if at least one of APSP and 3SUM hypotheses is correct then the graph $D = (V, E, L)$ in CFL reachability problem should contain:

- cycle and the shortest S-path of length at least $\tilde{\Omega}(1)$ between some pair of vertices (at least $\tilde{\Omega}(n^{c'}), c' < \frac{2.5}{c \cdot \omega} - 1$ if D has no more that $\mathcal{O}(n^c)$ vertices by Corollary 6.1)
- or $n' = \Omega(n^{\frac{2.5}{\omega}})$ vertices (so that $\tilde{\mathcal{O}}(n'^\omega) \geq \tilde{\mathcal{O}}(n^{2.5})$)

8 Conclusion and future work

In the course of this work, the following results were obtained:

1. We have studied the existing algorithms for CFL reachability and proposed a faster algorithm for the subcase of finding paths of bounded length (see Sec. 3.3 and 6).
2. We gave an overview on the reductions around CFL reachability problems (see Sec. 4).

3.1. Some of the techniques used in existing conditional lower bounds were translated to CFL reachability and the problems connected to it, new conditional lower bounds were achieved (see Sec. 5 and Fig. 1).

3.2. We have presented new technique that may be used in creating new conditional lower bounds on CFL reachability and discussed its limitations (see Sec 7).

3.3. Several problems with lower bounds based on APSP, 3SUM and SAT hypotheses were studied. In Sec. 5.5 and 7 we discussed reasons why the reductions from them will be either hard, unlikely or will refute some hypotheses.

There are still many open questions in the area of this work. We point out some of them.

First question considers the existence of new conditional lower bounds for CFL reachability. We want to highlight two problems as a candidates to obtain them. They are APSP problem and OMV problem. We propose APSP problem and it's many subcubic equivalent analogues [46] as APSP is connected to problems on paths, and OMV problem as it is connected to dynamic problems as is CFL reachability problem.

One of the promising way of research is creating a reduction from all-pairs to s-t CFL reachability. There are known examples (e.g. All-Pairs Negative Triangle and Negative Triangle [47]) where such problems have the same complexity. Approach that was used in Triangle-connected problems is not naively applicable to CFL reachability as it is based on splitting sets of vertices in a tripartite graph. Still some existing ideas ideas may be taken into account when creating such a reduction between CFL reachability variations.

Despite the fact that Theorem 7.2 gives the intuition of the complexities behind the generalisation of the approach with line-edges it does admit the existence of other approaches: we can prove the analogous theorem for algorithms with some specific complexities, i.e. cubic or subcubic. It would be interesting to try another way to deal with long edges.

References

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. "If the current clique algorithms are optimal, so is Valiant's parser". In: *SIAM Journal on Computing* 47.6 (2018), pp. 2527–2555.

[2] 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 Symposium on Theory of Computing*. STOC '15. Portland, Oregon, USA: Association for Computing Machinery, 2015, 41–50. ISBN: 9781450335362. DOI: 10.1145/2746539.2746594. URL: https://doi.org/10.1145/2746539.2746594.

[3] Alfred V. Aho and Thomas G. Peterson. "A Minimum Distance Error-Correcting Parser for Context-Free Languages". In: *SIAM J. Comput.* 1.4 (1972), 305–312. ISSN: 0097-5397. DOI: 10.1137/0201022. URL: https://doi.org/10.1137/0201022.

[4] Josh Alman and Virginia Vassilevska Williams. "A Refined Laser Method and Faster Matrix Multiplication". In: *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '21. Virtual Event, Virginia: Society for Industrial and Applied Mathematics, 2021, 522–539. ISBN: 9781611976465.

[5] Noga Alon, Zvi Galil, and Oded Margalit. "On the Exponent of the All Pairs Shortest Path Problem". In: *J. Comput. Syst. Sci.* 54.2 (1997), 255–262. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1388. URL: https://doi.org/10.1006/jcss.1997.1388.

[6] Phillip Bradford and David Thomas. "Labeled shortest paths in digraphs with negative and positive edge weights". In: *http://dx.doi.org/10.1051/ita/2009011* 43 (July 2009). DOI: 10.1051/ita/2009011.

[7] Phillip G Bradford. "Efficient exact paths for Dyck and semi-Dyck labeled path reachability". In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE. 2017, pp. 247–253.

[8] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. "Dynamic Matrix Inverse: Improved Algorithms and Matching Conditional Lower Bounds". In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. 2019, pp. 456–480. DOI: 10.1109/FOCS.2019.00036.

[9] Karl Bringmann. "Fine-Grained Complexity Theory". In: *36th International Symposium on Theoretical Aspects of Computer Science*. 2019, p. 1.

[10] Karl Bringmann et al. "Truly Sub-cubic Algorithms for Language Edit Distance and RNA Folding via Fast Bounded-Difference Min-Plus Product". In: *CoRR* abs/1707.05095 (2017). arXiv: 1707.05095. URL: http://arxiv.org/abs/1707.05095.

[11] Marco L. Carmosino et al. "Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-Reducibility". In: *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*. ITCS '16. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2016, 261–270. ISBN: 9781450340571. DOI: 10.1145/2840728.2840746. URL: https://doi.org/10.1145/2840728.2840746.

[12] Krishnendu Chatterjee, Bhavya Choudhary, and Andreas Pavlogiannis. "Optimal Dyck Reachability for Data-Dependence and Alias Analysis". In: *Proc. ACM Program. Lang.* 2.POPL (Dec. 2017). DOI: 10.1145/3158118. URL: https://doi.org/10.1145/3158118.

[13] Swarat Chaudhuri. "Subcubic Algorithms for Recursive State Machines". In: *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '08. San Francisco, California, USA: Association for Computing Machinery, 2008, 159–169. ISBN: 9781595936899. DOI: 10.1145/1328438.1328460. URL: https://doi.org/10.1145/1328438.1328460.

[14] Dmitry Chistikov, Rupak Majumdar, and Philipp Schepper. "Subcubic Certificates for CFL Reachability". In: *arXiv preprint arXiv:2102.13095* (2021).

[15] Noam Chomsky. "On certain formal properties of grammars". In: *Information and Control* 2.2 (1959), pp. 137–167. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(59)90362-6. URL: https://www.sciencedirect.com/science/article/pii/S0019995859903626.

[16] Kathrin Hanauer, Monika Henzinger, and Christian Schulz. "Faster Fully Dynamic Transitive Closure in Practice". In: *ArXiv* abs/2002.00813 (2020).

[17] Jakob Cetti Hansen, Adam Husted Kjelstrøm, and Andreas Pavlogiannis. "Tight bounds for reachability problems on one-counter and pushdown systems". In: *Information Processing Letters* 171 (2021), p. 106135.

[18] Nevin Heintze and David McAllester. "On the Cubic Bottleneck in Subtyping and Flow Analysis". In: *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*. LICS '97. USA: IEEE Computer Society, 1997, p. 342. ISBN: 0818679255.

[19] Monika Henzinger et al. "Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture". In: *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '15. Portland, Oregon, USA: Association for Computing Machinery, 2015, 21–30. ISBN: 9781450335362. DOI: 10.1145/2746539.2746609. URL: https://doi.org/10.1145/2746539.2746609.

[20] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN: 0321455363.

[21] Russell Impagliazzo and Ramamohan Paturi. "On the Complexity of K-SAT". In: *J. Comput. Syst. Sci.* 62.2 (2001), 367–375. ISSN: 0022-0000. DOI: 10.1006/jcss.2000.1727. URL: https://doi.org/10.1006/jcss.2000.1727.

[22] Paraschos Koutris and Shaleen Deep. "The Fine-Grained Complexity of CFL Reachability". In: *Proc. ACM Program. Lang.* 7.POPL (2023). DOI: 10.1145/3571252. URL: https://doi.org/10.1145/3571252.

[23] Lillian Lee. "Fast Context-Free Grammar Parsing Requires Fast Boolean Matrix Multiplication". In: *J. ACM* 49.1 (Jan. 2002), 1–15. ISSN: 0004-5411. DOI: 10.1145/505241.505242. URL: https://doi.org/10.1145/505241.505242.

[24] Andrea Lincoln, Adam Polak, and Virginia Vassilevska Williams. "Monochromatic Triangles, Intermediate Matrix Products, and Convolutions". en. In: (2020). DOI: 10.4230/LIPICS.ITCS.2020.53. URL: https://drops.dagstuhl.de/opus/volltexte/2020/11738/.

[25] Anders Alnor Mathiasen and Andreas Pavlogiannis. "The Fine-Grained and Parallel Complexity of Andersen's Pointer Analysis". In: *Proc. ACM Program. Lang.* 5.POPL (Jan. 2021). DOI: 10.1145/3434315. URL: https://doi.org/10.1145/3434315.

[26] G. Matthews. "Chomsky N. and Schützenberger M. P.. The algebraic theory of context-free languages. Computer programming and formal systems, edited by Braffort P. and Hirschberg D., Studies in logic and the foundations of mathematics, North-Holland Publishing Company, Amsterdam 1963, pp. 118–161." In: *The Journal of Symbolic Logic* 32 (Oct. 2014), pp. 388–389. DOI: 10.2307/2270782.

[27] David Melski and Thomas Reps. "Interconvertbility of Set Constraints and Context-Free Language Reachability". In: *SIGPLAN Not.* 32.12 (Dec. 1997), 74–89. ISSN: 0362-1340. DOI: 10.1145/258994.259006. URL: https://doi.org/10.1145/258994.259006.

[28] Gene Myers. "Approximately Matching Context-Free Languages". In: *Inf. Process. Lett.* 54.2 (1995), 85–92. ISSN: 0020-0190. DOI: 10.1016/0020-0190(95)00007-Y. URL: https://doi.org/10.1016/0020-0190(95)00007-Y.

[29] Laurent Pierre. "Rational indexes of generators of the cone of context-free languages". In: *Theoretical Computer Science* 95.2 (1992), pp. 279–305. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(92)90269-L. URL: https://www.sciencedirect.com/science/article/pii/030439759290269L.

[30] Jakob Rehof and Manuel Fähndrich. "Type-Base Flow Analysis: From Polymorphic Subtyping to CFL-Reachability". In: *SIGPLAN Not.* 36.3 (Jan. 2001), 54–66. ISSN: 0362-1340. DOI: 10.1145/373243.360208. URL: https://doi.org/10.1145/373243.360208.

[31] Thomas Reps. "Program analysis via graph reachability1An abbreviated version of this paper appeared as an invited paper in the Proceedings of the 1997 International Symposium on Logic Programming [84].1". In: *Information and Software Technology* 40.11 (1998), pp. 701–726. ISSN: 0950-5849. DOI: https://doi.org/10.1016/S0950-5849(98)00093-7. URL: https://www.sciencedirect.com/science/article/pii/S0950584998000937.

[32] Thomas Reps, Susan Horwitz, and Mooly Sagiv. “Precise Interprocedural Dataflow Analysis via Graph Reachability”. In: *Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL ’95. San Francisco, California, USA: Association for Computing Machinery, 1995, 49–61. ISBN: 0897916921. DOI: 10.1145/199448.199462. URL: https://doi.org/10.1145/199448.199462.

[33] Walter L. Ruzzo. “On the Complexity of General Context-Free Language Parsing and Recognition (Extended Abstract)”. In: *Proceedings of the 6th Colloquium, on Automata, Languages and Programming*. Berlin, Heidelberg: Springer-Verlag, 1979, 489–497. ISBN: 3540095101.

[34] Barna Saha. “Faster Language Edit Distance, Connection to All-pairs Shortest Paths and Related Problems”. In: *CoRR* abs/1411.7315 (2014). arXiv: 1411.7315. URL: http://arxiv.org/abs/1411.7315.

[35] Philipp Johann Schepper. “The Complexity of Formal Language Decision Problems”. In: (2018).

[36] Petteri Sevon and Lauri Eronen. “Subgraph Queries by Context-free Grammars”. In: *Journal of Integrative Bioinformatics* 5.2 (2008), pp. 157 –172. DOI: https://doi.org/10.1515/jib-2008-100. URL: https://www.degruyter.com/view/journals/jib/5/2/article-p157.xml.

[37] Ekaterina Shemetova et al. *One Algorithm to Evaluate Them All: Unified Linear Algebra Based Approach to Evaluate Both Regular and Context-Free Path Queries*. 2021. arXiv: 2103.14688 [cs.DB].

[38] Michael Sipser. *Introduction to the Theory of Computation*. 1st. International Thomson Publishing, 1996. ISBN: 053494728X.

[39] Manu Sridharan and Rastislav Bodík. “Refinement-Based Context-Sensitive Points-to Analysis for Java”. In: *SIGPLAN Not.* 41.6 (June 2006), 387–400. ISSN: 0362-1340. DOI: 10.1145/1133255.1134027. URL: https://doi.org/10.1145/1133255.1134027.

[40] Manu Sridharan et al. “Demand-Driven Points-to Analysis for Java”. In: *SIGPLAN Not.* 40.10 (Oct. 2005), 59–76. ISSN: 0362-1340. DOI: 10.1145/1103845.1094817. URL: https://doi.org/10.1145/1103845.1094817.

[41] Leslie G Valiant. “General context-free recognition in less than cubic time”. In: *Journal of computer and system sciences* 10.2 (1975), pp. 308–315.

[42] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. "Finding Heaviest H-Subgraphs in Real Weighted Graphs, with Applications". In: *ACM Trans. Algorithms* 6.3 (2010). ISSN: 1549-6325. DOI: 10.1145/1798596.1798597. URL: https://doi.org/10.1145/1798596.1798597.

[43] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. "Finding the Smallest H-Subgraph in Real Weighted Graphs and Related Problems". In: *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part I*. ICALP'06. Venice, Italy: Springer-Verlag, 2006, 262–273. ISBN: 3540359044. DOI: 10.1007/11786986_24. URL: https://doi.org/10.1007/11786986_24.

[44] Ryan Williams. "A New Algorithm for Optimal 2-Constraint Satisfaction and Its Implications". In: *Theor. Comput. Sci.* 348.2 (2005), 357–365. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2005.09.023. URL: https://doi.org/10.1016/j.tcs.2005.09.023.

[45] Virginia Vassilevska Williams. "On some fine-grained questions in algorithms and complexity". In: *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*. World Scientific. 2018, pp. 3447–3487.

[46] Virginia Vassilevska Williams and R. Ryan Williams. "Subcubic Equivalences Between Path, Matrix, and Triangle Problems". In: *J. ACM* 65.5 (Aug. 2018). ISSN: 0004-5411. DOI: 10.1145/3186893. URL: https://doi.org/10.1145/3186893.

[47] Virginia Vassilevska Williams and Ryan Williams. "Triangle Detection Versus Matrix Multiplication : A Study of Truly Subcubic Reducibility ". In: 2009.

[48] Mihalis Yannakakis. "Graph-Theoretic Methods in Database Theory". In: *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS '90. Nashville, Tennessee, USA: Association for Computing Machinery, 1990, 230–242. ISBN: 0897913523. DOI: 10.1145/298514.298576. URL: https://doi.org/10.1145/298514.298576.