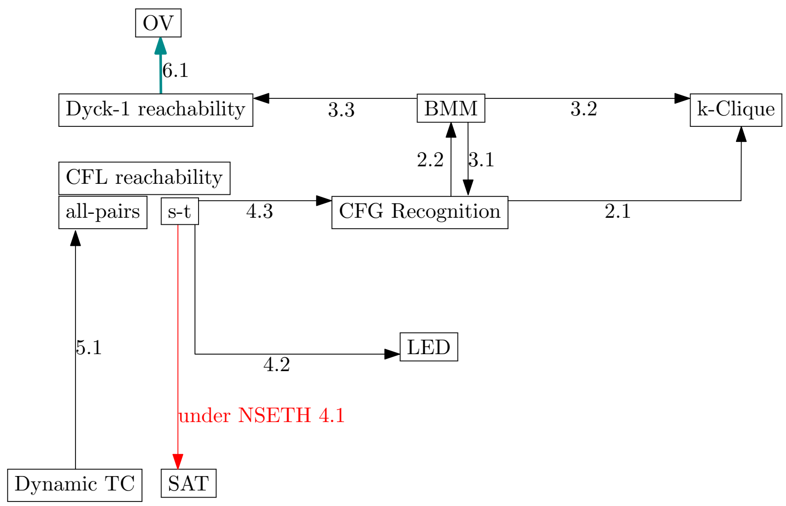
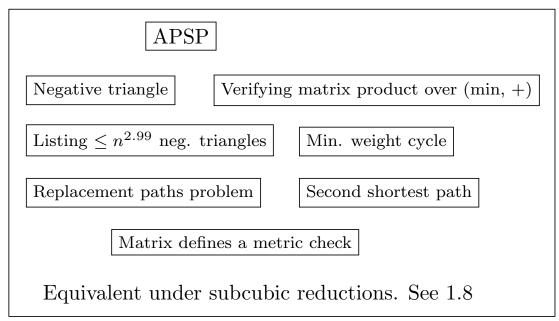
Reduction map





$A \to B \iff$ B has reduction to A
$\longrightarrow$ – no reduction
hypothesis

1. Problem definitions

1.1. **CFG recognition:** given a CFG $\mathcal{G}$ and a string $w \in \Sigma^*$ determine if w can be obtained from G (i.e. whether $w \in \mathcal{L}(\mathcal{G})$)

Best upper bound [12]: $\mathcal{O}(n^\omega \cdot |\mathcal{G}|^2)$

Best combinatorial algorithms: $\mathcal{O}(n^3/\log^3 n)$

CFG parsing problem: if $w \in \mathcal{L}(\mathcal{G})$, output a possible derivation sequence.

CFG recognition is as hard as CFG parsing up to logarithmic factors [11].

1.2. **CFL reachability:** given a CFG $\mathcal{L}$ and a labelled graph G

1.3. **k-Clique:** decide whether a given undirected unweighted graph on n nodes and $\mathcal{O}(n^2)$ edges contains a clique on k nodes.

It is a parameterized version of the NP-hard Max-Clique problem.

Let $0 \leq F \leq \omega$ and $0 \leq C \leq 3$ be the smallest numbers such that $3k$-Clique can be solved combinatorially in $O(n^{Ck})$ time and in $O(n^{Fk})$ time by any algorithm, for any (large enough) constant $k \geq 1$. A conjecture in graph algorithms and parameterized complexity is that $C = 3$ and $F = \omega$.

1.4. **SAT.**

1.4.1. *k-SAT:.* Given a k-CNF formula ϕ on n variables. Is there an assignment to the variables that satisfies ϕ?

1.4.2. *ETH (Exponential time hypothesis):* There is no algorithm that solves k-SAT in time $2^{o(n)}$ for any k.

1.4.3. *SETH (Strong exponential time hypothesis):* There is no $\epsilon > 0$ such that k-SAT can be solved in time $2^{(1-\epsilon)n}$ for any k.

1.4.4. *NSETH (Nondeterministic strong exponential time hypothesis):* There is no $\epsilon > 0$ such that k-SAT can be solved in time $2^{(1-\epsilon)n}$ co-nondeterministically for any k.

1.5. **BMM.** Given two $n \times n$ matrices A, B over $\{0, 1\}$, Boolean Matrix Multiplication (BMM) is $(AB)[i, j] = \vee_{k=1}^{n}(A(i, k) \wedge B(k, j))$.

Note that BMM can be computed using an algorithm for integer matrix multiplication in $\mathcal{O}(n^{\omega})$ time. Still, no combinatorial $\mathcal{O}(n^{3-\epsilon})$ algorithm is known.

1.5.1. *BMM conjecture [2].* In the Word RAM model with words of $\mathcal{O}(\log n)$ bits, any combinatorial algorithm requires $n^{3-o(1)}$ time in expectation to compute the boolean product of two $n \times n$ matrices.

1.6. **LED.** Given a string $w \in \Sigma^*$ and a context-free grammar $\mathcal{G}$ defined over the same alphabet, how many minimum number of repairs (insertions, deletions and substitutions) are required to map w into a valid member of $\mathcal{G}$?

1.7. **3SUM.** Determine whether a set $S \subset \{-n^3, \dots, n^3\}$ of $|S| = n$ integers contains three distinct elements $a, b, c \in S$ with $a + b = c$.

1.7.1. *3SUM conjecture.* In the Word RAM model with words of $\mathcal{O}(\log n)$ bits, any algorithm requires $n^{2-o(1)}$ time in expectation to solve 3SUM problem.

1.8. **Triangle detection.**

1.8.1. *Triangle conjecture.* There is a constant $\delta > 0$, such that in the Word RAM model with words of $\mathcal{O}(\log n)$ bits, any algorithm requires $m^{1+\delta-o(1)}$ time in expectation to detect whether an m edge graph contains a triangle.

1.9. **APSP (all pairs shortest paths) and others [13]:**

- The all-pairs shortest paths problem on weighted digraphs (APSP).
- Detecting if a weighted graph has a triangle of negative total edge weight.
- Listing up to $n^{2.99}$ negative triangles in an edge-weighted graph.
- Finding a minimum weight cycle in a graph of non-negative edge weights.
- The replacement paths problem on weighted digraphs.
- Finding the second shortest simple path between two nodes in a weighted digraph.
- Checking whether a given matrix defines a metric.
- Verifying the correctness of a matrix product over the $(min, +)$-semiring (distance product).

1.9.1. *APSP conjecture.* There is a constant c, such that in the Word RAM model with words of $\mathcal{O}(\log n)$ bits, any algorithm requires $n^{3-o(1)}$ time in expectation to compute the distances between every pair of vertices in an n node graph with edge weights in $\{1, \dots, n^c\}$.

1.9.2. *Negative Triangles Over Structures.* The negative triangle problem over $\mathcal{R}$ is defined on a weighted tripartite graph with parts I, J, K. Edge weights between I and J are from $\mathbb{Z}$, and all other edge weights are from $\mathcal{R}$. The problem is to detect if there are $i \in I, j \in J, k \in K$ so that $(w(i,k) \bigodot w(k,j)) + w(i,j) < 0$. Note that if one negates all weights of edges between I and J, the condition becomes $(w(i,k) \bigodot (k,j)) < w(i,j)$.

In the special case when $\bigodot = +$ and $\mathcal{R} \subseteq \mathbb{Z} \cup \{- - \infty, \infty\}$, the tripartiteness requirement is unnecessary, and the negative triangle problem is defined on an arbitrary graph with edge weights from $\mathbb{Z} \cup \{- - \infty, \infty\}$. This holds for the negative triangle problem over both the $(min, +)$ and Boolean semirings.

Note that detecting and finding one negative triangle in a graph problems reduce one to another:

Lemma (Folklore)**.** *Let $T(n)$ be a function so that $\frac{T(n)}{n}$ is nondecreasing. If there is a $T(n)$ time algorithm for negative triangle detection over $\mathcal{R}$ on a graph $G = (I \cup J \cup K, E)$, then there is an $\mathcal{O}(T(n))$ algorithm which returns a negative triangle over $\mathcal{R}$ in G if one exists.*

1.10. **OV.** The input to the Orthogonal Vectors (OV) problem is two sets X, Y, each containing n vectors in $\{0,1\}^D$, for some dimension $D = \omega(\log n)$. The task is to determine if there exists a pair $(x, y) \in (X, Y)$ that is orthogonal, i.e., for each $j \in D$, we have $x[j] \cdot y[j] = 0$.

The respective hypothesis states that the problem cannot be solved in time $\mathcal{O}(n^{2-\epsilon})$, for any fixed $\epsilon > 0$.

2. To CFG reconngnition

2.1. **From k-Clique problem [1]:**

Summary: $T(n) \Rightarrow \mathcal{O}(T(n^{k/3+1})$ $(|G| = \mathcal{O}(1))$

2.2. **From BMM [7]:**

Summary: $\mathcal{O}(|G| \cdot n^{3-\epsilon}) \Rightarrow \mathcal{O}(m^{3-\epsilon/3})$ $(|G| = \Omega(n^6)!)$, combinatorial reduction

3. To BMM

3.1. **From CFG recognition [12]:**

Summary: $\mathcal{O}(n^\omega) \Rightarrow \mathcal{O}(n^\omega)$

3.2. **From k-Clique [9], [6]:**

Summary: $\mathcal{O}(n^\omega) \Rightarrow \mathcal{O}(n^{i+\omega \cdot l}), k = 3 \cdot l + i, i = 0, 1, 2$

or $\mathcal{O}(n^{\omega(\lfloor k/3 \rfloor, \lfloor (k-1)/3 \rfloor, \lfloor k/3 \rfloor)}$, where $\mathcal{O}(n^{\omega(r,s,t)})$ denotes the running time of the multiplication of an $n^r \times n^s$ matrix by an $n^s \times n^t$ matrix.

Note that detecting arbitrary k-vertex (induced or not) subgraph in n-vertex graph is of the same complexity as detecting k-clique [9].

3.3. **From Dyck-1 reachability [3], [8]:**

Summary: $\mathcal{O}(n^\omega) \Rightarrow \mathcal{O}(n^\omega \cdot \log^2 n)$, combinatorial reduction [8]

Idea: find bell-shaped paths in $\mathcal{O}(\log n n^\omega)$ by getting $\mathcal{O}(\log n)$ transitive closures; combine Dyck-1 path from bell-shaped paths: in each of $\mathcal{O}(\log n)$ iterations we remove bell-shaped paths of height at most $2^i, i = 1, \ldots, \log n$.

or $\mathcal{O}(n^\omega) \Rightarrow \mathcal{O}(n^\omega \cdot \log^3 n)$ in [3] via algebraic matrix encoding of flat Dyck1-Reachability to $\mathcal{O}(\log n)$ AGMY matrix multiplications.

4. To CFL reachability

4.1. **To s-t reachability from SAT [5]:**

Summary: subcubic certificates for CFL reachability (positive and negative), no reductions under NSETH from SAT (SETH) to CFL reachability.

Proof is based on the following lemma, where by instance (G, λ, s, t) of CFL reachability problem G is a graph, λ is edge-labelling function, s, t are vertices between which we search a path labelled with word from the language.

Lemma. *Let (G, λ, s, t) be an instance of the CFL reachability problem. There is a linear-time reduction (in the bit-size of the input) to an instance (G', λ', s', t') of the Dyck-2 reachability problem.*

4.2. **To s-t reachability from LED.** Naive reduction: build a path graph labelled with letters of w, add loops marked with every symbol of Σ on every vertex, add edges marked with every symbol of Σ and ϵ between pairs of adjacent vertices. Make original edges of zero weight, edges with Σ and ϵ symbols of positive weight (you can choose different weight-cost for different operations). Find path between end vertices marked with word formed from grammar with minimum weight.

4.3. **To s-t reachability from CFG recognition [4]:**
Summary: $\mathcal{O}(n^{3-\epsilon}) \Rightarrow \mathcal{O}(n^{3-\epsilon})$, combinatorial

Lemma. *If there exists a combinatorial algorithm that solves the pair Dyck reachability problem in time $T(n)$, where n is the number of nodes of the input graph, then there exists a combinatorial algorithm that solves the CFL parsing problem in time $\mathcal{O}(n + T(n))$.*

In combination with reduction to BMM we have:

Theorem (BMM-hardness: Conditional cubic lower bound)**.** *For any fixed $\epsilon > 0$, if there is a combinatorial algorithm that solves the pair Dyck reachability problem in $\mathcal{O}(n^{3-\epsilon})$ time, then there is a combinatorial algorithm that solves Boolean Matrix Multiplication in $\mathcal{O}(n^{3-\epsilon})$ time.*

5. To Dynamic TC

5.1. **From all-pairs CFL reachability [10]:**
Summary: $\mathcal{O}(n^{3-\epsilon}) \Rightarrow \mathcal{O}(n^{3-\epsilon})$

6. To Dyck-1 Reachability

6.1. **From OV (inspired by [8]):**
Not working idea!

Summary: $\mathcal{O}(n^{2-\epsilon}) \Rightarrow \mathcal{O}(n^{2-\epsilon})$

Intuition. Without loss of generality suppose D is even. Consider the first two vectors of the given sets: $x_1 \in X, y_1 \in Y$. Let us build part of the Dyck-1 graph G that will check orthogonality of x_1, y_1. We create two vertices a_1^1, u_1^1, with the goal that there exist Dyck-1 path from a_1^1 to u_1^1 if and only if $x_1^1 \cdot y_1^1 = 0$. For this purpose we create two additional vertices $z^1, \tilde{z}^1$ (which will be common for all pairs of vectors from $X \times Y$) and add edges $a_1^1 \xrightarrow{(} z^1$ and $\tilde{z}^1 \xrightarrow{)} u_1^1$. Also, if $x_1^1 = 0$, we add edge $a_1^1 \xrightarrow{(} \tilde{z}^1$ and, similarly, if $y_1^1 = 0$, we add edge $z^1 \xrightarrow{)} u_1^1$ (see Fig. 6.1). Observe that if and only if $x_1^1 \cdot y_1^1 = 0$ we have path labelled $() \in Dyck-1$ from a_1^1 to u_1^1.

For consistency (of the case $D = 1$) we add vertex v_1^1 corresponding to u_1^1 and we want to maintain the property that there is a path from a_1^1 to v_1^1 labelled with Dyck-1 word if and only if $x_1^1 \cdot y_1^1 = 0$. For that we add the edge $u_1^1 \xrightarrow{)} v_1^1$ and edge $u_1^1 \xrightarrow{()} a_1^1$. The only way to reach v_1^1 from a_1^1 if still through z^1 or $\tilde{z}^1$ and the path $a_1^1(z^1/\tilde{z}^1)u_1^1a_1^1(z^1/\tilde{z}^1)u_1^1v_1^1$ gives us the desired Dyck-1 word — $()(())$.

Now we proceed with checking the second coordinates — $x_1^2 \cdot y_1^2 = 0$. We create vertices $a_1^2, u_1^2, z^2, \tilde{z}^2$, but this time we add edges $u_1^2 \xrightarrow{(} \tilde{z}^2$ and $z^2 \xrightarrow{)} a_1^2$. Edges $u_1^2 \xrightarrow{(} z^2$ and $\tilde{z}^2 \xrightarrow{)} a_1^2$ are added if the corresponding coordinates equal 0. To connect with the previous part of the graph we add edges $a_1^2 \xrightarrow{()} a_1^1$ and $v_1^1 \xrightarrow{(} u_1^2$. We also create vertex b_1^2 and an edge $a_1^2 \xrightarrow{)} b_1^2$.

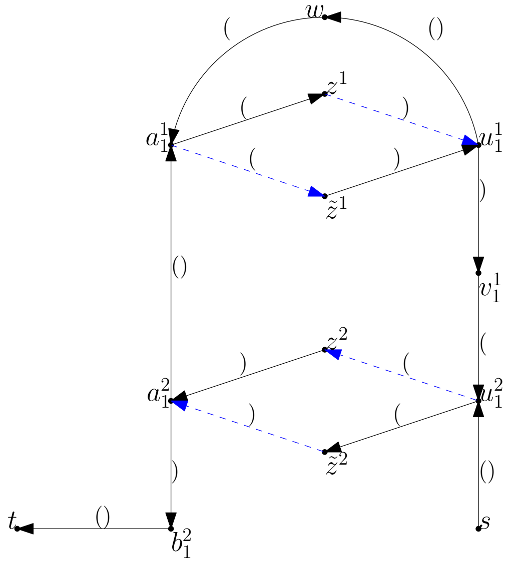


FIGURE 1. Part of the reduction graph G for vectors $x_1, y_1 \in \{0,1\}^D, D = 2$.
Blue edges exist if the corresponding coordinates equal 0.

The desired property is that we have a Dyck-1 path from u_1^2 to a_1^2 if and only if $x_1^1 \cdot y_1^1 = 0$ and $x_1^2 \cdot y_1^2 = 0$. Path from u_1^2 to a_1^2 can exist only when $x_1^2 \cdot y_1^2 = 0$. If $x_1^1 \cdot y_1^1 = 0$, the the is Dyck-1 path from a_1^1 to v_1^1 and the path $u_1^2(z^2/\tilde{z}^2)a_1^2a_1^1 \xrightarrow{\text{Dyck-1}} v_1^1u_1^2(z^2/\tilde{z}^2)a_1^2b_1^2$ gives us the desired Dyck-1 word — $()()S_1(()), S_1 \in Dyck - 1$. If $x_1^1 \cdot y_1^1 \neq 0$, then the only possible path between u_1^2 and a_1^2 is a path $u_1^2(z^2/\tilde{z}^2)a_1^2b_1^2$ which labelled with Dyck-1 word.

For the rest of the coordinates of x_1, y_1 we create vertices analogously, following the idea the Dyck-1 path from u_1^i to a_1^i (for even i; from a_1^i to u_1^i for odd i) exists if and only if $x_1^i \cdot y_1^i = 0$ and the is a Dyck-1 path from a_1^{i-1} to u_1^{i-1} (or from u_1^{i-1} to a_1^{i-1}). Remark the to get an additional "(" to compensate the ")" from the edge $a_1^i \xrightarrow{)} b_1^i$ (or $u_1^i \xrightarrow{)} v_1^i$) we need to make a loop through a_1^{i-1}, u_1^{i-1} and reach the edge $u_1^1 \xrightarrow{)} a_1^1$ because before it all the brackets are balanced.

After building the parts of G for all vectors from X, Y we add two vertices s, t and edges $s \xrightarrow{()} u_i^D$ for all $u_i \in Y$, $b_j^D \xrightarrow{()} t$ for all $a_j \in X$. Consequently there is a Dyck-1 path from s to t through the vertices u_k^D, b_l^D if and only if x_k and y_l are orthogonal.

Problem: path from z^i to z^j can be walked using edged corresponding to more than one vector. Example: $su_1^2\tilde{z}^2a_6^2a_6^1\tilde{z}^1u_1^1wa_6^1\tilde{z}^1u_1^1v_1^1u_1^2z_2a_1^2b_1^2t$ and $a_1 = (1,0), a_6 = (0,1), u_1 = (1,1)$

Formal construction Recall, that without loss of generality we suppose that D is even. We build labelled graph G of Dyck-1 problem consisting of the following vertices and edges. First, we introduce vertices $z^1, \dots, z^n$ and $\tilde{z}^1, \dots, \tilde{z}^n$.

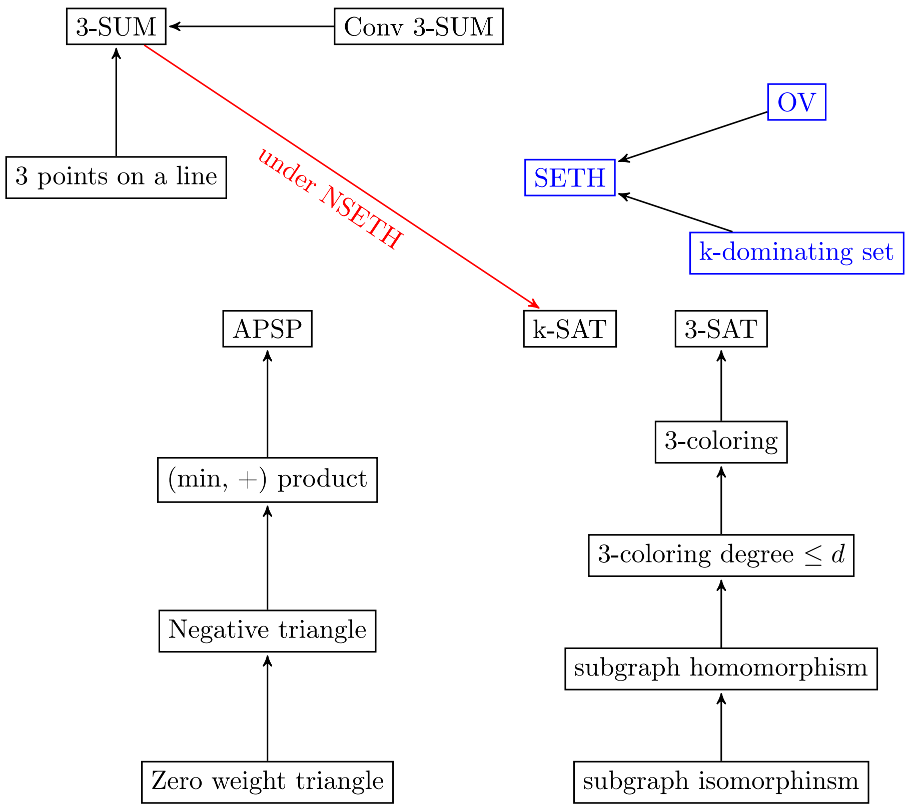
For every vector $x_i \in X$ we introduce vertices $a_i^1, \dots, a_i^D$, vertices $b_i^2, b_i^4, \dots, b_i^D$ and edges:

- $w \xrightarrow{(} a_i^1$
- $a_i^j \xrightarrow{(} z^j$ for every odd j
- $a_i^j \xrightarrow{(} \tilde{z}^j$ for every odd j if $x_i^j = 0$
- $z^j \xrightarrow{)} a_i^j$ for every even j
- $\tilde{z}^j \xrightarrow{)} a_i^j$ for every even j if $x_i^j = 0$
- $a_i^j \xrightarrow{()} a_i^{j-1}$ for every even j
- $a_i^j \xrightarrow{)} b_i^j, b_i^j \xrightarrow{(} a_i^{j+1}$ for every even j

For every vector $y_i \in Y$ we introduce vertices $u_i^1, \dots, u_i^D$, vertices $v_i^1, v_i^3, \dots, v_i^{D-1}$ and edges:

- $u_i^1 \xrightarrow{()} w$
- $z^j \xrightarrow{(} u_i^j$ for every odd j if $y_i^j = 0$
- $\tilde{z}^j \xrightarrow{(} u_i^j$ for every odd j
- $u_i^j \xrightarrow{(} z^j$ for every even j if $y_i^j = 0$
- $u_i^j \xrightarrow{(} \tilde{z}^j$ for every even j
- $u_i^{j+1} \xrightarrow{()} u_i^j$ for every even j
- $u_i^j \xrightarrow{)} v_i^j, v_i^j \xrightarrow{(} u_i^{j+1}$ for every odd j

Finally, we introduce vertices s, t and edges $b_i^D \xrightarrow{()} t, s \xrightarrow{()} u_i^D, i \in 1, \ldots, n$.
Note that $|G| = \mathcal{O}(n \cdot D)$.



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 and Virginia Vassilevska Williams. "Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems". In: *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. FOCS '14. USA: IEEE Computer Society, 2014, 434–443. ISBN: 9781479965175. DOI: `10.1109/FOCS.2014.53`. URL: `https://doi.org/10.1109/FOCS.2014.53`.

[3] 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.

[4] 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`.

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

[6] Friedrich Eisenbrand and Fabrizio Grandoni. "On the Complexity of Fixed Parameter Clique and Dominating Set". In: *Theor. Comput. Sci.* 326.1–3 (Oct. 2004), 57–67. ISSN: 0304-3975. DOI: `10.1016/j.tcs.2004.05.009`. URL: `https://doi.org/10.1016/j.tcs.2004.05.009`.

[7] 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`.

[8] 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`.

[9] Jaroslav Nešetřil and Svatopluk Poljak. “On the complexity of the subgraph problem”. In: *Commentationes Mathematicae Universitatis Carolinae* 26.2 (1985), pp. 415–419.

[10] Egor Orachev et al. “Context-Free Path Querying by Kronecker Product”. In: Aug. 2020, pp. 49–59. ISBN: 978-3-030-54831-5. DOI: 10.1007/978-3-030-54832-2_6.

[11] 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.

[12] 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.

[13] 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.