Parity Separation: A Scientifically Proven Method for Permanent Weight Loss*

Radu Curticapean

Simons Institute for the Theory of Computing, UC Berkeley, USA; and
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
radu.curticapean@gmail.com

Abstract

Given an edge-weighted graph G, let PerfMatch(G) denote the weighted sum over all perfect matchings M in G, weighting each matching M by the product of weights of edges in M. If G is unweighted, this plainly counts the perfect matchings of G.

In this paper, we introduce parity separation, a new method for reducing PerfMatch to unweighted instances: For graphs G with edge-weights 1 and -1, we construct two unweighted graphs G1 and G2 such that PerfMatch(G) = PerfMatch(G1) − PerfMatch(G2). This yields a novel weight removal technique for counting perfect matchings, in addition to those known from classical #P-hardness proofs. Our technique is based upon the Holant framework and matchgates. We derive the following applications:

Firstly, an alternative #P-completeness proof for counting unweighted perfect matchings.

Secondly, C=P-completeness for deciding whether two given unweighted graphs have the same number of perfect matchings. To the best of our knowledge, this is the first C=P-completeness result for the "equality-testing version" of any natural counting problem that is not already #P-hard under parsimonious reductions.

Thirdly, an alternative tight lower bound for counting unweighted perfect matchings under the counting exponential-time hypothesis #ETH.

1998 ACM Subject Classification G.2.1 Combinatorics, F.1.3 Complexity Measures and Classes

Keywords and phrases perfect matchings, counting complexity, structural complexity, exponential-time hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.47

1 Introduction

The problem of counting perfect matchings has played a central role in counting complexity since Valiant [27] introduced the class #P and established #P-completeness of counting perfect matchings in unweighted bipartite graphs. This problem was previously already considered in statistical physics [26, 21, 22] and Valiant's computational hardness result explains the lack of progress encountered in this area for finding efficient algorithms for counting perfect matchings.

As complexity theorists, we can appreciate this seminal #P-completeness result from another perspective: The problem of counting perfect matchings in *unweighted* graphs presented the first example of a natural hard counting problem with an easy decision version, since Edmond's classical algorithm [14] allows to decide in polynomial time whether a graph

* Part of this work was carried out while the author was a PhD student at the Department of Computer Science at Saarland University, Saarbrücken, Germany. It also appears in his PhD thesis [11] and [10].

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 47; pp. 47:1–47:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

contains *at least one* perfect matching. This showed exemplarily that the complexity-theoretic study of counting problems amounts to more than merely checking whether NP-hardness proofs for decision problems carry over to their counting versions.

For instance, a fundamental peculiarity of counting problems that is not shared by decision problems are *cancellations*: In (weighted) counting problems, witness structures may cancel each other out, and this can have strong effects on the complexity of the problem. The most prominent example of this phenomenon might be the situation of the *determinant* and the *permanent*, both summing over the same permutations, however with different weights. This results in the permanent being #P-complete by Valiant's result, whereas the determinant can be computed in polynomial time. The *accidental* and *holographic* algorithms introduced by Valiant [28, 29] provide examples for further and more unexpected cancellations that render counting problems easy.

However, cancellations are also crucial for negative results: In many #P-hardness proofs, such as [4, 2, 3], we first define an intermediate variant of the target problem on weights ± 1. Examples for this strategy include the original reduction from #SAT to counting unweighted perfect matchings [27]: In this setting, let G be a graph with edge-weights $w : E(G) \to \{-1, 1\}$, let $\mathcal{PM}[G]$ denote its set of perfect matchings, and define

$$\text{PerfMatch}(G) := \sum_{M \in \mathcal{PM}[G]} \prod_{e \in M} w(e). \tag{1}$$

Given an instance to this weighted problem, that is, a graph G derived from a 3-CNF formula, its space of witness structures $\mathcal{PM}[G]$ can then be partitioned into "good" structures that correspond to satisfying assignments, and "bad" structures that could be called combinatorial noise. By careful construction of such a graph G on edge-weights ± 1, we can ensure that bad structures come in pairs of weight $+1$ and -1, thus canceling out, whereas good structures all have weight $+1$.

To conclude #P-completeness of counting unweighted perfect matchings, it remains to simulate the weight -1 from the intermediate problem. This can be achieved by several techniques, which we survey in the next part of the introduction. Let us however first point out that the main contribution of this paper is a novel technique for precisely this part of the reduction: Using a method we call *parity separation*, we reduce the computation of $\text{PerfMatch}(G)$ for a ± 1-weighted graph G to the difference of PerfMatch for two unweighted graphs, that is, to the difference of two numbers of perfect matchings.

▶ **Lemma 1** (Parity Separation). *Let G be a graph on n vertices and m edges that is weighted by a function $w : E(G) \to \{-1, 1\}$. Then we can construct in time $\mathcal{O}(n + m)$ two unweighted graphs G_1 and G_2, each on $\mathcal{O}(n + m)$ vertices and edges, such that*

$$\text{PerfMatch}(G) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2). \tag{2}$$

Intuitively speaking, this allows us to "collect" positive and negative terms of $\text{PerfMatch}(G)$ for ± 1-weighted graphs. This way, we can reduce the effect of cancellations incurred *within* PerfMatch to a mere difference *outside* of PerfMatch.

In the remainder of this introduction, we present parity separation in more detail and demonstrate three applications that can be derived from it: Firstly, and not surprisingly, we obtain a new #P-completeness proof for counting perfect matchings. Secondly, we can show $\mathsf{C}_{=}\mathsf{P}$-completeness of deciding whether two graphs have the same number of perfect matchings. Thirdly, we also obtain tight lower bounds under the exponential-time hypothesis.

1.1 #P-completeness via parity separation

To put parity separation into context, we first recapitulate Valiant's #P-hardness result for counting perfect matchings in more detail. Let us denote the problem of evaluating PerfMatch on graphs with edge-weights from $A \subseteq \mathbb{Q}$ by PerfMatch^A. For consistency with [13], we write PerfMatch^{0,1} for the problem of counting perfect matchings in unweighted graphs. That is, we explicitly include $0 \in A$ for PerfMatch^A, although zero-weight edges could be simply deleted.

First step: From #SAT to PerfMatch$^{-1,0,1}$

It is shown in [27, Lemma 3.1] that PerfMatch^W is #P-hard for $W := \{-1, 0, 1, 2, 3\}$. More precisely, from a 3-CNF formula φ, a number $t(\varphi) \in \mathbb{N}$ and a bipartite graph $G = G(\varphi)$ on weights W are constructed in polynomial time, such that

$$\#\mathsf{SAT}(\varphi) = \frac{\mathrm{PerfMatch}(G)}{4^{t(\varphi)}}, \tag{3}$$

This however only yields hardness for a weighted generalization of counting perfect matchings. To obtain a useful reduction source for further problems, it is crucial to reduce PerfMatch^W to PerfMatch^{0,1}, as reductions from PerfMatch to other problems would otherwise need to take care of the weights in W, which is particularly problematic for the edge-weight -1 in the case of unweighted reduction targets.

In fact, the weight -1 is the only problem we encounter: Edges e of positive integer edge-weight w can be simulated easily by replacing e with w parallel edges of unit weight, possibly subdividing edges twice to obtain simple graphs. This trick however clearly does not apply for the weight -1, so we need a different strategy.

Second step: From PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1}

By now, two different strategies are known for removing the weight -1, which we briefly survey in the following. Let G be a graph with n vertices and $m > 0$ edges, all on weights -1 and 1.

- **Modular arithmetic**: Variations of the following approach were originally used by Valiant [27] and later by Zanko [32] and Ben-Dor and Halevi [1]: Write $M = 2^m + 1$ and observe that $\mathrm{PerfMatch}(G) < M$. We can hence replace the weight -1 by the positive integer $M - 1$ to obtain a graph G' satisfying $\mathrm{PerfMatch}(G) \equiv \mathrm{PerfMatch}(G')$ modulo M. The weight $M - 1$ can be simulated by a gadget as in the previous paragraph, and using a more involved construction [32], it can be seen that a gadget on $\mathcal{O}(m)$ vertices and edges suffices, yielding a total number of $\mathcal{O}(nm)$ vertices and $\mathcal{O}(m^2)$ edges in G'. Then we compute $\mathrm{PerfMatch}(G')$ modulo M and obtain $\mathrm{PerfMatch}(G)$, as we may assume from (3) that $\mathrm{PerfMatch}(G) \geq 0$. In total, we obtain one reduction image for PerfMatch^{0,1} on $\mathcal{O}(nm)$ vertices and $\mathcal{O}(m^2)$ edges.
- **Polynomial interpolation:** An alternative technique for removing the edge-weight -1 from G is to replace it by an indeterminate x. This gives rise to a graph G_x on edge-weights $\{1, x\}$ for which $\mathrm{PerfMatch}(G_x)$ is a polynomial $p(x) \in \mathbb{Z}[x]$ of degree at most $n/2$. We can evaluate $p(i)$ for $i \in \{0, \ldots, n/2\}$ by substituting $x \leftarrow i$ in G_x and simulating this positive weight by a gadget as discussed before. This allows us to recover $p(-1) = \mathrm{PerfMatch}(G)$ via Lagrangian interpolation. In total, using gadgets as in [32, 13], we obtain $\mathcal{O}(n)$ reduction images for PerfMatch^{0,1} on $\mathcal{O}(m \log n)$ vertices and $\mathcal{O}(m \log n + m)$ edges each.

Both weight removal techniques allow to reduce PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1} and thus complete the #P-completeness proof of the latter problem. Note however that both approaches map weighted graphs G with m edges to unweighted graphs with a super-linear number of edges. Using parity separation, we obtain a third way of performing the weight removal step, which differs substantially from both approaches mentioned before and features only constant blowup:

- **Parity separation:** Using Lemma 1, compute two unweighted graphs G_1 and G_2 from G such that PerfMatch(G) is the mere difference of PerfMatch(G_1) and PerfMatch(G_2). In total, we obtain 2 reduction images for PerfMatch^{0,1} on $\mathcal{O}(n+m)$ vertices and edges.

Together with the first step, this implies an alternative #P-completeness proof for the problem PerfMatch^{0,1} of counting perfect matchings in unweighted graphs. For the sake of completeness, we also include a self-contained reduction from #SAT to PerfMatch$^{-1,0,1}$.

▶ **Theorem 2.** PerfMatch^{0,1} *is* #P*-complete under polynomial-time Turing reductions.*

1.2 $\mathsf{C}_=\mathsf{P}$-completeness via parity separation

Apart from an alternative #P-completeness proof, Lemma 1 also yields implications for the structural complexity of PerfMatch: We show that deciding whether two unweighted graphs have the same number of perfect matchings is complete for the complexity class $\mathsf{C}_=\mathsf{P}$, which was introduced in [25, 31] and further elaborated in [16, 15].

To define $\mathsf{C}_=\mathsf{P}$, let us associate the following language $\mathsf{A}_=$ with each counting problem $\mathsf{A} \in \#\mathsf{P}$: The inputs to $\mathsf{A}_=$ are pairs (x, y) of instances to A, and we are asked to determine whether $\mathsf{A}(x) = \mathsf{A}(y)$ holds. We can then define[1] the class $\mathsf{C}_=\mathsf{P} := \{\mathsf{A}_= \mid \mathsf{A} \in \#\mathsf{P}\}$.

For instance, it is clear that $\#\mathsf{SAT}_=$, the problem that asks whether two 3-CNF formulas have the same number of satisfying assignments, is $\mathsf{C}_=\mathsf{P}$-complete under polynomial-time many-one reductions. In fact, $\mathsf{C}_=\mathsf{P}$-completeness holds for every problem $\mathsf{A}_=$ whose counting version #A is #P-complete under parsimonious reductions. We recall the notion of parsimonious (and other) reductions in Definition 5.

The relationship between $\mathsf{C}_=\mathsf{P}$ and other complexity classes has been studied in structural complexity theory, and several results are surveyed in [15]. For instance, we clearly have $\mathsf{coNP} \subseteq \mathsf{C}_=\mathsf{P}$, and using the witness isolation technique [30], we see that NP is contained in $\mathsf{C}_=\mathsf{P}$ under randomized reductions. Let us also observe that $\mathsf{NP}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{C}_=\mathsf{P}}$: Whenever we issue an oracle call to #P, we may instead guess the output number, and then check whether we guessed correctly by using the $\mathsf{C}_=\mathsf{P}$ oracle.

To the best of the author's knowledge, no *natural* $\mathsf{C}_=\mathsf{P}$-complete problem $\mathsf{A}_=$ is known whose counting version A is *not* #P-complete under parsimonious reductions.[2] It is clear that the problem PerfMatch^{0,1} of counting unweighted perfect matchings cannot be #P-complete under parsimonious reductions, unless $\mathsf{P} = \mathsf{NP}$. Therefore, the following completeness

[1] We deviate here from the standard definition of $\mathsf{C}_=\mathsf{P}$, according to which we have $L \in \mathsf{C}_=\mathsf{P}$ iff the following holds: There is a polynomial-time nondeterministic Turing machine M such that $x \in L$ iff the numbers of accepting and rejecting computation paths of $M(x)$ are equal. It can be verified easily that this is equivalent to our definition.

[2] Here, we stressed *natural*, because we can easily construct artificial $\mathsf{C}_=\mathsf{P}$-complete problems $\mathsf{A}_=$ whose counting version #A admits no parsimonious reduction from #SAT: Consider as an example the counting problem $\#\mathsf{SAT}'$ that asks to count satisfying assignments, incremented by 1. If $\#\mathsf{SAT}'$ had a parsimonious reduction from #SAT, then every CNF-formula would be satisfiable. On the other hand, the reduction from $\#\mathsf{SAT}_=$ to $\#\mathsf{SAT}'_=$ is trivial.

result for $\text{PerfMatch}^{0,1}_{=}$ seems relevant for structural complexity theory, as it establishes a $\mathsf{C}_{=}\mathsf{P}$-variant of Valiant's result.

▶ **Theorem 3.** $\text{PerfMatch}^{0,1}_{=}$ *(deciding whether two unweighted graphs have the same number of perfect matchings) is* $\mathsf{C}_{=}\mathsf{P}$*-complete under polynomial-time many-one reductions.*

To prove this theorem, we first reduce instances (φ, φ') for $\#\mathsf{SAT}_{=}$ to ± 1-weighted graphs G that satisfy $\text{PerfMatch}(G) = 0$ iff $\#\mathsf{SAT}(\varphi) = \#\mathsf{SAT}(\varphi')$. This requires a modification of the first step in the $\#\mathsf{P}$-hardness reduction, which is however supported easily by our alternative proof. Then we apply Lemma 1 on the graph G to obtain unweighted graphs G_1 and G_2 satisfying (2). In particular, their numbers of perfect matchings agree iff $\text{PerfMatch}(G)$ vanishes, that is, iff (φ, φ') is a yes-instance for $\#\mathsf{SAT}_{=}$.

To conclude this subsection, we note that the complexity of a similar problem was posed as an open question in [8]: Given two directed acyclic graphs, decide whether their numbers of topological orderings agree. It was shown in [5] that counting topological orderings is $\#\mathsf{P}$-complete under Turing reductions, but the decision version is trivial for DAGs. Our result for $\text{PerfMatch}^{0,1}_{=}$ might be useful to prove $\mathsf{C}_{=}\mathsf{P}$-completeness for this and other problems. For instance, a reduction was recently found [24] from $\text{PerfMatch}^{0,1}_{=}$ to deciding whether two formulas in 2-CNF are satisfied by the same number of assignments.

1.3 Tight lower bounds via parity separation

We turn our attention to conditional *quantitative* lower bounds: A relatively new subfield in computational complexity makes use of assumptions stronger than $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{FP} \neq \#\mathsf{P}$ to prove tight (exponential) lower bounds on the running times needed to solve computational problems. A popular such assumption is the exponential-time hypothesis ETH, introduced by Impagliazzo et al. [19, 20], which states that the satisfiability of n-variable formulas φ in 3-CNF cannot be decided in time $2^{o(n)}$. For counting problems, an analogous variant $\#\mathsf{ETH}$ was introduced by Dell et al. [13], and it postulates the same for the problem of *counting* satisfying assignments to φ.

Assuming ETH, it was shown for a vast body of popular decision problems that the known exponential-time exact algorithms are somewhat optimal: For instance, there is a trivial $2^{\mathcal{O}(m)}$ time algorithm for finding a Hamiltonian cycle (or various other structures) in an m-edge graph, but $2^{o(m)}$ time algorithms would refute ETH. See [23] for a nice survey.

Similar lower bounds were shown for counting problems under $\#\mathsf{ETH}$, see [17, 18, 13], and a very recent paper [9] introduced *block interpolation*, an approach to make the technique of polynomial interpolation (as seen in the second step of Section 1.1) compatible with tight lower bounds under $\#\mathsf{ETH}$. For several problems, that of counting perfect matchings being among them, block interpolation gave the first tight $2^{\Omega(m)}$ lower bounds under $\#\mathsf{ETH}$.

When applying this framework to $\text{PerfMatch}^{0,1}$, we would first reduce $\#\mathsf{SAT}$ on n-variable 3-CNFs φ to instances $G = G(\varphi)$ for $\text{PerfMatch}^{-1,0,1}$ with $\mathcal{O}(n)$ edges as in the first step of the $\#\mathsf{P}$-hardness proof. Then we apply the block interpolation technique to reduce G to $2^{o(n)}$ unweighted instances G' for $\text{PerfMatch}^{0,1}$ with $\mathcal{O}(n)$ edges. While this sub-exponential number of instances is compatible with the goal of proving tight lower bounds, it leaves open the natural question whether the same reduction could be achieved with only polynomially many oracle calls on graphs with $\mathcal{O}(n)$ edges.

Using Lemma 1, we obtain a strong positive answer to this question: Replacing the application of block interpolation by one of parity separation, we obtain a reduction to merely *two* instances of $\text{PerfMatch}^{0,1}$. And as a synthesis of structural and quantitative complexity, we also obtain a tight lower bound for the equality-testing problem $\text{PerfMatch}^{0,1}_{=}$.

▶ **Theorem 4.** *Unless* #ETH *fails, the problem* $\text{PerfMatch}^{0,1}$ *admits no algorithm with running time* $2^{o(m)}$ *on simple graphs with* m *edges. Furthermore, the same applies to* $\text{PerfMatch}^{0,1}_{=}$ *under the decision version* ETH*.*

Organization of this paper

The remainder of this paper is structured as follows: In Section 2, we introduce the Holant framework and matchgates, concepts that are crucial to our constructions. These are put to use in Section 3, where we prove Lemma 1, our main result. Its applications, as discussed above, are shown in Section 4.

2 Preliminaries

Graphs in this paper may be edge- or vertex-weighted. Given a graph G and $v \in V(G)$, denote the edges incident with v by $I(v)$. If the context of an argument unambiguously determines a graph G, we write $n = |V(G)|$ and $m = |E(G)|$.

We denote the Hamming weight of strings $x \in \{0,1\}^*$ by $\text{hw}(x)$. Given a statement φ, we let $[\varphi] = 1$ if φ is true, and $[\varphi] = 0$ otherwise. For convenience, we recall that several reduction notions are distinguished in the study of counting complexity: The most restrictive notion is that of *parsimonious* (many-one) reductions, which can be slightly relaxed to *weakly* parsimonious reductions. The most permissive notion is that of *Turing* reductions.

▶ **Definition 5.** Let A and B be counting problems. Let $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \to \mathbb{Q}$ be polynomial-time computable functions. If $\mathsf{A}(x) = g(x) \cdot \mathsf{B}(f(x))$ holds for all $x \in \{0,1\}^*$, then we call (f, g) a *weakly parsimonious (polynomial-time) reduction* from A to B and write $\mathsf{A} \leq_p \mathsf{B}$. If additionally $g(x) = 1$ holds for all $x \in \{0,1\}^*$, then we call f *parsimonious* and write $\mathsf{A} \leq_p^{pars} \mathsf{B}$.

If $\mathbb{T}$ is a deterministic polynomial-time algorithm that solves A with an oracle for B, then we call $\mathbb{T}$ a *Turing reduction* from A to B and write $\mathsf{A} \leq_p^T \mathsf{B}$.

2.1 Weighted sums of (perfect) matchings

The quantity PerfMatch on edge-weighted graphs, as defined in (1) and [29], will be the central object of investigation in this paper. For intermediate steps, we also consider the quantity MatchSum introduced in [29].

▶ **Definition 6.** For vertex-weighted graphs G with $w : V(G) \to \mathbb{Q}$, let $\mathcal{M}[G]$ denote the set of (not necessarily perfect) matchings in G. Recall that $\mathcal{PM}[G] \subseteq \mathcal{M}[G]$ denotes the perfect matchings in G. For $M \in \mathcal{M}[G]$, let $\text{usat}(M)$ denote the set of unmatched vertices in M. Then we define

$$\text{MatchSum}(G) = \sum_{M \in \mathcal{M}[G]} \prod_{v \in \text{usat}(M)} w(v).$$

Given $W \subseteq \mathbb{Q}$, we write PerfMatch^W for the problem of evaluating $\text{PerfMatch}(G)$ on graphs G with weights $w : E(G) \to W$. Likewise, write MatchSum^W on graphs with weights $w : V(G) \to W$. Please note that an edge of weight 0 in PerfMatch can be treated as if it were not present, whereas weight 0 at a vertex v in MatchSum signifies that v must be matched. We can easily reduce PerfMatch^W for finite $W \subseteq \mathbb{Q}$ to $\text{PerfMatch}^{-1,0,1}$:

▶ **Lemma 7** (folklore). *Let G be edge-weighted by $w : E(G) \to \mathbb{Q}$. Let $q \in \mathbb{N}$ denote the least common denominator of the weights in G, and let $T = \max_{e \in E(G)} q \cdot w(e)$. Then we can compute a number $B \in \mathbb{N}$ and an edge-weighted graph G' on $\mathcal{O}(n + Tm)$ vertices and edges, all of weight ± 1, such that* $\text{PerfMatch}(G) = q^{-B} \cdot \text{PerfMatch}(G')$.

2.2 Holant problems

We give an introduction to the *Holant framework*, summarizing ideas from [29, 6, 7]. A more detailed introduction to our notation can be found in [11].

▶ **Definition 8** (adapted from [29]). A *signature graph* is an edge-weighted graph Ω, which may feature parallel edges, with a *vertex function* $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ at each $v \in V(\Omega)$.

The *Holant* of Ω is a particular sum over edge assignments $x \in \{0,1\}^{E(\Omega)}$. We sometimes identify x with the set $x^{-1}(1)$ of indices that have value 1 under x. Given $S \subseteq E(\Omega)$, we write $x|_S$ for the restriction of x to S, which is the unique assignment in $\{0,1\}^S$ that agrees with x on S. Then we define

$$\text{Holant}(\Omega) := \sum_{x \in \{0,1\}^{E(\Omega)}} \left(\prod_{e \in x} w(e) \right) \left(\prod_{v \in V(\Omega)} f_v(x|_{I(v)}) \right). \tag{4}$$

As a first example, we can reformulate $\text{PerfMatch}(G)$ easily as the Holant of a signature graph $\Omega = \Omega(G)$ by declaring $f_v : \{0,1\}^{I(v)} \to \{0,1\}$ for $v \in V(G)$ to be the vertex function that maps $x \in \{0,1\}^*$ to 1 iff $\text{hw}(x) = 1$ and to 0 else.

When considering signature graphs Ω in the following, we will always assume that $I(v)$ for each $v \in V(\Omega)$ is ordered in a fixed (usually implicit) way. This way, if v is a vertex of degree $d \in \mathbb{N}$, we can simply view f_v as a function $f_v : \{0,1\}^d \to \mathbb{Q}$, and we call this representation a *signature*.

▶ **Example 9.** The following are signatures of arity $k \in \mathbb{N}$ on inputs $x \in \{0,1\}^{[k]}$ with $x = (x_1, \ldots, x_k)$.

$$\begin{aligned}
\texttt{EQ} &: x \mapsto [x_1 = \ldots = x_k] \\
\texttt{HW}_{=1} &: x \mapsto [\text{hw}(x) = 1] \\
\texttt{HW}_{\leq 1} &: x \mapsto [\text{hw}(x) \leq 1] \\
\texttt{ODD} &: x \mapsto x_1 \oplus \ldots \oplus x_k \\
\texttt{EVEN} &: x \mapsto 1 \oplus x_1 \oplus \ldots \oplus x_k.
\end{aligned}$$

We may write, say, $\texttt{EQ}_4$ to denote the arity-4 signature $\texttt{EQ}$. Note that these signatures are symmetric, as they depend only upon the Hamming weight on the input.

Similarly as for PerfMatch, we can also express MatchSum as a Holant problem.

▶ **Lemma 10.** *Let G be a graph with vertex-weights $w : V(G) \to \mathbb{Q}$. Then* $\text{MatchSum}(G) = \text{Holant}(\Omega)$ *holds with the signature graph Ω that is derived from G by placing $\texttt{VTX}_{w(v)}$ at $v \in V(G)$ and assigning weight 1 to all edges. Here, $\texttt{VTX}_w$ for $w \in \mathbb{Q}$ is defined as*

$$\texttt{VTX}_w : \quad x \mapsto \begin{cases} w & \text{if } \text{hw}(x) = 0, \\ 1 & \text{if } \text{hw}(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

We can easily reduce edge-weighted Holant problems to unweighted versions as follows.

▶ **Lemma 11.** *Let Ω' be defined as follows from Ω: Subdivide each edge $e \in E(\Omega)$ into two edges, assign weight 1 to the obtained subdivision edges, and equip the obtained subdivision vertices with the signature $\texttt{EDGE}_{w(e)}$, where*

$$\texttt{EDGE}_w : \quad x \mapsto \begin{cases} w & \text{if } x = 11, \\ 0 & \text{if } x \in \{01, 10\}, \\ 1 & \text{if } x = 00. \end{cases}$$

Then Ω' features only the edge-weight 1, and we have $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega')$.

Finally, a signature is called *even* if its support contains only bitstrings of even Hamming weight. The problem #SAT can be rephrased as a Holant problem with even signatures:

▶ **Lemma 12.** *For $n, m, d \in \mathbb{N}$, let φ be a d-CNF formula on variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$. We construct a signature graph Ω as follows:*

- *For each $i \in [n]$, let $r(i)$ denote the number of occurrences of x_i (as a positive or negative literal) in φ. Create a* variable vertex *v_i in Ω, with signature $\texttt{EQ}_{2r(i)}$.*
- *For each $j \in [m]$, let $x_{i_1}, \ldots, x_{i_d}$ be the variables that clause c_j depends upon. We create a* clause vertex *w_j in Ω, and for $\kappa \in [d]$, we add two parallel edges of weight 1 between w_j and v_{i_κ} as the $2\kappa - 1$-th and 2κ-th edges in the ordering of $I(w_j)$.*
- *For each $j \in [m]$, consider clause c_j as a Boolean function on variables $z_1, \ldots, z_d$, where z_i for $i \in [d]$ represents the i-th variable in c_j. Define a function c'_j on variables $y_1, \ldots, y_{2d}$ that outputs $c_j(y_1, y_3, \ldots, y_{2d-1})$ if $y_{2i} = y_{2i-1}$ for all $i \in [d]$. On all other inputs, the value of c'_j is defined to be zero. Assign such a signature c'_j to the vertex w_j.*

Then we have $\#\mathsf{SAT}(\varphi) = \mathrm{Holant}(\Omega)$. Note that Ω is a signature graph with $n + m$ vertices and $2dm$ edges that features only even signatures and the edge-weight 1.

▶ Remark. The degree of clause vertices above is $2d$ rather than d to ensure that their signatures are even. The need for this will become clear in the next subsection.

2.3 Gates and matchgates

Given a signature graph Ω, we can sometimes simulate vertex functions by gadgets or *gates*, which are signature graphs with so-called *dangling edges* that feature only one endpoint. These notions are borrowed from the $\mathcal{F}$-gates in [7]. Matchgates were first considered in [29].

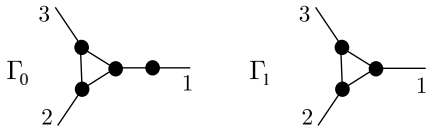
▶ **Definition 13** (adapted from [7])**.** For disjoint sets A, B, and for assignments $x \in \{0,1\}^A$ and $y \in \{0,1\}^B$, we write $xy \in \{0,1\}^{A \cup B}$ for the assignment that agrees with x on A, and with y on B. We also say that the assignment xy *extends* x. A *gate* is a signature graph Γ containing a set $D \subseteq E(\Gamma)$ of dangling edges, all of which have edge-weight 1. The *signature realized by* Γ is the function $\mathrm{Sig}(\Gamma) : \{0,1\}^D \to \mathbb{Q}$ that maps x to

$$\mathrm{Sig}(\Gamma, x) = \sum_{y \in \{0,1\}^{E(\Gamma) \setminus D}} \left(\prod_{e \in xy} w(e) \right) \left(\prod_{v \in V(\Gamma)} f_v(xy|_{I(v)}) \right). \tag{5}$$

A gate Γ is a *matchgate* if it features only the signature $\texttt{HW}_{=1}$.

In the following, we consider the dangling edges D of gates Γ to be labelled as $1, \ldots, |D|$. This way, we can view $\mathrm{Sig}(\Gamma)$ as a function of type $\{0,1\}^{|D|} \to \mathbb{Q}$ instead of $\{0,1\}^D \to \mathbb{Q}$. We will use gates to realize required signatures as "gadgets" consisting of other (usually simpler) signatures. Consider the following example, which appeared in [29].

▶ **Example 14.** It can be verified that $\texttt{EVEN}_3$ and $\texttt{ODD}_3$ are realized by the matchgates Γ_0 and Γ_1 below, where all vertices are assigned $\texttt{HW}_{=1}$ and all edges have weight 1.



Using this, we can realize the signatures $\texttt{ODD}_k$ and $\texttt{EVEN}_k$ for any arity $k \geq 3$ as matchgates, noted also in [29, Theorem 3.3]. This will be required in Section 3.

▶ **Example 15.** For all $k \geq 3$, there exists a gate $\Gamma_{\texttt{EVEN}}$ with $\text{Sig}(\Gamma_{\texttt{EVEN}}) = \texttt{EVEN}_k$. It consists of vertices $v_1, \ldots, v_{k-2}$ equipped with $\texttt{EVEN}_3$, edges $e_1, \ldots, e_{k-3}$ of weight 1, and dangling edges $[k]$.

We can likewise realize $\texttt{ODD}_k$ by a gate $\Gamma_{\texttt{ODD}}$ that is constructed as above, but with $\texttt{ODD}_3$ rather than $\texttt{EVEN}_3$ at v_{k-2}.

In the following, we formalize the operation of *inserting* a gate Γ into a signature graph so as to simulate a desired signature. A more detailed version of this operation can be found in Definition 2.10 and Lemma 2.11 of [11].

▶ **Lemma 16.** *Let Ω be a signature graph, let $v \in V(\Omega)$ with $D = I(v)$ and let Γ be a gate with dangling edges D. We can* insert Γ at v *by deleting v and keeping D as dangling edges, and then placing Γ into Ω and identifying each dangling edge $e \in D$ across Γ and Ω. If Ω' is derived from Ω by inserting a gate Γ with $\text{Sig}(\Gamma) = f_v$ at v, then $\text{Holant}(\Omega) = \text{Holant}(\Omega')$.*

By an argument presented in the author's PhD thesis [11], also used in [12], we can realize every even signature f by some matchgate $\Gamma = \Gamma(f)$. If the image of f is W, then Γ contains $W \cup \{\pm 1, 1/2\}$ as edge-weights. For sake of completeness, we include a self-contained proof in the full version.

▶ **Lemma 17** ([11, 12])**.** *Let Ω be a signature graph on n vertices and m edges, with even vertex functions $\{f_v\}_{v \in V(\Omega)}$ that map into $W \subseteq \mathbb{Q}$. Let $s = \max_{v \in V(\Omega)} |\text{supp}(f_v)|$. Then we can construct, in linear time, a graph G on $\mathcal{O}(n + sm)$ vertices and edges such that $\text{Holant}(\Omega) = \text{PerfMatch}(G)$. The edge-weights of G are $W \cup \{\pm 1, 1/2\}$.*

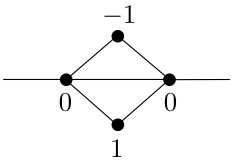
3 The parity separation technique

We are ready to prove Lemma 1, our main result. The proof proceeds by establishing, with several intermediate steps, the reduction chain

$$\text{PerfMatch}^{-1,0,1} \leq_p \text{MatchSum}^{-1,0,1} \leq_p^T \text{PerfMatch}^{0,1}. \tag{6}$$

For the first reduction in (6), we apply a gadget Γ realizing the signature $\texttt{EDGE}_{-1}$ from Lemma 11 to all edges of weight -1.

▶ **Lemma 18.** *We have $\texttt{EDGE}_{-1} = \text{Sig}(\Gamma)$, where Γ is the gate below. In Γ, each vertex features the signature $\texttt{VTX}_w$ for the number $w \in \{-1, 0, 1\}$ it is annotated with in the drawing.*



This allows us to transform an instance for PerfMatch$^{-1,0,1}$ to one for MatchSum$^{-1,0,1}$.

▶ **Lemma 19.** *Let G be a graph with n vertices and m edges, all of weight ± 1. Then we can compute a graph G' on $\mathcal{O}(n+m)$ edges, with vertices of weight $\{-1,0,1\}$, such that* PerfMatch(G) = MatchSum(G').

Proof. We assume that $|V(G)|$ is even, as otherwise PerfMatch$(G) = 0$. First, let Ω be the signature graph constructed by assigning $\texttt{HW}_{=1}$ to all vertices of G, and then applying the signature $\texttt{EDGE}_{-1}$ as in Lemma 11. We obtain PerfMatch(G) = Holant(Ω).

Then realize each occurrence of $\texttt{EDGE}_{-1}$ by the gate Γ from Lemma 18. Note that Γ features no edge-weights, and only $\texttt{VTX}_w$ for $w \in \{-1,0,1\}$. We obtain a signature graph Ω' whose signatures are all of the type $\texttt{VTX}_w$ for $w \in \{-1,0,1\}$, and which satisfies Holant(Ω) = Holant(Ω'). Note that $\texttt{HW}_{=1} = \texttt{VTX}_0$, so this indeed covers all vertices of Ω'.

By Lemma 10, we may equivalently consider Holant(Ω') = MatchSum(G'), where G' is a vertex-weighted graph obtained from Ω' as follows: Keep all vertices and edges of Ω' intact, and if $v \in V(\Omega')$ features the signature $\texttt{VTX}_w$, for $w \in \{-1,0,1\}$, then assign the vertex weight w to v in G'. ◀

For the second reduction in (6), we perform the actual act of parity separation: We will split the vertex-weighted graph G' into an even part G_0 and an odd part G_1, both unweighted, such that the *perfect* matchings of the even (resp. odd) part correspond bijectively to the matchings of G' with an even (resp. odd) number of unmatched vertices of weight -1.[3] Since $(-1)^{even} = 1$ and $(-1)^{odd} = -1$, this clearly implies that MatchSum(G) is the difference of PerfMatch(G_0) and PerfMatch(G_1).

To proceed, we first use the signatures EVEN and ODD from Example 9 to obtain an alternative reformulation of MatchSum$^{-1,0,1}$ as the difference of two Holants.

▶ **Lemma 20.** *Let G' be a graph with vertex-weights $\{-1,0,1\}$. For $a,b \in \{0,1\}$, let $\Phi_{ab} = \Phi_{ab}(G')$ be the signature graph obtained as follows:*

1. *Assign the signature $\texttt{HW}_{=1}$ to all vertices of G'.*
2. *For $x \in \{-1,0,1\}$, let $V_x \subseteq V(G')$ denote the set of vertices of weight x in G'. For $x \in \{-1,1\}$, add a vertex u_x connected to V_x.*
   - *Assign to u_{-1} the signature EVEN if $a = 0$, and assign ODD if $a = 1$.*
   - *Assign to u_1 the signature EVEN if $b = 0$, and assign ODD if $b = 1$.*

   Then we have MatchSum(G') = Holant(Φ_{00}) − Holant(Φ_{11}).

The second reduction in (6) follows by realizing the signatures ODD and EVEN appearing in Φ_{00} and Φ_{11} via matchgates that feature neither edge- nor vertex-weights. Note that the only other appearing signature $\texttt{HW}_{=1}$ is trivially realized by such a matchgate.

Proof of Lemma 1. Follows from Lemma 19 (to reduce PerfMatch$^{-1,0,1}$ to MatchSum$^{-1,0,1}$) with Lemma 20 (to reformulate MatchSum$^{-1,0,1}$ as a Holant problem) and Example 15 (to realize the ODD and EVEN signatures in the Holant problem by unweighted matchgates). ◀

[3] This step is inspired by a reduction [29, Theorem 3.3] from certain instances of MatchSum on planar graphs to PerfMatch on planar graphs.

4 Parity separation in action

In the final section of this paper, we cover the three applications of parity separation that we discussed in the introduction.

4.1 Completeness for #P

We can easily show the #P-completeness of $\text{PerfMatch}^{0,1}$ via parity separation. To this end, we first express #SAT as a Holant problem on even signature graphs, as seen in Lemma 12. Together with Lemma 17, this yields $\#\mathsf{SAT} \leq_p \text{PerfMatch}^B$ with $B = \{-1, 0, 1/2, 1\}$. We use Lemma 7 to remove the edge-weight $1/2$, and finally remove the weight -1 by parity separation as in Lemma 1. Altogether, we obtain the following lemma.

▶ **Lemma 21.** *Let φ be a 3-CNF formula with n variables and m clauses. Then we can compute a number $T \in \mathbb{N}$ and construct two unweighted graphs G_1 and G_2 on $\mathcal{O}(n+m)$ vertices and edges, all in time $\mathcal{O}(n+m)$, such that $2^T \cdot \#\mathsf{SAT}(\varphi) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2)$.*

This readily implies Theorem 2, the desired #P-completeness result.

4.2 Completeness for $\mathsf{C}_=\mathsf{P}$

For our next application, we apply the parity separation technique to prove Theorem 3. That is, we prove $\mathsf{C}_=\mathsf{P}$-completeness of the problem $\text{PerfMatch}^{0,1}_=$ that asks, given two unweighted graphs G_1 and G_2, whether their numbers of perfect matchings agree. We call graphs satisfying this property *equipollent graphs* and will likewise speak of *equipollent formulas* if their numbers of satisfying assignments agree.

Proof of Theorem 3. The problem $\text{PerfMatch}^{0,1}_=$ is clearly contained in $\mathsf{C}_=\mathsf{P}$. For the hardness part, we reduce from the $\mathsf{C}_=\mathsf{P}$-complete problem $\#\mathsf{SAT}_=$ that asks, given 3-CNF formulas φ and φ', to determine whether they are equipollent. To this end, we construct unweighted graphs G and G' that are equipollent if and only if φ and φ' are.

Assume that φ and φ' are defined on the same set of variables $x_1, \ldots, x_n$ and feature the same number m of clauses. This can be achieved by renaming variables, and by adding dummy variables and clauses. If, say, φ has less variables than φ', then we can add dummy variables to φ', together with clauses that ensure that every dummy variable has the same assignment as x_1. We can also duplicate clauses.

Let $C_1, \ldots, C_m$ and $C'_1, \ldots, C'_m$ denote the clauses in φ and φ', respectively. We introduce a *selector* variable x^* and define a formula ψ on the variable set $\mathcal{X} = \{x^*, x_1, \ldots, x_n\}$, which has clauses $D_1, \ldots, D_m$ and $D'_1, \ldots, D'_m$, where $D_i := (x^* \vee C_i)$ and $D'_i := (\neg x^* \vee C'_i)$ for $i \in [m]$. If $a(x^*) = 0$ holds in an assignment $a \in \{0,1\}^{\mathcal{X}}$, then all clauses $D'_1, \ldots, D'_m$ are satisfied by $\neg x^*$, but in order for a to satisfy ψ, the clauses $D_1, \ldots, D_m$ have to be satisfied by $x_1, \ldots, x_n$. In other words, if a satisfies ψ and $a(x^*) = 0$, then the restriction of a to $x_1, \ldots, x_n$ satisfies φ. Likewise, if a satisfies ψ and $a(x^*) = 1$, then the restriction of a to $x_1, \ldots, x_n$ satisfies φ'. Hence, we can define the following quantity

$$S := \sum_{a \in \{0,1\}^{\mathcal{X}}} (-1)^{a(x^*)} \cdot [\psi \text{ satisfied by } a]$$

and we observe that $S = \#\mathsf{SAT}(\varphi) - \#\mathsf{SAT}(\varphi')$. It is clear that $S = 0$ if and only if φ and φ' are equipollent. As in Lemma 12, we then express $S = \text{Holant}(\Omega)$ for a signature graph

$\Omega = \Omega(\psi)$, with one modification: At the vertex v^* corresponding to the variable x^*, we do not use the signature $\texttt{EQ}$, but rather a modified signature

$$\texttt{EQ}_{-} : \quad y \mapsto \begin{cases} -1 & \text{if } y = 1 \dots 1, \\ 1 & \text{if } y = 0 \dots 0, \\ 0 & \text{otherwise.} \end{cases}$$

We realize Ω via Lemma 17 to obtain a graph G on edge-weights $1/2, \pm 1$, simulate the edge-weight $1/2$ via Lemma 7, and obtain an edge-weighted graph H with weights ± 1 together with a number $T \in \mathbb{N}$ such that

$$S = \text{Holant}(\Omega) = 2^{-T} \cdot \text{PerfMatch}(H). \tag{7}$$

Using Lemma 1, we then obtain unweighted graphs G and G' such that

$$\text{PerfMatch}(H) = \text{PerfMatch}(G) - \text{PerfMatch}(G'). \tag{8}$$

Then G and G' are equipollent iff $S = 0$, which in turn holds iff φ and φ' are equipollent. ◀

4.3 Tight lower bounds under #ETH

By the exponential-time hypothesis #ETH, there is no $2^{o(n)}$ time algorithm for counting satisfying assignments to 3-CNF formulas φ with n variables. Applying the counting version of the so-called sparsification lemma, shown in [13], we may additionally assume that φ features $m = \mathcal{O}(n)$ clauses. Then Lemma 21 clearly implies the lower bound for $\text{PerfMatch}^{0,1}$ claimed in Theorem 4.

Concerning $\text{PerfMatch}^{0,1}_{=}$, it is even easier to prove lower bounds under ETH than to prove its $\mathsf{C}_{=}\mathsf{P}$-completeness, as we may (i) reduce from SAT rather than $\mathsf{SAT}_{=}$, and (ii) use the more permissive notion of Turing (rather than many-one) reductions: With Lemma 21, we can construct unweighted graphs G_1 and G_2 on $\mathcal{O}(m)$ vertices and edges that are equipollent iff φ is *unsatisfiable*, thus a $2^{o(m)}$ time algorithm would contradict ETH. This proves Theorem 4.

5 Conclusion and future work

We have added a new method to the known techniques (modular arithmetic and polynomial interpolation) for removing the edge-weight -1 from $\text{PerfMatch}^{-1,0,1}$. This method is based on matchgates and the simple observation that $(-1)^{even} = 1$ and $(-1)^{odd} = -1$. We obtained non-trivial applications that could not be obtained via the previously known techniques.

Our work leaves some interesting questions open for further investigations. For instance, we could not find a way to show #P-completeness of $\text{PerfMatch}^{0,1}$ on *bipartite* graphs by parity separation. Is there a complexity-theoretic explanation for this? On another note, can we prove $\mathsf{C}_{=}\mathsf{P}$-completeness for other "equality-testing" versions of counting problems?

Acknowledgments. The author wishes to thank Markus Bläser, Mingji Xia, Meena Mahajan and Jin-Yi Cai for interesting discussions. Furthermore, thanks to Patrick Scharpfenecker for pointing out [24].

References

1 Amir Ben-Dor and Shai Halevi. Zero-one permanent is *#P*-complete, A simpler proof. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993. Proceedings*, pages 108–117, 1993.

2 Markus Bläser and Radu Curticapean. The complexity of the cover polynomials for planar graphs of bounded degree. In *Mathematical Foundations of Computer Science 2011 – 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, pages 96–107, 2011. doi:10.1007/978-3-642-22993-0_12.

3 Markus Bläser and Radu Curticapean. Weighted counting of k-matchings is #W[1]-hard. In *Parameterized and Exact Computation – 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings*, pages 171–181, 2012. doi:10.1007/978-3-642-33293-7_17.

4 Markus Bläser and Holger Dell. Complexity of the cover polynomial. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 801–812, 2007. doi:10.1007/978-3-540-73420-8_69.

5 Graham Brightwell and Peter Winkler. Counting linear extensions is #P-complete. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 175–181, 1991. doi:10.1145/103418.103441.

6 Jin-Yi Cai and Pinyan Lu. Holographic algorithms: From art to science. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 401–410, New York, NY, USA, 2007. ACM. doi:http://doi.acm.org/10.1145/1250790.1250850.

7 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 644–653. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.34.

8 Mike Chen. The complexity of checking whether two DAG have the same number of topological sorts, November 2010. URL: http://cstheory.stackexchange.com/questions/3105.

9 Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 380–392, 2015. doi:10.1007/978-3-662-47672-7_31.

10 Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. *CoRR*, abs/1511.07480, 2015. URL: http://arxiv.org/abs/1511.07480.

11 Radu Curticapean. *The simple, little and slow things count: on parameterized counting complexity*. PhD thesis, Saarland University, August 2015.

12 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669, 2016. doi:10.1137/1.9781611974331.ch113.

13 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms*, 10(4):21, 2014. doi:10.1145/2635812.

14 Jack Edmonds. Paths, trees, and flowers. In *Classic Papers in Combinatorics*, Modern Birkhauser Classics, pages 361–379. Birkhauser Boston, 1987. doi:10.1007/978-0-8176-4842-8_26.

15 Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Springer, 2002.

16 Lane A. Hemaspaandra and Heribert Vollmer. The satanic notations: counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, 1995. doi:10.1145/203610.203611.

17 Christian Hoffmann. Exponential time complexity of weighted counting of independent sets. In *Parameterized and Exact Computation – 5th International Symposium, IPEC*

2010, Chennai, India, December 13-15, 2010. Proceedings, pages 180–191, 2010. doi:10.1007/978-3-642-17493-3_18.

18 Thore Husfeldt and Nina Taslaman. The exponential time complexity of computing the probability that a graph is connected. In *Parameterized and Exact Computation – 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, pages 192–203, 2010. doi:10.1007/978-3-642-17493-3_19.

19 Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

20 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

21 Pieter W. Kasteleyn. The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961. doi:10.1016/0031-8914(61)90063-5.

22 Pieter W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

23 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

24 Patrick Scharpfenecker and Jacobo Torán. Solution-graphs of boolean formulas and isomorphism. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:24, 2016. URL: http://eccc.hpi-web.de/report/2016/024.

25 J. Simon. *On Some Central Problems in Computational Complexity.* PhD thesis, Cornell University, 1975.

26 H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics – an exact result. *Philosophical Magazine*, 6(68):1478–6435, 1961.

27 Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

28 Leslie G. Valiant. Accidental algorithms. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 509–517, 2006. doi:10.1109/FOCS.2006.7.

29 Leslie G. Valiant. Holographic algorithms. *SIAM Journal on Computing*, 37(5):1565–1594, 2008. doi:10.1137/070682575.

30 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

31 Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986. doi:10.1007/BF00289117.

32 Viktoria Zanko. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science*, 2(1):77–82, 1991. doi:10.1142/S0129054191000066.