# Completeness for Parallel Access to NP and Counting Class Separations

I n a u g u r a l - D i s s e r t a t i o n
zur
Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von
Holger Spakowski
aus Teterow

Düsseldorf
2005

Teile dieser Arbeit wurden bereits in den folgenden Schriften veröffentlicht bzw. zur Publikation angenommen: [SV00], [RSV02, RSV03], [HSV], [HRS02, HRS], [STT03, STT], [ST04].

Gedruckt mit der Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine-Universität Düsseldorf

Referent: Prof. Dr. Jörg Rothe

Korreferenten: Prof. Dr. Klaus W. Wagner, Prof. Dr. Egon Wanke

Tag der mündlichen Prüfung: 14.07.2005

# Acknowledgments

I am deeply grateful for the guidance, support, and encouragement of my thesis advisor Professor Jörg Rothe. The current thesis greatly benefited from countless inspiring discussions I had with him. Particularly, I am much indebted to him for facilitating my one-year visit to the University of Rochester. I am very grateful to Professor Lane Hemaspaandra for inviting me to the Department of Computer Science at the University of Rochester.

I would like to thank my research collaborators and coauthors Edith Hemaspaandra, Jörg Rothe, Mayur Thakur, Rahul Tripathi, and Jörg Vogel.

The work described in Chapter 3 is joint with Jörg Vogel [SV00]. The work described in Chapter 4 is joint with Jörg Rothe and Jörg Vogel [RSV02, RSV03]. The work described in Chapter 5 is joint with Edith Hemaspaandra and Jörg Vogel [HSV]. The work described in Chapter 6 is joint with Edith Hemaspaandra and Jörg Rothe [HRS02, HRS]. The work described in Chapter 7 is joint with Mayur Thakur and Rahul Tripathi [STT03, STT], [ST04].

The Deutscher Akademischer Austauschdienst (DAAD) supported with a fellowship my research at the University of Rochester, and the DFG supported my research under grant RO 1202/9-1.

I thank Professor Klaus Wagner and Professor Egon Wanke for serving as co-reviewers for this thesis.

# Contents

# Chapter 1

# Introduction

One of the major goals of computational complexity theory is to understand the amount of resources (time, space) needed to solve computational problems that we care about. In particular, we would like to be able to distinguish the problems that are solvable efficiently from the ones that are intractable. A computational problem is usually regarded to be efficiently solvable if it can be decided by a deterministic Turing machine with polynomially many steps in the size of the problem instances. The class of such problems is denoted by P.

There is another fundamental class of problems: The class NP consisting of those problems that can be accepted by *nondeterministic* polynomial-time bounded Turing machines. Equivalently, NP is the class of problems for which all "yes"-instances can be verified in polynomial time. Consider for example the decision problem `Vertex Cover`. Given a graph $G$, a *vertex cover* of $G$ is a subset $V'$ of the set of vertices in $G$ such that at least one vertex of each edge of $G$ is in $V'$. Denote by $\tau(G)$ the number of vertices in a minimum vertex cover of $G$. Figure 1.1 shows a graph $G$ with a minimum vertex cover of size 3. The problem

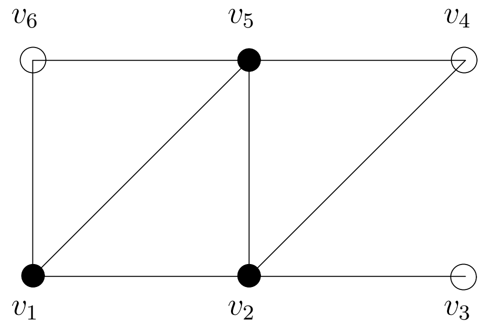

Figure 1.1: Graph $G$ with minimum vertex cover $\{v_1, v_2, v_5\}$ of size 3

`Vertex Cover` is defined as follows: Given a graph $G$ and an integer $k$ (a pair $\langle G, k\rangle$), is it true that $\tau(G) \leq k$? This problem has the following crucial property:

If the answer is "yes" then there exists a proof[1] for the correctness of the "yes"-answer that can be verified in polynomial time. Such a proof consists of a subset $V'$ of the vertex set of $G$ of size $\leq k$ forming a vertex cover of $G$. The correctness of the proof can be verified easily as follows. First, check that $V'$ contains no more than $k$ vertices. Second, check that each edge in $G$ has at least one of its end vertices contained in $V'$. This can clearly be done efficiently (in polynomial time). We have thus proved that `Vertex Cover` is in NP. All problems in NP share this property, i.e, they have efficiently verifiable membership proofs.

It is clear that P $\subseteq$ NP, but it is not known whether this inclusion is proper, i.e., whether there is any NP problem that provably cannot be solved by a deterministic polynomial-time algorithm. This is one of the most outstanding open questions of theoretical computer science and mathematics. The importance of the P $\overset{?}{=}$ NP question stems largely from the fact that there are thousands of practically and theoretically significant problems (e.g. the Traveling Salesperson Problem, various scheduling problems, the satisfiability problem for boolean formulas, graph coloring problems) that can easily be shown to be in NP, but have resisted all attempts to devise efficient (deterministic) algorithms to solve them. Therefore, it is widely believed that P is a proper subset of NP. However, the proof of this conjecture seems to be out of reach by currently available mathematical techniques.[2]

This motivated the search for hardest problems within NP, the now well-known NP-complete problems. To define the notion of NP-completeness, we need the concept of complexity-bounded reduction, which is an important tool in complexity theory to compare the difficulty of problems. A reduction is a transformation of one problem into another problem. There are different kinds of reduction. The most important one is the polynomial-time many-one reduction.[3] A problem $A$ is *polynomial-time many-one reducible* to a problem $B$ (written $A \leq_{\mathrm{m}}^{\mathrm{p}} B$) if and only if there exists a polynomial-time computable function $f$ such that for every input $x \in \Sigma^*$, $x \in A \Leftrightarrow f(x) \in B$. If $A \leq_{\mathrm{m}}^{\mathrm{p}} B$ then problem $A$ is not much harder than problem $B$, because any algorithm for $B$ can, via the reducing function $f$, be used to solve problem $A$. In particular, if $B \in$ P then also $A \in$ P. Equivalently, $A \notin$ P implies $B \notin$ P.

A problem $A$ is NP-complete if and only if (i) *every* problem in NP can be reduced to it in polynomial time (i.e., $A$ is NP-hard ), and (ii) $A$ is in NP. Identifying a problem $A$ as NP-complete is useful for two purposes. First, it makes $A$ a good candidate that can be used to prove the strict inclusion of P in NP. Second, a proof that $A$ is NP-complete provides strong evidence that $A$ cannot be solved

[1] Here, we don't care about the difficulty of actually *finding* such a proof.

[2] There are many other similar questions we are currently unable to solve. For example, there is the NL $\overset{?}{=}$ L problem regarding the relative power of nondeterministic versus deterministic log-space computations. The general belief is that NL $\neq$ L.

[3] All reductions in this thesis are polynomial-time many-one reductions, unless mentioned otherwise.

by any deterministic polynomial-time algorithm, because a polynomial-time algorithm for an NP-complete problem $A$ would imply that *every* problem in $A$ can be solved in polynomial time, i.e., would imply $\mathrm{P} = \mathrm{NP}$.

The theory of NP-completeness was initiated by the work of Cook, Karp, and Levin. Cook [Coo71] proved that the boolean formula satisfiability problem SAT is NP-complete.[4] SAT is the following decision problem: Given a boolean formula, does there exist an assignment of the variables that satisfies the formula? By a reduction from the satisfiability problem, he proved the NP-completeness of the subgraph isomorphism problem. Building on Cook's result, Karp [Kar72] showed that 20 other natural problems (among them, for instance, the above-mentioned problem `Vertex Cover`) are NP-complete as well, thus demonstrating that NP-completeness is a common phenomenon. We remark that Levin [Lev73] independently obtained similar results. Since the 1970's, thousands of problems have been proved NP-complete [GJ79].

Let us come back to the `Vertex cover` problem given above. As mentioned there, `Vertex Cover` is NP-complete. We change the question slightly. Given a graph $G$, we want to know if the minimum vertex cover size $\tau(G)$ is odd. Call this problem `Odd Minimum Vertex Cover`. What is the complexity of `Odd Minimum Vertex Cover`? It can be shown that this problem is NP-hard. However, it is not clear whether this problem is also NP-complete, i.e., whether it is contained in NP. We try to give an intuitive explanation of why this problem may be harder than those contained in NP. It is certainly easy to give an efficiently verifiable proof for the assertion that a graph $G$ has a vertex cover of size *not larger* than a given $k$. But this obviously gives us no information about the *parity* of the minimum vertex cover size $\tau(G)$. Other attempts to place this problem into NP failed similarly.

There is a complexity-theoretic explanation for this failure: The `Odd Minimum Vertex Cover` problem has been proved by Wagner [Wag87] complete for the complexity class $\mathrm{P}_{\|}^{\mathrm{NP}}$, a class that is strongly believed to be a strict superset of NP. The class $\mathrm{P}_{\|}^{\mathrm{NP}}$ is defined to be the set of problems that can be solved in polynomial time with a number of parallel queries to an NP oracle (see Section 2 for formal definitions). The notion of $\mathrm{P}_{\|}^{\mathrm{NP}}$-completeness is analogous to the one of NP-completeness. That is, a problem $A$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete if and only if (i) *every* problem in $\mathrm{P}_{\|}^{\mathrm{NP}}$ is polynomial-time many-one reducible to $A$ (i.e., $A$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard ), and (ii) $A$ is contained in $\mathrm{P}_{\|}^{\mathrm{NP}}$. The significance of this definition is also analogous to the NP case: A problem that is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete cannot be contained, for example,

[4]The mere existence of NP-complete sets is no surprise and can be seen much easier by the canonical complete problem $\{\langle N, w, 0^t\rangle\} \mid N$ is an encoding of a nondeterministic Turing machine that accepts $w$ within $t$ steps$\}$. However, the existence of this NP-complete problem alone does not give any hint that there exists any NP-complete problem that we might be interested to solve in the real world.

in the smaller class NP, unless $\mathrm{P}_{\|}^{\mathrm{NP}} = \mathrm{NP}$.[5]

Let us first show that `Odd Minimum Vertex Cover` is indeed contained in $\mathrm{P}_{\|}^{\mathrm{NP}}$. We have to describe a polynomial-time algorithm that solves the `Odd Minimum Vertex Cover` problem with the help of a number of parallel queries to an oracle in NP. As oracle, we take the problem `Vertex Cover`, which we know is in NP. Let $G$ be any given graph with $n$ vertices. Clearly, the minimum vertex cover size $\tau(G)$ is less than $n$. Make the following list of queries: $\langle G, 0\rangle, \langle G, 1\rangle, \dots, \langle G, n-1\rangle$. Give this list of queries to the oracle `Vertex Cover`. The oracle returns the answers to the questions, say, "no", "no", ..., "no", "yes", ..., "yes".[6] With this list of answers in hand, it is trivial to determine $\tau(G)$. Accept $G$ if and only if $\tau(G)$ is odd. In our example graph $G$ of Figure 1.1, we have as list of queries $\langle G, 0\rangle, \langle G, 1\rangle, \langle G, 2\rangle, \langle G, 3\rangle, \langle G, 4\rangle, \langle G, 5\rangle$. The `Vertex Cover` oracle returns as list of answers "no", "no", "no", "yes", "yes", "yes". We conclude that $\tau(G) = 3$. Hence $G \in$ `Odd Minimum Vertex Cover`

Before turning to the $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness of `Odd Minimum Vertex Cover`, let us mention some basic facts about $\mathrm{P}_{\|}^{\mathrm{NP}}$. The class $\mathrm{P}_{\|}^{\mathrm{NP}}$ has several equivalent characterizations (see [Wag90]). The most central one is the characterization of $\mathrm{P}_{\|}^{\mathrm{NP}}$ as $\mathrm{P}^{\mathrm{NP[log]}}$, the class of problems that can be decided by a logarithmic number of adaptive queries[7] to an NP-oracle [Hem89, KSW87, BH91].[8] It follows immediately from the definition of $\mathrm{P}_{\|}^{\mathrm{NP}}$ that $\mathrm{P}_{\|}^{\mathrm{NP}}$ lies between the first and second levels of the polynomial hierarchy, i.e., $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{P}_{\|}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}}$.

We return to the issue of $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness. How can we prove a problem $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard? Wagner [Wag87] provided a strong tool for proving $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness (see Lemma 3.2.16 in Section 3.2.3). This tool has been applied for a large number of problems; see the survey [HHR97b].

In Chapter 3, we present an alternative approach for proving $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness. Our starting point is a characterization of the class $\mathrm{P}^{\mathrm{NP}}$ by nondeterministic Turing machines with a special acceptance type, which goes back to Krentel [Kre88]. Turing machine computations can straightforwardly be encoded into boolean formulas. So we get $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete versions of the boolean satisfiability problem. To make the presentation easier, we take an encoding into boolean circuits as an intermediate step. (Various proofs of Cook's theorem in the literature make use of boolean circuits. These proofs are arguably easier to comprehend than the ones that encode NPTMs directly into boolean formulas. See, e.g., [ALR04].)

In Section 3.2, we prove that the three satisfiability problems `MAX TRUE 3SAT`

[5]And also not in coNP, since $\mathrm{P}_{\|}^{\mathrm{NP}}$ is closed under complement.

[6]It is easy to see that in this case there cannot be a "no" after a "yes."

[7]The queries may depend on the answers to previous queries. Hence they are "adaptive."

[8]The above proof that `Odd Minimum Vertex Cover` is contained in $\mathrm{P}_{\|}^{\mathrm{NP}}$ can also be carried out using the characterization as $\mathrm{P}^{\mathrm{NP[log]}}$. Here, a binary search along the number of vertices is performed.

COMPARE, MAX TRUE 3SAT EQUALITY, and ODD MAX TRUE 3SAT are $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete. These are variants of the classical NP-complete satisfiability problem 3SAT. We introduce the following notation. For any 3-CNF formula $F$ (boolean formula in conjunctive normal form having no more than 3 literals in every clause), let max-$\mathbb{1}(F)$ denote the maximum number of variables set to true (the number of 1's) in satisfying assignments for $F$. The problem MAX TRUE 3SAT COMPARE is the following: Given two 3-CNF formulas $F_1$ and $F_2$, is it true that max-$\mathbb{1}(F_1) \leq$ max-$\mathbb{1}(F_2)$? The problem MAX TRUE 3SAT EQUALITY is defined analogously, but with the question "max-$\mathbb{1}(F_1) =$ max-$\mathbb{1}(F_2)$?". For the ODD MAX TRUE 3SAT problem, we have as input one 3-CNF formula $F$, and the problem is to decide if max-$\mathbb{1}(F)$ is odd.

As in the theory of NP-completeness, these satisfiability problems are useful for proving $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness, because they serve well as "first problems" for reductions. In Section 3.3, we give an example of such a reduction regarding a variant of the the Vertex Cover problem. In particular, we prove that the problem Minimum Vertex Cover Compare is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete by a reduction from MAX TRUE 3SAT COMPARE. We note that the $\mathrm{P}_{\|}^{\mathrm{NP}}$-completeness of Minimum Vertex Cover Compare was proved earlier by Wagner [Wag87] using his sufficient condition for $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness. The Minimum Vertex Cover Compare problem will be used as a starting point for the reductions in Chapters 4, 5, and 6.

In Section 3.4, we apply the ideas of Section 3.2 to prove $\mathrm{P}^{\mathrm{NP}}$-completeness of the satisfiability problems MAX LEX 3SAT COMPARE and MAX LEX 3SAT EQUALITY. These decision problems are similar to MAX TRUE 3SAT COMPARE and MAX TRUE 3SAT EQUALITY. Here, we compare the lexicographic maximum satisfying assignments of two given formulas. By reductions from these decision problems, we also prove that the following problem is $\mathrm{P}^{\mathrm{NP}}$-complete: Given two instances of the Traveling Salesperson problem, decide if the length of an optimal tour for the first instance is less than or equal to the length of an optimal tour for the second instance.

In Chapters 4 and 5, we are concerned with computational complexity aspects of voting schemes. What kind voting schemes do we consider? We consider preferential elections, where voters rank the candidates (or more generally: "alternatives"[9]) from most preferred to least preferred. As an example, see the following four-voter *profile* (i.e., a listing of all voter's preferences), where the candidates are $a$, $b$, $c$, and $d$.

[9]We always use the term "candidate." But of course, voting schemes can be applied in all situations where groups of people have to make a choice among a set of options or alternatives, for example the choice of a holiday destination, of what movie to watch next, of where to dine out, etc.

**Example 1**

$$\begin{array}{lllllllll} \text{Voter} & 1: & a & > & b & > & c & > & d, \\ \text{Voter} & 2: & a & > & b & > & c & > & d, \\ \text{Voter} & 3: & d & > & a & > & b & > & c, \\ \text{Voter} & 4: & c & > & d & > & a & > & b, \\ \text{Voter} & 5: & b & > & c & > & d & > & a. \end{array}$$

Voter 1 and Voter 2 rank candidate $a$ best, candidate $b$ second best, and candidate $d$ last, etc. Who is the winner of such an election? That depends on the voting scheme that is used. A *voting scheme* is a rule for how to determine the winner(s) of an election that is given by the preferences of the voters. There is no unique best rule for determining winners. Look at the following voter profile:

**Example 2**

$$\begin{array}{llllll} \text{40\% of the voters:} & a & > & b & > & c, \\ \text{30\% of the voters:} & b & > & c & > & a, \\ \text{30\% of the voters:} & c & > & b & > & a. \end{array}$$

Should $a$ be declared the winner? It is not clear. Candidate $a$ is the candidate with most first-ranked votes (40%). On the other hand, 60% view $a$ as the least desirable choice.

Social choice theorists have suggested quite a large number of different voting schemes. In this thesis, we consider only three voting schemes: Young's voting scheme, Dodgson's voting scheme, and Kemeny's voting scheme. These three voting schemes are so-called Condorcet voting schemes. A *Condorcet voting scheme* is a voting scheme that respects the Condorcet principle [Con85]. A *Condorcet winner* is a candidate who beats every other candidate in pairwise contest. Candidate $a$ beats candidate $b$ in pairwise contest if and only if strictly more than half of the voters prefer $a$ to $b$. A voting scheme *respects the Condorcet principle* if it elects the Condorcet winner[10] whenever one exists. The election given in Example 2 has candidate $b$ (and not $a$!) as Condorcet winner: (i) $b$ beats $a$, because 60% of the voters prefer $b$ to $a$, and (ii) $b$ beats $c$, because 70% of the voters prefer $b$ to $c$. However, a Condorcet winner does not always exist for a given voter profile. For example, it can easily be seen that there is no Condorcet winner for the election in Example 1.

We now give an informal description of the voting schemes treated in this thesis. Dodgson's voting scheme was proposed by Charles L. Dodgson (more commonly known by his pen name, Lewis Carroll) [Dod76]. Given the preference orders of the voters, the winners according to Dodgson's voting scheme are determined as follows. For each candidate, determine the number of adjacent switches in the voters' preference orders that are necessary to make the candidate a Condorcet winner. The candidates with the fewest required switches are the Dodgson winners.

[10]It is easy to see that a Condorcet winner is uniquely determined if one exists.

Young [You77] suggested a voting scheme that is similar to Dodgson's voting scheme in the sense that it is also based on altered voter profiles. It works as follows. For each candidate, determine the number of voter preferences that need to be removed to make the candidate a Condorcet winner. The candidate with the fewest required removals is the Young winner.

Kemeny [Kem59] introduced a voting scheme that is based on the notion of consensus ranking. A consensus ranking is a preference order that is closest to the preferences of the voters. The winners according to Kemeny's voting scheme are the candidates that are first-ranked in some consensus ranking. Here, distance between two given preferences is defined by the number of unordered pairs of candidates that are ranked differently in these preferences.

We are concerned with the computational complexity of determining the winning candidates under these voting schemes. The investigation of voting schemes with respect to their computational properties was initiated more than a decade ago by Bartholdi, Tovey, and Trick [BTT89b, BTT89a, BTT92]. They proved (among other related problems) that the winner problems for Dodgson's and Kemeny's voting schemes are NP-hard. They left open the problem of whether these winner problems are also NP-complete, i.e., if they are contained in NP. Hemaspaandra, Hemaspaandra, and Rothe [HHR97a] resolved this question for Dodgson's voting scheme. In particular, they improved Bartholdi, Tovey, and Trick's NP-hardness result to $\mathrm{P}_{\|}^{\mathrm{NP}}$-completeness.[11] This implies that the winner problem for Dodgson elections is not in NP, unless $\mathrm{P}_{\|}^{\mathrm{NP}}$ coincides with NP.

We extend this line of research. In Chapter 4, we prove that the winner problem for Young's voting scheme is $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard. To this end, we give a reduction from the `Maximum Set Packing Compare` problem, which in turn is easily shown $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete by reduction from the `Minimum Vertex Cover Compare` problem. Furthermore, we investigate a homogeneous variant of Dodgson's voting scheme. We show that this variant of Dodgson's voting scheme can be solved efficiently by a linear program that is based on an integer linear program given by Bartholdi et al. [BTT89b].

Bartholdi et al. [BTT89b] proved that the winner problem for Kemeny's voting scheme is NP-hard by a reduction from the classical NP-complete digraph problem `Feedback Arc Set`. In Chapter 5, we improve this NP-hardness result to $\mathrm{P}_{\|}^{\mathrm{NP}}$-completeness. To this end, we define the new problems `Feedback Arc Set Member` and `Vertex Cover Member`, and prove them $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete by reducing from the `Minimum Vertex Cover Compare` problem. We adapt the reduction given by Bartholdi et al. so that it becomes a reduction from `Feedback Arc Set Member` to the problem `Kemeny Winner`.

These results raise the question of what is special about $\mathrm{P}_{\|}^{\mathrm{NP}}$ that all these voting schemes are complete for this class. The above-mentioned voting schemes

[11]For this to make sense, it is necessary to phrase the winner problem appropriately as a decision problem.

have in common that they assign a hard-to-compute score to each candidate, and the candidates with lowest (or highest in the case of Young's voting system) score are the winners. The value of the score is in all three cases polynomially bounded in the size of the input. That easily yields $\mathrm{P}_{\|}^{\mathrm{NP}}$ as an upper bound. We know that "comparison versions" of NP-hard optimizations problems are often complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$ (such as the `Minimum Vertex Cover Compare` problem). That gives us some hint that these voting schemes might be $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete. However, it is offhand not clear whether the voting system at hand is *indeed* complete for this class.

In Chapter 6, we investigate heuristics for the minimum vertex cover problem. Two of the most prominent such heuristics are the *edge deletion heuristic* and the *maximum-degree greedy heuristic*, see, e.g., [PS82, Pap94]. For both heuristics, we study the problem of recognizing those graphs for which that heuristic can approximate the size of a minimum vertex cover within a constant factor of $r$, where $r$ is a fixed rational number. To this end, we introduce the decision problems $\mathcal{S}_r^{\mathrm{ED}}$ and $\mathcal{S}_r^{\mathrm{MDG}}$. For any fixed rational $r \geq 1$, $\mathcal{S}_r^{\mathrm{ED}}$ (respectively, $\mathcal{S}_r^{\mathrm{MDG}}$) is the class of graphs for which the edge deletion heuristic (respectively, the maximum-degree greedy heuristic) can output a vertex cover of size at most $r$ times the size of a minimum vertex cover. We prove that these problems are $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete if $r$ is from a suitable range. As in the preceding chapters, we obtain $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness by a reduction from the `Minimum Vertex Cover Compare` problem. To achieve the $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness result of $\mathcal{S}_r^{\mathrm{MDG}}$, we modify a construction by Papadimitriou and Steiglitz [PS82] that they use to analyze the worst-case approximation behavior of the maximum-degree greedy heuristic.

The analogous decision problem $\mathcal{S}_r$ for the maximum independent set problem has been studied earlier by Bodlaender, Thilikos, and Yamazaki [BTY97]. They proved that $\mathcal{S}_r$ is coNP-hard and belongs to $\mathrm{P}^{\mathrm{NP}}$. Closing the gap between these lower and upper bounds, Hemaspaandra and Rothe [HR98] proved that $\mathcal{S}_r$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete, see also the survey by E. Hemaspaandra, L. Hemaspaandra, and Rothe [HHR97b].

In Chapter 7, we study complexity classes that are based on counting the number of accepting and rejecting computation paths of NPTMs. Valiant [Val79] introduced the famous class #P, which is the set of all functions that can be defined by the number of accepting paths of some NPTM. He proved several natural problems complete for #P, for example the problem of computing the permanent of a zero-one matrix. Fenner, Fortnow, and Kurtz [FFK94] generalized #P to GapP and developed a theory of gap-definable counting classes.[12] The class GapP is the set of functions that can be defined by the difference (the "*gap*") between the number of accepting and rejecting paths of an NPTM. Many prominent counting classes, including PP, $\oplus$P, and $\mathrm{C}_{=}\mathrm{P}$, can be conveniently characterized in terms of GapP functions. Fenner et al. defined the new complexity classes SPP, LWPP,

[12] Gupta [Gup95] defined independently the same class under the name $\mathcal{Z}\#\mathrm{P}$.

and WPP.[13] The class SPP is the "gap analog" of the well-known complexity class UP: The definition of SPP is via GapP functions instead of #P functions. Since every #P function is a GapP function, we have UP $\subseteq$ SPP. The class SPP is known to contain an important natural problem—the graph isomorphism problem [AK02]. Arvind and Vinodchandran [AV97] and Vinodchandran [Vin04] showed that many group-theoretic computational problems are in SPP or LWPP.

Fenner, Fortnow, and Kurtz [FFK94] introduced the notion of gap-definability. A *gap-definable* counting class is a collection of all sets such that, for any set in the class, the membership of a string in the set depends (in a way particular to the class) on the difference (gap) between the number of accepting and rejecting paths produced by some NPTM associated with the set. A formal definition of gap-definability is given in terms of GapP functions and disjoint sets $A, R \subseteq \Sigma^* \times \mathbb{Z}$ (see upcoming Definition 7.5.2). Based on the mechanism of relativizing this definition, Fenner et al. suggested two ways of defining gap-definability for a relativizable class: *uniform* and *nonuniform* gap-definability. A relativizable class is *uniformly* gap-definable if it is gap-definable in every relativized world, where the choice of $A$ and $R$ is fixed and is independent of the oracle. On the other hand, a relativizable class is *nonuniformly* gap-definable if it is gap-definable in every relativized world, where the choice of $A$ and $R$ may depend on the oracle. Examples of *uniformly* gap-definable counting classes are PP, $\mathrm{C}_{=}\mathrm{P}$, $\oplus$P, and SPP. Fenner et al. prove that also LWPP and WPP are *nonuniformly* gap-definable, but leave the question open of whether these classes are also *uniformly* gap-definable [FFK94]. Fenner et al. proved that SPP is low for GapP. (A class $\mathcal{D}$ is *low* for a class $\mathcal{C}$ if $\mathcal{C}^{\mathcal{D}} \subseteq \mathcal{C}$.) This implies that SPP is low for every *uniformly* gap-definable counting class.

We prove that there is a relativized world where UP$\cap$coUP (and hence SPP) is low neither for LWPP nor for WPP. As a consequence, we show that both LWPP and WPP are not uniformly gap-definable. This settles the above mentioned question by Fenner et al. To get this result, we prove a new combinatorial property of low-degree multilinear polynomials (Section 7.3), and make use of the well-known polynomial encoding technique.

Using a similar proof technique, we furthermore construct a relativized world where WPP is not closed under polynomial-time Turing reductions. This gives a relativized answer to another open question of Fenner et al. [FFK94]: Relativizable proof techniques are not sufficient to prove that WPP is closed under polynomial-time Turing reductions.

[13]SPP was independently introduced by Gupta [Gup95] under the name $\mathcal{Z}$UP, and by Ogiwara and Hemachandra [OH93] under the name XP. The first SPP machine is implicit in a paper by Köbler, Schöning, Toda, and Torán [KSTT92].

# Chapter 2

# Preliminaries

## 2.1 Strings and Languages

Let $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{Z}$ denote the set of nonnegative integers, rational numbers, and integers, respectively.

Fix the two-letter alphabet $\Sigma = \{0,1\}$. Let $\Sigma^*$ be the set of all finite length strings over $\Sigma$. For every $n \in \mathbb{N}$, $\Sigma^n$ denotes the set of all strings of length $n$ in $\Sigma^*$. For any $n \in \mathbb{N}$, and any $x \in \Sigma^*$, $x\Sigma^n = \{xw \mid w \in \Sigma^n\}$. For any $x \in \Sigma^*$, $|x|$ denotes the length of the string $x$, while $|x|_0$ and $|x|_1$ denote, respectively, the number of 0's and the number of 1's in $x$. For any $A \subseteq \Sigma^*$ and $n \in \mathbb{N}$, $A^{=n}$ is the set of strings of length $n$ in $A$ and $A^{\leq n}$ is the set of strings of length at most $n$ in $A$. The complement of a language $A \subseteq \Sigma^*$ is denoted by $\overline{A}$ and is defined by $\overline{A} = \Sigma^* - A$.

For any $x \in \Sigma^*$, the integer $\mathrm{number}(x)$ is defined as the value of the binary number $1x$. We obtain the standard lexicographic ordering $\leq$ on $\Sigma^*$ by defining for any $u, v \in \Sigma^*$ that $u \leq v$ if and only if $\mathrm{number}(u) \leq \mathrm{number}(v)$. For any $x \in \Sigma^*$, let $\mathrm{pos}(x) = \mathrm{number}(x) - 2^{|x|}$ represent the lexicographic rank of $x$ among the strings of length $|x|$. For example, $\mathrm{pos}(000) = 0$ and $\mathrm{pos}(010) = 2$.

For any set $A$, let $\|A\|$ denote the number of elements of $A$, and let $\chi_A$ denote the *characteristic function* of $A$, i.e., $\chi_A(x) = 1$ if $x \in A$, and $\chi_A(x) = 0$ if $x \notin A$.

For any $n \in \mathbb{N}$, let $[n] \overset{\mathrm{df}}{=} \{1, 2, \ldots, n\}$. For $A, B \subseteq \Sigma^*$, define the marked (or disjoint) union of $A$ and $B$ by $A \oplus B = \{0w \mid \in A\} \cup \{1w \mid w \in B\}$.

Let $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ be a standard pairing function, i.e., $\langle \cdot, \cdot \rangle$ is a total, one-to-one, polynomial-time computable function that has polynomial-time computable inverses.

## 2.2 Machines

Our complexity considerations are based on the Turing machine model. For formal definitions of Turing machines, see any textbook about computational complexity (e.g, [BC93, HO02, Pap94, Rot05]). A *Turing machine* has an input tape and a fixed finite number of work tapes. If the Turing machine is used to compute functions then it has additionally an output tape. Turing machines that compute functions are often called *transducers*.

We distinguish deterministic and nondeterministic Turing machines.

Each configuration of a deterministic Turing machine has at most one successor configuration, which is determined uniquely by the finite transition table of the machine. A set $A$ is accepted by a deterministic Turing machine $M$ if for every $x \in \Sigma^*$, $x \in A$ if and only if $M$ working on input $x$ reaches an accepting configuration (a configuration with an accepting state).

A nondeterministic Turing machine may have more than one successor configuration. The computation of a nondeterministic Turing machine on an input string can conveniently be described by a *computation tree*: The root node of the tree is the start configuration of the machine. Each node corresponds to a configuration reachable from the start configuration in a finite number of steps. The child nodes of any node represent the successor configurations of that node. The computation tree of a nondeterministic Turing machine is determined by the input and the finite transition table of the machine. We use the shorthand $N(x)$ to denote the computation of machine $N$ on input string $x$. A *computation path* $\rho$ of a nondeterministic Turing machine with input $x$ is any sequence of configurations, where the configurations correspond to the nodes of a path in the computation tree of $N(x)$ that starts with the root node. A nondeterministic Turing machine $N$ accepts the string $x$ if and only if $N$ has a computation path that ends with an accepting configuration. A set $A$ is accepted by a nondeterministic Turing machine $N$ if for every $x \in \Sigma^*$, $x \in A$ if and only if $N$ accepts $x$.

Every deterministic Turing machine is by definition also a nondeterministic Turing machine. The computation tree of a deterministic Turing machine is degenerated to a single path.

As we are concerned with computational complexity investigations, we use Turing machines with time bounds. In particular, we consider polynomial-time bounded deterministic Turing machines (DPTMs) and polynomial-time bounded nondeterministic Turing machines (NPTMs). A DPTM is a deterministic Turing machine that for a fixed polynomial $p$ makes at most $p(|x|)$ computation steps on any input $x$ before reaching an (accepting or rejecting) final configuration. An NPTM is a nondeterministic Turing machine with the property that every computation path on input $x$ has length at most $p(|x|)$ for some fixed polynomial $p$.

Now we describe the important concept of computation relative to an oracle. An oracle Turing machine $N$ is a Turing machine that can make use of external information provided by a set of strings $A \subseteq \Sigma^*$. Such a machine can make queries to the oracle set $A$ in the following way: It writes down a string $q$ on an additional tape, called the query tape, and changes to a special query state $z_?$. If $q \in A$, then the state of $N$ changes to $q_{yes}$, otherwise to state $q_{no}$. The contents of the query tape is erased. We say that $N$ computes relative to oracle $A$. Both deterministic and nondeterministic Turing machines can be defined relative to an oracle. All definitions made so far transfer easily to oracle Turing machines. We write $N^A(x)$ for the computation of oracle Turing machine $N$ working on input $x$ with oracle $A$. We make the convention that the computation paths of $N^A(x)$ include the query strings and answers from the oracle $A$. We write $N^{(\cdot)}$ for an oracle Turing machine with unspecified oracle.

For any (deterministic or nondeterministic) Turing machine $N$, let $L(N)$ be the language accepted by $N$. For any (deterministic or nondeterministic) oracle Turing machine $N$ with oracle $A$, let $L(N^A)$ be the language accepted by machine $N$ computing relative to oracle $A$.

## 2.3 Some Complexity Classes

We introduce some complexity classes based on the above machines. All the notations are standard. See any complexity theory textbook for further explanation.

P is the class of languages that are accepted by a deterministic polynomial-time Turing machines. NP is the class of languages accepted by some nondeterministic polynomial-time Turing machine. Let FP denote the class of polynomial-time computable functions.

These classes can be relativized in a natural way. For any $A \subseteq \Sigma^*$, let $\mathrm{P}^A$ be the class of all languages $L$ such that there exists an oracle DPTM $M$ with $L = L(M^A)$. Likewise, let $\mathrm{NP}^A$ be the class of all languages $L$ such that there exists an oracle NPTM $N$ with $L = L(N^A)$. For any $A \subseteq \Sigma^*$, let $\mathrm{FP}^A$ be the set of functions computable by a polynomial-time bounded oracle transducer with the help of oracle $A$. We extend these definitions to the notion of computation relative to a complexity class $\mathcal{C}$: Let $\mathrm{P}^{\mathcal{C}} = \bigcup_{A\in\mathcal{C}} \mathrm{P}^A$, $\mathrm{NP}^{\mathcal{C}} = \bigcup_{A\in\mathcal{C}} \mathrm{NP}^A$, and $\mathrm{FP}^{\mathcal{C}} = \bigcup_{A\in\mathcal{C}} \mathrm{FP}^A$.

For any complexity class $\mathcal{C}$, co$\mathcal{C}$ is defined by $\mathrm{co}\mathcal{C} = \{A \mid \overline{A} \in \mathcal{C}\}$.

The polynomial hierarchy is defined as follows:

**Definition 2.3.1 (Polynomial Hierarchy [MS72, Sto76])** *The* polynomial hierarchy *is inductively defined as follows:*

- $\Delta_0^{\mathrm{p}} = \Sigma_0^{\mathrm{p}} = \Pi_0^{\mathrm{p}} = \mathrm{P}$,
- *For every* $k \geq 0$, $\Delta_{k+1}^{\mathrm{p}} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}}$, $\Sigma_{k+1}^{\mathrm{p}} = \mathrm{NP}^{\Sigma_k^{\mathrm{p}}}$, *and* $\Pi_{k+1}^{\mathrm{p}} = co\Sigma_{k+1}^{\mathrm{p}}$,
- $\mathrm{PH} = \bigcup_{k\geq 0} \Sigma_k^{\mathrm{p}}$.

*In particular,* $\Delta_1^{\mathrm{p}} = \mathrm{P}$, $\Sigma_1^{\mathrm{p}} = \mathrm{NP}$, *and* $\Pi_1^{\mathrm{p}} = \mathrm{coNP}$

**Theorem 2.3.2 ([MS72, Sto76])**

1. *For each* $k \geq 0$, $\Sigma_k^{\mathrm{p}} \cup \Pi_k^{\mathrm{p}} \subseteq \Delta_{k+1}^{\mathrm{p}} \subseteq \Sigma_{k+1}^{\mathrm{p}} \cap \Pi_{k+1}^{\mathrm{p}}$.
2. *For each* $k \geq 1$, $\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}}$ *implies*
$$\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}} = \Delta_{k+1}^{\mathrm{p}} = \Sigma_{k+1}^{\mathrm{p}} = \Pi_{k+1}^{\mathrm{p}} = \cdots = \mathrm{PH}.$$
3. *For each* $k \geq 0$, $\Sigma_k^{\mathrm{p}} = \Sigma_{k+1}^{\mathrm{p}}$ *implies*
$$\Sigma_k^{\mathrm{p}} = \Pi_k^{\mathrm{p}} = \Delta_{k+1}^{\mathrm{p}} = \Sigma_{k+1}^{\mathrm{p}} = \Pi_{k+1}^{\mathrm{p}} = \cdots = \mathrm{PH}.$$

## 2.4 Reductions and Hardness

Reductions can be used to compare the hardness of languages. In this thesis we need the following kinds of polynomial-time bounded reductions.

- For languages $A$ and $B$, we say that $A$ is *polynomial-time many-one reducible* to $B$ ($A \leq_{\mathrm{m}}^{\mathrm{p}} B$) if and only if there exists a polynomial-time computable function $f$ such that for all inputs $x \in \Sigma^*$, $x \in A$ if and only if $f(x) \in B$.
- For languages $A$ and $B$, we say that $A$ is *polynomial-time Turing reducible* to $B$ ($A \leq_{\mathrm{T}}^{\mathrm{p}} B$) if and only if there exists an oracle DPTM $M$ such that $A = L(M^B)$.
- We say that $A$ is *polynomial-time truth-table reducible* to $B$ ($A \leq_{\mathrm{tt}}^{\mathrm{p}} B$) if and only if there exists a DPTM $M$ and a polynomial-time computable function $f$ such that, for each $x \in \Sigma^*$ , there exists an integer $m$ such that
    1. $f(x) = \langle q_1, q_2, \ldots q_m \rangle$, and
    2. $M(\langle x, \chi_B(q_1), \chi_B(q_2), \ldots, \chi_B(q_m) \rangle)$ accepts if and only if $x \in A$.

For any reduction $\leq_b^a$ defined above and any complexity class $\mathcal{C}$, a set $A$ is called $\leq_b^a$-hard for $\mathcal{C}$ if and only if for all $B \in \mathcal{C}$, $B \leq_b^a A$. A set $A$ is called $\leq_b^a$-complete for $\mathcal{C}$ if and only if $A$ is $\leq_b^a$-hard for $\mathcal{C}$ and $A \in \mathcal{C}$. We denote by

$$\mathrm{R}_b^a(\mathcal{C}) = \{L \mid (\exists B \in \mathcal{C})[L \leq_b^a B]\}$$

the *reducibility closure* of $\mathcal{C}$ with respect to $\leq_b^a$. A complexity class $\mathcal{C}$ is *closed under* $\leq_b^a$ if and only if $\mathrm{R}_b^a(\mathcal{C}) \subseteq \mathcal{C}$.

Unless stated otherwise, the hardness and completeness results in this thesis are with respect to the polynomial-time many-one reducibility.

We remark that all complexity classes in this thesis are closed under polynomial-time many-one reductions.

## 2.5 Graphs

All undirected graphs are simple, i.e., without multiple edges or loops. For any graph $G$, let $V(G)$ denote the set of vertices of $G$, and let $E(G)$ denote the set of edges of $G$. For any vertex $v \in V(G)$, the *neighborhood* of $v$ (denoted $N(v)$) is the set of vertices in $G$ that are adjacent to $v$. For any vertex $v \in V(G)$, the *degree of* $v$ (denoted by $\deg_G(v)$) is the number of vertices adjacent to $v$ in $G$; if $G$ is clear from the context, we omit the subscript and simply write $\deg(v)$. Let $\Delta(G) = \max_{v \in V(G)} \deg(v)$ denote the maximum degree of the vertices of graph $G$.

**Definition 2.5.1 (Vertex Cover)** • *For any graph $G$, a subset $C \subseteq V(G)$ is a* vertex cover of $G$ *if for all edges $\{v,w\} \in E(G)$, $\{v,w\} \cap C \neq \emptyset$.*

- *A vertex cover is said to be a* minimum vertex cover of $G$ *if it is of minimum size. For any graph $G$, let $\tau(G)$ denote the size of a minimum vertex cover of $G$.*
- *The problem* `Vertex Cover` *is defined as follows:*

$$\texttt{Vertex Cover} = \{\langle G,k\rangle \mid G \textit{ is a graph and } k \in \mathbb{Z} \textit{ such that } \tau(G) \leq k\}.$$

**Definition 2.5.2 (Independent Set)** • *For any graph $G$, a subset $I \subseteq V(G)$ is an* independent set of $G$ *if for all $u,v \in I$, $\{u,v\} \notin E(G)$.*

- *An independent set is said to be a* maximum independent set of $G$ *if it is of maximum size. For any graph $G$, let $\alpha(G)$ denote the size of a maximum independent set of $G$.*
- *The problem* `Independent Set` *is defined as follows:*

$$\texttt{Independent Set} = \{\langle G,k\rangle \mid G \textit{ is a graph and } k \in \mathbb{Z} \textit{ such that } \alpha(G) \geq k\}.$$

**Definition 2.5.3 (Disjoint union, join)** *Let $G$ and $H$ be two disjoint graphs.*

- *The* disjoint union of $G$ and $H$ *is defined to be the graph $U = G \cup H$ with vertex set $V(U) = V(G) \cup V(H)$ and edge set $E(U) = E(G) \cup E(H)$.*
- *The* join of $G$ and $H$ *is defined to be the graph $J = G \bowtie H$ with vertex set $V(J) = V(G) \cup V(H)$ and edge set $E(J) = E(G) \cup E(H) \cup \{\{x,y\} \mid x \in V(G) \ \wedge \ y \in V(H)\}$.*

**Definition 2.5.4 (Hamilton Cycle)**

- *A* Hamilton cycle *in an undirected graph is a cycle that visits every vertex of the graph exactly once. The problem* `Undirected Hamilton Cycle` *contains exactly those undirected graphs that have a Hamilton cycle.*
- *A* Hamilton cycle *in a directed graph is a directed cycle that visits every vertex of the graph exactly once. The problem* `Directed Hamilton Cycle` *contains exactly those directed graphs that have a Hamilton cycle.*

## 2.6 Boolean Functions, Circuits, and Formulas

A *boolean function* is a mapping $\phi : \{0,1\}^n \rightarrow \{0,1\}$. A *truth assignment* for a boolean function $\phi(x_1, x_2, \ldots, x_n)$ assigns "true" or "false" (i.e., 1 or 0) to each variable $x_1, x_2 \ldots, x_n$. A truth assignment *satisfies* $\phi$ if it makes $\phi$ true, i.e., if $\phi(x_1, x_2, \ldots, x_n) = 1$ with the given assignment. A function $\phi$ is *satisfiable* if there is some truth assignment that satisfies it.

Standard ways of representing boolean functions are representations as boolean circuits and boolean formulas. A *boolean circuit* $C(x_1, x_2, \ldots, x_n)$ is a directed acyclic graph, where each vertex with nonzero indegree (the gates) is labeled with boolean connectives $\wedge$, $\vee$, $\neg$, [1] and each vertex with indegree 0 is labeled by a variable in $\{x_1, x_2, \ldots, x_n\}$ (the input variables of the circuit). One of the vertices is distinguished as ouput gate. An input assignment $y_1, y_2, \ldots, y_n$ to $x_1, x_2, \ldots, x_n$ maps each input vertex to 0 or 1. The value of each gate is obtained by applying the boolean connective given by the label to the values of its immediate predecessors. The value $C(y_1, y_2, \ldots, y_n)$ that is computed by the circuit $C$ for the assignment $y_1, y_2, \ldots, y_n$ is defined by the value of the output gate. Occasionally, we write $C(y)$ to denote $C(y_1, y_2, \ldots, y_n)$, where $y$ is the string $y_1 y_2 \ldots y_n$. The size of a circuit $C$ is the number of vertices in $C$.

A *boolean formula* is a circuit in which every gate has outdegree at most one. A boolean formula is in *CNF* (in *conjunctive normal form*) if it is a conjuction of clauses. A *clause* is a disjunction of literals. A *literal* is an occurrence of a variable or its negation. A formula is in *3-CNF* if it is in CNF, and each clause contains at most 3 literals. For example, the following formula is in 3-CNF: $(\neg x_3 \vee x_1 \vee \neg x_2) \wedge (x_3) \wedge (x_2 \vee x_3)$. Here, the clause $(\neg x_3 \vee x_1 \vee \neg x_2)$ contains the literals $\neg x_3$, $x_1$, and $\neg x_2$. The size of a CNF-formula $F$ is the number of occurrences of variables in $F$. An assignment satisfies a CNF-formula $F$ if it makes every clause in $F$ true, i.e., if it makes at least one literal in every clause true. The decision problem `3SAT` is defined by `3SAT` $= \{F \mid F$ is a 3-CNF formula that has a satisfying assignment$\}$.

[1] Circuits can be defined over other boolean connectives as well. The set of connectives chosen here is sufficient to represent all boolean functions. (That is, $\{\wedge, \vee, \neg\}$ is an example for a complete basis.)

# Chapter 3

# Parallel Access to NP

## 3.1 The class $\mathrm{P}_{\|}^{\mathrm{NP}}$

The class $\mathrm{P}^{\mathrm{NP}}$ contains exactly those languages that can be accepted by a deterministic polynomial-time Turing machine with the help from an oracle in NP. We obtain subclasses of $\mathrm{P}^{\mathrm{NP}}$ by restricting the oracle access of the DPTM. We obtain the class $\mathrm{P}_{\|}^{\mathrm{NP}}$ if we require the DPTM to make all queries to the NP oracle *in parallel.* That means the following. For any input, the machine computes a list of query strings and gives that list to the oracle. The oracle returns for each string in the list the answer (i.e, "yes" or "no" depending on the containment of the string in the oracle). Finally, the machine accepts or rejects the input string based on the answers it got from the oracle. Hence the queries to the oracle may not depend on the answers to previous queries. From the definitions it is immediate that $\mathrm{P}_{\|}^{\mathrm{NP}}$ coincides with $\mathrm{R}_{\mathrm{tt}}^{\mathrm{p}}(\mathrm{NP})$, the reducibility closure of NP with respect to polynomial-time truth-table reduction.

Papadimitriou and Zachos [PZ83] introduced the class $\mathrm{P}^{\mathrm{NP}[\log]}$, the set of all languages that can be accepted by a DPTM that makes at most $O(\log n)$ queries to an NP oracle. Here the queries may depend on the answers to previous queries.

It turned out that the class $\mathrm{P}^{\mathrm{NP}[\log]}$ coincides with $\mathrm{P}_{\|}^{\mathrm{NP}}$ (independently shown by Hemaspaandra [Hem89], Köbler, Schöning, Wagner [KSW87], Buss and Hay [BH91]).

There are a number of other natural characterizations of $\mathrm{P}_{\|}^{\mathrm{NP}}$: $\mathrm{P}_{\|}^{\mathrm{NP}}$ equals the logarithmic space classes $\mathrm{L}^{\mathrm{NP}}$, $\mathrm{L}^{\mathrm{NP}[\log]}$, and $\mathrm{L}_{\|}^{\mathrm{NP}}$ [Wag90]. Wagner [Wag90] generalized $\mathrm{P}_{\|}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}[\log]}$ to the polynomial hierarchy. For every $i \geq 1$, he defined $\Theta_{i+1}^{\mathrm{p}} = \mathrm{P}^{\Sigma_i^{\mathrm{p}}[\log]}$, the class of languages that can be decided in polynomial time with at most $O(\log n)$ queries to a $\Sigma_i^{\mathrm{p}}$-oracle. It is easy to see that $\Theta_i^{\mathrm{p}} \subseteq \Delta_i^{\mathrm{p}} \subseteq \Sigma_i^{\mathrm{p}} \cup \Pi_i^{\mathrm{p}} \subseteq \Theta_{i+1}^{\mathrm{p}} \subseteq \Delta_{i+1}^{\mathrm{p}}$. All these inclusions are strongly conjectured to be strict. Thus $\mathrm{P}_{\|}^{\mathrm{NP}} = \Theta_2^{\mathrm{p}}$ lies between the NP/coNP- and $\Delta_2^{\mathrm{p}}$-levels of the polynomial hierarchy.

The class $\mathrm{P}_{||}^{\mathrm{NP}}$ is of importance in the investigation of the existence of sparse Turing-complete sets for NP. Mahaney [Mah82] showed that if NP has a sparse polynomial-time Turing-complete set, then the polynomial hierarchy is contained in $\Delta_2^{\mathrm{p}}$. Kadin [Kad89] improved this result to a collapse to $\mathrm{P}_{||}^{\mathrm{NP}}$: If there exists a sparse polynomial-time Turing-complete set for NP, then $\mathrm{PH} \subseteq \mathrm{P}_{||}^{\mathrm{NP}}$. There is relativized evidence that this collapse is optimal [Kad89].

The naturalness of a complexity class is often judged by the existence of important complete problems. From the above characterizations of $\mathrm{P}_{||}^{\mathrm{NP}}$ it is not immediately clear how to prove a given problem $\mathrm{P}_{||}^{\mathrm{NP}}$-hard.[1] However, Wagner [Wag87] has provided a powerful tool for proving $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness (see Theorem 3.2.16). Using that tool, he was able to show that NP-hard optimizations problems often give rise to a $\mathrm{P}_{||}^{\mathrm{NP}}$-complete problem. Consider for instance the classical NP-complete problem `Vertex Cover`. Wagner [Wag87] has proved that the following vertex cover problems are $\mathrm{P}_{||}^{\mathrm{NP}}$-complete:

- Given a graph $G$, determine if a minimum vertex cover of $G$ has odd size (`Odd Minimum Vertex Cover` problem).
- Given two graphs $G_1$ and $G_2$, determine if the minimum vertex cover sizes of $G_1$ and $G_2$ equal (`Minimum Vertex Cover Equality` problem).
- Given two graphs $G_1$ and $G_2$, determine if the size of a minimum vertex cover of $G_1$ is smaller than or equal to the size of a minimum vertex cover of $G_2$ (`Minimum Vertex Cover Compare` problem).

There are even more natural $\mathrm{P}_{||}^{\mathrm{NP}}$-complete problems. As an example, see the beautiful paper by Hemaspaandra, Hemaspaandra, and Rothe [HHR97a]. In that paper, the authors show that the winner problem for a sophisticated voting scheme proposed by Lewis Dodgson more than 100 years ago is complete for $\mathrm{P}_{||}^{\mathrm{NP}}$. For more results in this line we recommend the survey paper by Hemaspaandra, Hemaspaandra, and Rothe [HHR97b].

In the next section we take a different approach for proving $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness, based on a characterization of $\mathrm{P}_{||}^{\mathrm{NP}}$ that can be derived from Krentel's characterization of $\mathrm{P}^{\mathrm{NP}}$-computations. Starting from this machine-based characterization, we derive satisfiability problems that are complete for $\mathrm{P}_{||}^{\mathrm{NP}}$. Of main interest are the satisfiability comparison problems. They allow an easy reduction to optimum comparison problems of several NP-complete optimization problems. In this way, we get an alternative proof for Wagner's result that the `Minimum Vertex Cover Compare` problem is $\mathrm{P}_{||}^{\mathrm{NP}}$-complete. We present our approach here because it is very simple and natural. It derives $\mathrm{P}_{||}^{\mathrm{NP}}$-complete satisfiability problems from a machine model in the same way as the NP-complete problem SAT in Cook's theorem [Coo71].

[1] We are only concerned with hardness relative to polynomial-time many-one reduction.

## 3.2 $\mathrm{P}_{\|}^{\mathrm{NP}}$-Complete Satisfiability Problems

In this section, we prove versions of the boolean satisfiability problem (see Definitions 3.2.5 and 3.2.12) complete for the complexity class $\mathrm{P}_{\|}^{\mathrm{NP}}$. We start with a short outline for the ideas behind of these results.

The class $\mathrm{P}_{\|}^{\mathrm{NP}}$ is defined by DPTMs that have oracle access to a set in NP. It is not immediately clear how to find complete problems for this class, since it is defined as "composition" of two complexity classes. What helps in this situation, is a characterization of the class by single NPTMs with a certain specific acceptance type. To that aim, we provide a characterization of $\mathrm{P}_{\|}^{\mathrm{NP}}$ in terms of NPTMs with output tape (Proposition 3.2.3). We get this characterization by a modification of an analogous characterization of $\mathrm{P}^{\mathrm{NP}}$ (Proposition 3.2.2), which goes back to Krentel [Kre88]. Such machine-based characterizations often lead to complete satisfiability problems in a straightforward way, because Turing machine computations can easily be encoded into boolean formulas. We get two such results, which are contained in Subsections 3.2.1 and 3.2.2.

In Subsection 3.2.1, we consider the following decision problem. We are given two 3-CNF formulas $F_1$ and $F_2$. Let max-$\mathbb{1}(F)$ denote the maximum number of variables set to true (the number of 1's) in satisfying assignments for 3-CNF formula $F$. We are interested in the problem of deciding if max-$\mathbb{1}(F_1) \leq$ max-$\mathbb{1}(F_2)$.[2] Theorem 3.2.6 establishes the $\mathrm{P}_{\|}^{\mathrm{NP}}$-completeness of this problem. Starting from the machine-based characterization of $\mathrm{P}_{\|}^{\mathrm{NP}}$ stated in Proposition 3.2.3, via the auxiliary result of Lemma 3.2.7, we first get a proof of Theorem 3.2.6 for circuits instead of formulas (Lemma 3.2.8). To complete the proof of Theorem 3.2.6, we apply the well-known technique of converting circuits into satisfiability equivalent 3-CNF formulas.

In Subsection 3.2.2, we are concerned with a related decision problem. Here, we are given only one 3-CNF formula $F$, and the problem consists of deciding if max-$\mathbb{1}(F)$ is odd or even. The $\mathrm{P}_{\|}^{\mathrm{NP}}$-completeness proof of this problem is similar to the proof of Theorem 3.2.6.

We make use of nondeterministic Turing machines with output tape. Each path of these machines is either accepting or rejecting, and additionally writes a string over $\Sigma^*$ on the output tape.

The following notation will be convenient.

**Definition 3.2.1** *We define the following functions:*

- *For every NPTM $N$ with output tape and $x \in \Sigma^*$, $\mathrm{out}_N(x, \rho)$ denotes the string that is output by $N(x)$ working on path $\rho$.*

[2]If one of $F_1$ or $F_2$ is not satisfiable, then max-$\mathbb{1}(F_1)$ or max-$\mathbb{1}(F_2)$ is undefined, and hence max-$\mathbb{1}(F_1) \leq$ max-$\mathbb{1}(F_2)$ is not true.

- *For every NPTM $N$ and input $x \in \Sigma^*$,*

$$\text{max-}\mathbb{1}(N,x) = \max\left\{ |\operatorname{out}_N(x,\rho)|_1 \mid \rho \textit{ is a path of } N(x)\right\}.^3$$

Proposition 3.2.2 goes back to Krentel [Kre88]. It gives a characterization of the complexity class $\mathrm{P}^{\mathrm{NP}}$ by nondeterministic polynomial-time bounded Turing machines with output tape.

**Proposition 3.2.2** *([Kre88]) A language $A \subseteq \Sigma^*$ is in $\mathrm{P}^{\mathrm{NP}}$ if and only if there is an NPTM $N$ with output tape such that the following statements are true for every $x \in \Sigma^*$:*

1. *For every path of $N(x)$, the output has the same length.*
2. *Every two paths $\rho_1$ and $\rho_2$ of $N(x)$ have the same acceptance behavior whenever they have the same output.*
3. *$x \in A$ if and only if $N$ accepts $x$ on $\rho_{\max}$, where $\rho_{\max}$ is any computation path with lexicographically maximum output.*

A modification gives a similar machine-based characterization of the class $\mathrm{P}^{\mathrm{NP}}_{\parallel}$.

**Proposition 3.2.3** *A language $A \subseteq \Sigma^*$ is in $\mathrm{P}^{\mathrm{NP}}_{\parallel}$ if and only if there is an NPTM $N$ with output tape such that the following statements are true for every $x \in \Sigma^*$:*

1. *Every two paths $\rho_1$ and $\rho_2$ of $N(x)$ have the same acceptance behavior whenever they have the same number of 1's in the output.*
2. *$x \in A$ if and only if $N$ accepts $x$ on $\rho_{\max}$, where $\rho_{\max}$ is a computation path with the maximum number of 1's in the output.*

**Proof** In the proof, we make use of the above mentioned fact that $\mathrm{P}^{\mathrm{NP}}_{\parallel} = \mathrm{P}^{\mathrm{NP[log]}}$.

"$\Longrightarrow$" Let $A \in \mathrm{P}^{\mathrm{NP[log]}}$ via oracle DPTM $M$ with oracle $C \in \mathrm{NP}$, where w.l.o.g. $M$ queries exactly $z(n) = O(\log n)$ strings to the oracle $C$ for every input of length $n$.

Since $C \in \mathrm{NP}$, there exist $D \in \mathrm{P}$ and a polynomial $r$ such that for every $x \in \Sigma^*$,

$$x \in C \Longleftrightarrow (\exists y : |y| \leq r(|x|))[\langle x, y\rangle \in D].$$

On input $x$, NPTM $N$ works as follows:

[3] Recall that for every $w \in \Sigma^*$, $|w|_1$ denotes the number of 1's in $w$.

*Step 1:* Nondeterministically guess $b_1, b_2, \dots, b_{z(|x|)} \in \{0, 1\}$.

*Step 2:* On each path $\rho$ with guessed $b_1, b_2, \dots, b_{z(|x|)}$, construct the oracle queries $q_1, q_2, \dots, q_{z(|x|)} \in \Sigma^*$ by simulation of $M^C(x)$, where the answers to the queries of $M$ to $C$ are substituted by $b_1, \dots, b_{z(|x|)}$. (Take "yes" as answer to the $i$th query if $b_i = 1$; take "no" as answer to the $i$th query if $b_i = 0$.)

*Step 3:* Successively, for each $i$ with $b_i = 1$, nondeterministically guess a string $y_i$ with $|y_i| \leq r(|q_i|)$. Verify that each $\langle q_i, y_i \rangle$ is in $D$. Output the string "0" and reject on the current path $\rho$ if at least one such test fails. Otherwise continue as follows on $\rho$.

*Step 4:* Output the string $w = 11 \dots 1$, where $|w| = \text{number}(b_1 \dots b_{z(|x|)}) + 1$. Since $z(n) = O(\log n)$, the length of $w$ is polynomially bounded in $|x|$.

*Step 5:* Accept on $\rho$ if and only if the computation of $M^C(x)$ simulated in Step 2 was accepting.

It is obvious that the machine $N$ thus described works in polynomial time.

Let $\rho_{\max}$ be any path reaching Step 4 with lexicographically maximum $b_1 \dots b_{z(|x|)}$ among all paths reaching Step 4. It is easy to see that

- The bits $b_1 \dots b_{z(|x|)}$ guessed on $\rho_{\max}$ represent the correct oracle answers to the queries made by $M^C$ on input $x$.

- The string $w$ output on $\rho_{\max}$ has maximum number of 1's in the output string among all paths of $N$, i.e., $|\text{out}_N(x, \rho_{\max})|_1 = \text{max-}\mathbb{1}(N, x)$.

Hence $M^C(x)$ accepts if and only if $N(x)$ accepts in Step 5 on path $\rho_{\max}$.

"$\Longleftarrow$" Let $N$ be an NPTM with output tape as in Proposition 3.2.3. Hence $x \in A$ if and only if $N$ accepts $x$ on $\rho_{\max}$, where $\rho_{\max}$ is a computation path with $\text{out}_N(x, \rho_{\max}) = \text{max-}\mathbb{1}(N, x)$. We have to show that $A \in \mathrm{P}^{\mathrm{NP[log]}}$.

Let $q$ be a polynomial that bounds the length of the ouputs of NPTM $N$. Our $\mathrm{P}^{\mathrm{NP[log]}}$-algorithm for deciding $A$ consists of two steps. First, we determine $\text{max-}\mathbb{1}(N, x)$ by binary search with $O(\log(q(|x|)))$ queries to the set $H_1$, where $H_1$ is defined by

$$H_1 = \{\langle x, k \rangle \mid N(x) \text{ has a path } \rho \text{ with } |\,\text{out}_N(x, \rho)|_1 \geq k\}.$$

Clearly, $H_1 \in \mathrm{NP}$. Second, we check whether $N(x)$ accepts on any (and hence on all) paths with $\text{max-}\mathbb{1}(N, x)$ 1's in the output. This can be done with one query to $H_2 \in \mathrm{NP}$, where

$$H_2 = \{\langle x, k \rangle \mid N(x) \text{ has an } \textit{accepting} \text{ path } \rho \text{ with } |\,\text{out}_N(x, \rho)|_1 = k\}.$$

We can replace the oracles $H_1$ and $H_2$ by a single oracle by taking the marked union $H = H_1 \oplus H_2$ of $H_1$ and $H_2$, which is also in NP. That yields a $\mathrm{P}_{\|}^{\mathrm{NP}}$-algorithm for $A$. ▌ (Proposition 3.2.3)

**Definition 3.2.4** *We define the following functions:* [4]

1. *For every $A \subseteq \Sigma^*$ and $x \in \Sigma^*$,*

$$\text{max-}\mathbb{1}(A,x) = \max\left\{|w|_1 \,\middle|\, \langle x,w\rangle \in A\right\};$$
$$\text{max-}\mathbb{1}\text{-set}(A,x) = \left\{w \,\middle|\, \langle x,w\rangle \in A \textit{ and } |w|_1 = \text{max-}\mathbb{1}(A,x)\right\}.$$

2. *For every boolean function (circuit or formula) $\phi$,*

$$\text{max-}\mathbb{1}(\phi) = \max\left\{|y|_1 \,\middle|\, \phi(y) = 1\right\};$$
$$\text{max-}\mathbb{1}\text{-set}(\phi) = \{y \mid \phi(y) = 1 \textit{ and } |y|_1 = \text{max-}\mathbb{1}(\phi)\}.^5$$

### 3.2.1 Satisfiability Comparison Problems

We define the problems that we prove $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete in this section.

**Definition 3.2.5** *We define the following satisfiability problems for boolean formulas:*

1. `MAX TRUE 3SAT COMPARE`

   *Instance: Two 3-CNF formulas $F_1$ and $F_2$.*
   *Question: Does it hold that* $\text{max-}\mathbb{1}(F_1) \le \text{max-}\mathbb{1}(F_2)$*?*

2. `MAX TRUE 3SAT EQUALITY`

   *Instance: Two 3-CNF formulas $F_1$ and $F_2$.*
   *Question: Does it hold that* $\text{max-}\mathbb{1}(F_1) = \text{max-}\mathbb{1}(F_2)$*?*

It is easy to show that `MAX TRUE 3SAT COMPARE` and `MAX TRUE 3SAT EQUALITY` are in $\mathrm{P}_{\|}^{\mathrm{NP}}$. We define the following auxiliary set $H$:

$$H = \{\langle F,k\rangle \mid F \text{ is a 3-CNF formula, and } \text{max-}\mathbb{1}(F) \ge k\}$$

[4]If the maximum is taken over the empty set, then the function is undefined.

[5]Recall that $\phi(y)$ stands for $\phi(y_1, y_2, \ldots, y_n)$, where $y_1 y_2 \ldots y_n = y$.

The set $H$ is clearly in NP. Let $F_1$ and $F_2$ be two 3-CNF formulas with no more than $n$ variables. We know that $\text{max-}\mathbb{1}(F_1) \leq n$ and $\text{max-}\mathbb{1}(F_2) \leq n$. We can in polynomial time in parallel query the tuples $\langle F_1, 0\rangle, \langle F_1, 1\rangle, \ldots, \langle F_1, n\rangle$ as well as $\langle F_2, 0\rangle, \langle F_2, 1\rangle, \ldots, \langle F_2, n\rangle$ to the NP-oracle $H$. With the answers to all these queries, we know $\text{max-}\mathbb{1}(F_i)$, $i \in \{1, 2\}$, since $\text{max-}\mathbb{1}(F_i)$ equals the largest $k$ such that $\langle F_i, k\rangle \in H$. It is then trivial to determine if $\text{max-}\mathbb{1}(F_1) \leq \text{max-}\mathbb{1}(F_2)$ or if $\text{max-}\mathbb{1}(F_1) = \text{max-}\mathbb{1}(F_2)$. Obviously this is a $\mathrm{P}_{\|}^{\mathrm{NP}}$-algorithm.

The next theorem establishes the $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness of `MAX TRUE 3SAT COMPARE` and `MAX TRUE 3SAT EQUALITY`. Note that the reduction has the additional property that it guarantees $\text{max-}\mathbb{1}(F_1) \geq \text{max-}\mathbb{1}(F_2)$. This will be useful in the next section.

**Theorem 3.2.6** *For every set $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$, there exists a polynomial-time computable function $f$ such that for each $x \in \Sigma^*$, $f(x) = \langle F_1, F_2\rangle$ is a pair of satisfiable 3-CNF formulas, and*

$$\begin{aligned} x \in A &\implies \text{max-}\mathbb{1}\text{-set}(F_1) = \text{max-}\mathbb{1}\text{-set}(F_2); \\ x \notin A &\implies \text{max-}\mathbb{1}(F_1) > \text{max-}\mathbb{1}(F_2). \end{aligned}$$

We are going to prove this theorem first for circuits instead of formulas, which is easier. As intermediate step, we need the following auxiliary lemma.

**Lemma 3.2.7** *For every set $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$, there are sets $B_1, B_2 \in \mathrm{P}$ and a polynomial $\tilde{p}$ such that for every $x \in \Sigma^*$*

1. $x \in A \implies \text{max-}\mathbb{1}(B_1, x) = \text{max-}\mathbb{1}(B_2, x)$;
   $x \notin A \implies \text{max-}\mathbb{1}(B_1, x) > \text{max-}\mathbb{1}(B_2, x)$.
2. $x \in A \implies \text{max-}\mathbb{1}\text{-set}(B_1, x) = \text{max-}\mathbb{1}\text{-set}(B_2, x)$.
3. $\{w \mid \langle x, w\rangle \in B_i\} \subseteq \Sigma^{\leq \tilde{p}(|x|)}$ *for every* $x \in \Sigma^*$ $(i \in \{1, 2\})$.

**Proof** Let $A$ be an arbitrary set in $\mathrm{P}_{\|}^{\mathrm{NP}}$. Let NPTM $N$ be as given by Proposition 3.2.3. We fix a function "code" that encodes computation paths as sets over $\Sigma^*$. Recall that we defined computation paths to be sequences of Turing machine configurations. The function "code" has to be one-to-one, and polynomial-time computable and invertible. Moreover, we require that for any given string "$w_1 \operatorname{code}(\rho) w_2$," the start and end of "$\operatorname{code}(\rho)$" can be identified in polynomial time.

Let $p$ be a polynomial bounding the length of $\operatorname{code}(\rho)$, i.e.,

$$p(n) \geq \max\{|\operatorname{code}(\rho)| \mid \rho \text{ is a computation path of } N(x) \text{ for some } x \in \Sigma^n\}$$

for every $n \in \mathbb{N}$. Let

$$B_1 =_{df} \{\langle x, w\rangle \mid x, w \in \Sigma^* \text{ and } w = 1\, \mathrm{out}_N(x, \rho)^{p(|x|)+1} \operatorname{code}(\rho)$$
$$\text{for some path } \rho \text{ of } N(x)\}\text{, and}$$

$$B_2 =_{df} \{\langle x, w\rangle \mid x, w \in \Sigma^* \text{ and } w = 1\,\mathrm{out}_N(x,\rho)^{p(|x|)+1}\,\mathrm{code}(\rho)$$
$$\text{for some } \textit{accepting} \text{ path } \rho \text{ of } N(x)\} \cup \{\langle x, 0\rangle\}.^{6}$$

Clearly, $B_1$ and $B_2$ are in P. Define $\tilde{p}$ to be a polynomial that is large enough such that for every $x, w \in \Sigma^*$ and $i \in \{1,2\}$,

$$\langle x, w\rangle \in B_i \Longrightarrow |w| \leq \tilde{p}(|x|).$$

Let $x \in A$. Then $N$ accepts $x$ on every path $\rho$ with $|\,\mathrm{out}_N(x,\rho)|_1 = \text{max-}\mathbb{1}(N,x)$. It is easy to see that this implies

$$\text{max-}\mathbb{1}\text{-set}(B_1, x) = \text{max-}\mathbb{1}\text{-set}(B_2, x).$$

Let $x \notin A$. Then $N$ rejects $x$ on every path $\rho$ with $|\,\mathrm{out}_N(x,\rho)|_1 = \text{max-}\mathbb{1}(N,x)$. Fix any path $\rho_{\max}$ with $|\,\mathrm{out}_N(x,\rho_{\max})|_1 = \text{max-}\mathbb{1}(N,x)$. Then for every accepting path $\rho_{\text{acc}}$ of $N(x)$,

$$|\,\mathrm{out}_N(x,\rho_{\max})|_1 > |\,\mathrm{out}_N(x,\rho_{\text{acc}})|_1.$$

Because $p(|x|)+1 > |\,\mathrm{code}(\rho)|$ is true for every path $\rho$ of $N(x)$, it follows that for every accepting path $\rho_{\text{acc}}$ of $N(x)$

$$|1\,\mathrm{out}_N(x,\rho_{\max})^{p(|x|)+1}\,\mathrm{code}(\rho_{\max})|_1 > |1\,\mathrm{out}_N(x,\rho_{\text{acc}})^{p(|x|)+1}\,\mathrm{code}(\rho_{\text{acc}})|_1. \quad (3.1)$$

To see that Equation 3.1 implies the desired inequality

$$\text{max-}\mathbb{1}(B_1, x) > \text{max-}\mathbb{1}(B_2, x),$$

note that

$$\text{max-}\mathbb{1}(B_1, x) \geq |1\,\mathrm{out}_N(x,\rho_{\max})^{p(|x|)+1}\,\mathrm{code}(\rho_{\max})|_1,$$

and

$$\text{max-}\mathbb{1}(B_2, x) =$$
$$\max(0, \max\{|1\,\mathrm{out}_N(x,\rho')^{p(|x|)+1}\,\mathrm{code}(\rho')|_1 \mid \rho' \text{ is an accepting path of } N(x)\}).$$

▌ (Lemma 3.2.7)

**Lemma 3.2.8** *For every set $A \in \mathrm{P}_{\parallel}^{\mathrm{NP}}$, there exists a polynomial-time computable function $f$ such that for each $x \in \Sigma^*$, $f(x) = \langle C_1, C_2\rangle$ is a pair of satisfiable boolean circuits, and*

$$\begin{aligned} x \in A &\Longrightarrow \text{max-}\mathbb{1}\text{-set}(C_1) = \text{max-}\mathbb{1}\text{-set}(C_2); \\ x \notin A &\Longrightarrow \text{max-}\mathbb{1}(C_1) > \text{max-}\mathbb{1}(C_2). \end{aligned}$$

[6] We put $\langle x, 0\rangle$ in $B_2$ in order to treat the case that $N(x)$ has no accepting paths at all.

**Proof** Let $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$. Let $B_1$ and $B_2$ be sets in P, and $\tilde{p}$ a polynomial belonging to $A$ as stated in Lemma 3.2.7. The proof essentially consists of transforming the sets $B_1$ and $B_2$ into boolean circuits $C_1$ and $C_2$. For technical reasons (because $C_1$ and $C_2$ will have a fixed number of inputs), we will work with the slightly modified sets $B_1'$ and $B_2'$ instead of $B_1$ and $B_2$:

$$B_i' = \{\langle x, w'\rangle \mid |w'| = \tilde{p}(|x|) \text{ and } (\exists w)[\langle x, w\rangle \in B_i \wedge w' = 0^*w]\} \quad (i \in \{1,2\}).$$

Let $M_{B_1'}$ and $M_{B_2'}$ be DPTMs that accept $B_1'$ and $B_2'$, respectively. It is well known (see, e.g., [ALR04, Pap94]) that DPTMs can be simulated by circuits of polynomial size. Let $x$ be an arbitrary string in $\Sigma^*$. We construct circuits $C_1$ and $C_2$ that simulate $M_{B_1'}$ and $M_{B_2'}$ on all inputs $\langle x, y\rangle$, where $y \in \Sigma^{\tilde{p}(|x|)}$, such that

$$\langle x, y\rangle \in B_i' \iff C_i(y) = 1 \qquad (i \in \{1,2\}).$$

Let $f(x) = \langle C_1, C_2\rangle$. It is easy to see that function $f$ has the properties stated in the lemma. ▮ (Lemma 3.2.8)

The following lemma formalizes the folklore result that every boolean circuit $C$ can be transformed in polynomial time into a boolean function $F$ in such a way that $C$ is satisfiable if and only $F$ is satisfiable.

**Lemma 3.2.9 (Folklore)** *There exists a polynomial-time computable function $f$ such that for every circuit $C(x_1, x_2, \ldots, x_n)$, $f(C)$ is a 3-CNF formula $G(x_1, x_2, \ldots, x_n, h_1, h_2, \ldots, h_m)$ with the following properties for all $x_1, x_2, \ldots x_n \in \{0,1\}$,*

- $C(x_1, x_2, \ldots, x_n) = 0 \Longrightarrow (\forall h_1 \cdots \forall h_m)[G(x_1, x_2, \ldots, x_n, h_1, \ldots, h_m)] = 0;$
- $C(x_1, x_2, \ldots, x_n) = 1 \Longrightarrow (\exists! h_1 \cdots \exists! h_m)[G(x_1, x_2, \ldots, x_n, h_1, \ldots, h_m)] = 1.$[7]

**Proof** As this is a standard result, we give only an outline of the idea. See, e.g., the textbooks [ALR04, CLRS01, Pap94] for more details. Suppose that $C$ has $n$ inputs and $m$ gates. The formula $G$ we construct has variables $x_1, x_2, \ldots, x_n$ representing the inputs of $C$, and variables $h_1, h_2, \ldots, h_m$, representing the values of gates $g_1, g_2, \ldots, g_m$ of $C$. For each gate $g_i$, we add a subformula $G^i$ to $G$. For example, if $g_i$ is an $\wedge$-gate that has incoming arcs from gates $g_j$ and $g_k$, then

$$G^i = h_i \leftrightarrow (h_j \wedge h_k),^{8} \tag{3.2}$$

which can be written equivalently with 3 clauses as

$$(\neg h_i \vee h_j) \wedge (\neg h_i \vee h_k) \wedge (\neg h_j \vee \neg h_k \vee h_i).$$

[7]We use $(\exists! h_1 \ldots \exists! h_m)[G(h_1, \ldots h_m)]$ to indicate that there is a unique assignment $h_1', \ldots, h_m' \in \{0,1\}$ for $G$ such that $G(h_1', \ldots h_m')$ is true.

[8]If $g_i$ has incoming arcs from input vertices, then take accordingly a subformula of the form $G^i = h_i \leftrightarrow (x_j \wedge x_k)$.

Finally, let

$$G = G^1 \wedge G^2 \wedge \ldots \wedge G^m \wedge h_m,$$

where $g_m$ is assumed to be the output gate of the circuit. The construction ensures that the following are true:

- If $C$ with input $x_1, x_2, \ldots, x_n$ evaluates to false, then there are no $h_1, h_2, \ldots, h_m$ making $G(x_1, x_2, \ldots, x_n, h_1, h_2, \ldots, h_m)$ true.
- If $C$ with input $x_1, x_2, \ldots, x_n$ evaluates to true, then there is a unique assignment $h_1, h_2, \ldots, h_m$ making $G(x_1, x_2, \ldots, x_n, h_1, h_2, \ldots, h_m)$ true. This assignment is uniquely determined by the values that the gates of $C$ get with input assignment $x_1, x_2, \ldots, x_n$: Variable $h_i$ gets value 1 if and only if the value of gate $g_i$ is 1. (This can be inductively proved using subformulas (3.2)). ▮ (Lemma 3.2.9)

We need the following slightly stronger result, because we have here to deal with a *pair* of circuits.

**Lemma 3.2.10** *There exists a polynomial-time computable function $f$ such that for every pair of boolean circuits $\langle C_1, C_2\rangle$, $f(C_1, C_2)$ is a pair of 3-CNF formulas $\langle G_1, G_2\rangle$ such that*

1. *$G_1$ and $G_2$ have the properties as in Lemma 3.2.9: For all $x_1, x_2, \ldots, x_n \in \{0,1\}$ and $i \in \{1,2\}$,*
   - $C_i(x_1, \ldots, x_n) = 0 \Longrightarrow (\forall h_1 \cdots \forall h_m)[G_i(x_1, \ldots, x_n, h_1, \ldots, h_m)] = 0;$
   - $C_i(x_1, \ldots, x_n) = 1 \Longrightarrow (\exists! h_1 \cdots \exists! h_m)[G_i(x_1, \ldots, x_n, h_1, \ldots, h_m)] = 1.$
2. *Additionally, for all $x_1, x_2, \ldots, x_n \in \{0,1\}$,*

$$C_1(x_1, x_2, \ldots, x_n) = 1 \wedge C_2(x_1, x_2, \ldots, x_n) = 1$$
$$\Longrightarrow$$
$$(\exists! h_1 \cdots \exists! h_m)[G_1(x_1, \ldots, x_n, h_1, \ldots, h_m) = 1 \wedge G_2(x_1, \ldots, x_n, h_1, \ldots, h_m) = 1].$$

**Proof** The proof uses the ideas of the previous lemma. A bit extra care is needed to treat the case that we have an assignment $x_1, x_2, \ldots, x_n$ that makes $C_1$ and $C_2$ true *simultaneously*: To make sure that $G_1(x_1, x_2, \ldots, x_n, h_1, h_2, \ldots, h_m)$ and $G_2(x_1, x_2, \ldots, x_n, h_1, h_2, \ldots, h_m)$ are satisfied *with the same* unique $h_1, h_2, \ldots, h_m$, we modify the construction in the proof of Lemma 3.2.9 slightly. Suppose that $C_1$ and $C_2$ have respectively $m_1$ and $m_2$ gates. Let $g_1, g_2, \ldots, g_m$ ($m = m_1 + m_2$) enumerate all gates occurring in $C_1$ or $C_2$. Let $G_1^1, G_1^2, \ldots, G_1^{m_1}$ and $G_2^1, G_2^2, \ldots, G_2^{m_2}$

be the subformulas obtained analogously to Eq. (3.2). The formulas $G_1$ and $G_2$ are then defined by

$$\begin{aligned} G_1 &= G_1^1 \wedge G_1^2 \wedge \ldots \wedge G_1^{m_1} \wedge G_2^1 \wedge G_2^2 \wedge \ldots \wedge G_2^{m_2} \wedge h_{m_1}; \\ G_2 &= G_1^1 \wedge G_1^2 \wedge \ldots \wedge G_1^{m_1} \wedge G_2^1 \wedge G_2^2 \wedge \ldots \wedge G_2^{m_2} \wedge h_{m_2}, \end{aligned}$$

where $g_{m_1}$ and $g_{m_2}$ are assumed to be the output gates of $C_1$ and $C_2$, respectively.

In summary, the simple idea is that we process the gates of *both* circuits in the construction of *both* formulas. Only the clauses that take the value from the output gate (clause $(h_{m_1})$ for $C_1$, clause $(h_{m_2})$ for $C_2$) are different. It is easy to see that this ensures that the second condition of the lemma is satisfied.

▮ (Lemma 3.2.10)

With the help of the previous lemma, we can now prove Theorem 3.2.6.

**Proof of Theorem 3.2.6** Because of Lemma 3.2.8, we only have to show that there is a polynomial-time computable function $f$ that transforms each pair $\langle C_1, C_2\rangle$ of satisfiable circuits to a pair of satisfiable 3-CNF formulas $\langle F_1, F_2\rangle$ such that

$$\begin{aligned} \text{max-}\mathbb{1}\text{-set}(C_1) = \text{max-}\mathbb{1}\text{-set}(C_2) &\implies \text{max-}\mathbb{1}\text{-set}(F_1) = \text{max-}\mathbb{1}\text{-set}(F_2); \quad (3.3) \\ \text{max-}\mathbb{1}(C_1) > \text{max-}\mathbb{1}(C_2) &\implies \text{max-}\mathbb{1}(F_1) > \text{max-}\mathbb{1}(F_2). \quad (3.4) \end{aligned}$$

Take the formulas $G_1$ and $G_2$ stated in Lemma 3.2.10. We define $F_i(x_1^1, \ldots, x_1^{m+1}, x_2^1, \ldots, x_2^{m+1}, \ldots, x_n^1, \ldots, x_n^{m+1}, h_1, \ldots, h_m)$ to be equivalent to $G_i$ ($i \in \{1,2\}$), but with each variable $x_k$ replicated $m+1$ times (by adding clauses equivalent to $x_k^1 \leftrightarrow x_k^j$ (i.e., clauses $(\neg x_k^1 \vee x_k^j)$ and $(x_k^1 \vee \neg x_k^j)$) for each $j \in \{2, 3, \ldots, m+1\}$).

It is easy to see that there exist integers $\delta_1, \delta_2 \in \{0, 1, \ldots, m\}$ such that,

$$\text{max-}\mathbb{1}(F_i) = (m+1)\,\text{max-}\mathbb{1}(C_i) + \delta_i \quad (i \in \{0,1\}).$$

Thus (3.4) is proved.

Statement (3.3) is also easy to prove: Let $C_1$ and $C_2$ be circuits with $\text{max-}\mathbb{1}\text{-set}(C_1) = \text{max-}\mathbb{1}\text{-set}(C_2)$. Because of the replication of variables in $F_1$ and $F_2$, assignments with maximum number of 1's within the $x$-variables in $F_i$ correspond to assignments with maximum number of 1's in $C_i$. For satisfying assignments to $F_1$ and $F_2$, the values of the $h$-variables are uniquely determined by the values of the $x$-variables. But we know that the satisfying assignments with maximum number of 1's for $C_1$ and $C_2$ are the same. Hence the assignments with maximum number of 1's are the same for $F_1$ and $F_2$, and (3.3) is proved.

▮ (Theorem 3.2.6)

**Corollary 3.2.11** `MAX TRUE 3SAT COMPARE` *and* `MAX TRUE 3SAT EQUALITY` *are complete in* $\mathrm{P}_{\|}^{\mathrm{NP}}$ *under polynomial-time many-one reductions.* ▮

### 3.2.2 Satisfiability Parity Problem

In this subsection we prove that the below defined problem `ODD MAX TRUE 3SAT` is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete. The proof is not presented in full detail, because it is similar to the one in Subsection 3.2.1.

**Definition 3.2.12** *We define the following satisfiability problem for boolean formulas:*

> `ODD MAX TRUE 3SAT`
>
> *Instance: A 3-CNF formula F.*
> *Question: Does it hold that* $\text{max-}\mathbb{1}(F) \equiv 1 \pmod 2$*?*

That this problem is contained in the class $\mathrm{P}_{\|}^{\mathrm{NP}}$ can be shown as demonstrated for `MAX TRUE 3SAT COMPARE` and `MAX TRUE 3SAT EQUALITY.` The next theorem establishes the $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness of `ODD MAX TRUE 3SAT`.

**Theorem 3.2.13** *For every set $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$, there exists a polynomial-time computable function $f$ such that for each $x \in \Sigma^*$, $f(x) = F$ is a satisfiable 3-CNF formula, and*

$$x \in A \iff \text{max-}\mathbb{1}(F) \equiv 1 \pmod 2. \tag{3.5}$$

We need the following auxiliary lemma.

**Lemma 3.2.14** *For every $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$ there are a set $B \in \mathrm{P}$ and a polynomial $\tilde{p}$ such that for every $x \in \Sigma^*$*

1. $x \in A \iff \text{max-}\mathbb{1}(B, x) \equiv 1(2)$, *and*
2. $\{w \mid \langle x, w\rangle \in B\} \subseteq \Sigma^{\leq \tilde{p}(|x|)}$ *for every* $x \in \Sigma^*$.

**Proof** Let $A$ be an arbitrary set in $\mathrm{P}_{\|}^{\mathrm{NP}}$. Let NPTM $N$ be as given by Proposition 3.2.3. As in the proof of Lemma 3.2.7, let $p$ be a polynomial bounding the length of $\text{code}(\rho)$. In place of two sets $B_1$ and $B_2$, we define now a single set $B$:

$$\begin{aligned} B =_{df} \big\{\langle x, w\rangle \,\big|\, & x, w \in \Sigma^* \text{ and } w = \text{out}_N(x,\rho)^{2p(|x|)+2}\,\text{code}(\rho)^2 0 \\ & \text{for some } \textit{rejecting} \text{ path } \rho \text{ of } N(x)\} \\ \cup \big\{\langle x, w\rangle \,\big|\, & x, w \in \Sigma^* \text{ and } w = \text{out}_N(x,\rho)^{2p(|x|)+2}\,\text{code}(\rho)^2 1 \\ & \text{for some } \textit{accepting} \text{ path } \rho \text{ of } N(x)\}\,. \end{aligned}$$

Clearly, $B \in \mathrm{P}$. Define $\tilde{p}$ to be a polynomial that is large enough such that for every $x, w \in \Sigma^*$,

$$\langle x, w\rangle \in B \implies |w| \leq \tilde{p}(|x|).$$

We prove the first property stated in the lemma. Let $x \in \Sigma^*$.

*Case 1:* $x \in A$.

Then $N(x)$ accepts on every path with maximum number of 1's in the output. Hence there is a path $\rho_{\max}$ with maximum number of 1's in the output[9] such that

$$\begin{aligned} \text{max-}\mathbb{1}(B,x) &= \max\{|y|_1 \mid \langle x,y\rangle \in B\} \\ &= \left|\text{out}_N\left(x,\rho_{\max}\right)^{2p(|x|)+2} \text{code}(\rho_{\max})^2 1\right|_1 . \end{aligned} \tag{3.6}$$

Hence $\text{max-}\mathbb{1}(B,x) \equiv 1(2)$.

*Case 2:* $x \notin A$.

Then $N(x)$ rejects on every path with maximum number of 1's in the output. Hence there is a path $\rho_{\max}$ with maximum number of 1's in the output such that

$$\begin{aligned} \text{max-}\mathbb{1}(B,x) &= \max\{|y|_1 \mid \langle x,y\rangle \in B\} \\ &= \left|\text{out}_N\left(x,\rho_{\max}\right)^{2p(|x|)+2} \text{code}(\rho_{\max})^2 0\right|_1 . \end{aligned} \tag{3.7}$$

Hence $\text{max-}\mathbb{1}(B,x) \equiv 0(2)$.

▮ (Lemma 3.2.14)

**Proof of Theorem 3.2.13** The proof is analogous to (and in fact easier than) the proof of Theorem 3.2.6: First, construct a circuit $C$ such that

$$x \in A \iff \text{max-}\mathbb{1}(C) \equiv 1 \pmod 2. \tag{3.8}$$

Second, construct a 3-CNF formula $G$ from $C$ as described in Lemma 3.2.9. Finally, obtain $F$ from $G$ by duplicating the variables $h_i$, so that the parity information of the variables $x_i$ is preserved. ▮ (Theorem 3.2.13)

**Corollary 3.2.15** `ODD MAX TRUE 3SAT` *is complete in* $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ *under polynomial-time many-one reduction.*

[9]It can be seen that this path $\rho_{\max}$ is a path with maximum number of 1's in the output that has additionally the property that $|\text{code}(\rho_{\max})|_1$ is maximum.

### 3.2.3 An Alternative Proof for Wagner's Lemma

Using Lemma 3.2.14, we can give an alternative proof of Wagner's lemma.

**Lemma 3.2.16 [Wagner 1987]**[10] *Let $D$ be an* NP*-complete set and let $E$ be an arbitrary set. If there exists a polynomial-time computable function $g$ such that*

$$\big\|\{i \mid x_i \in D\}\big\| \equiv 1 \pmod 2 \iff g(x_1, \ldots, x_{2k}) \in E \tag{3.9}$$

*for all $k \geq 1$, $x_1, \ldots, x_{2k} \in \Sigma^*$ with $\chi_D(x_1) \geq \chi_D(x_2) \geq \cdots \geq \chi_D(x_{2k})$, then $E$ is* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-hard.*

**Proof** Let $E$ be a set satisfying the conditions of Wagner's lemma. Let $A$ be an arbitrary set in $\mathrm{P}_{\|}^{\mathrm{NP}}$. We have to prove that there is a polynomial-time many-one reduction from $A$ to $E$.

Let $B$ be a set in P and $\tilde{p}$ be a polynomial that belong to $A$ according to Lemma 3.2.14. Let

$$\widehat{B} = \big\{\langle x, n\rangle \bigm| n \in \mathbb{N} \text{ and } (\exists y)\,[\langle x, y\rangle \in B \text{ and } |y|_1 \geq n]\big\}.$$

Clearly, $\widehat{B} \in$ NP. Because $D$ is NP-complete, there exists a polynomial-time many-one reduction $f_{\widehat{B}}$ from $\widehat{B}$ to the $D$. Using Lemma 3.2.14, we have

$$\begin{aligned}
x \in A &\iff \max\big\{|w|_1 \bigm| |w| \leq \tilde{p}(|x|) \text{ and } \langle x, w\rangle \in B\big\} \equiv 1 \pmod 2\\
&\iff \max\big\{|w|_1 \bigm| |w| \leq 2\tilde{p}(|x|) \text{ and } \langle x, w\rangle \in B\big\} \equiv 1 \pmod 2\\
&\iff \max\Big\{n \bigm| n \leq 2\tilde{p}(|x|) \text{ and } \langle x, n\rangle \in \widehat{B}\Big\} \equiv 1 \pmod 2\\
&\iff \Big\|\Big\{n \bigm| 1 \leq n \leq 2\tilde{p}(|x|) \text{ and } \langle x, n\rangle \in \widehat{B}\Big\}\Big\| \equiv 1 \pmod 2\\
&\iff \big\|\big\{n \bigm| 1 \leq n \leq 2\tilde{p}(|x|) \text{ and } f_{\widehat{B}}(\langle x, n\rangle) \in D\big\}\big\| \equiv 1 \pmod 2
\end{aligned}$$

From the definition of $\widehat{B}$, it is clear that

$$\chi_{\widehat{B}}(\langle x, 1\rangle) \geq \chi_{\widehat{B}}(\langle x, 2\rangle) \geq \ldots \geq \chi_{\widehat{B}}(\langle x, 2\tilde{p}(|x|)\rangle),$$

and hence

$$\chi_D(f_{\widehat{B}}(\langle x, 1\rangle)) \geq \chi_D(f_{\widehat{B}}(\langle x, 2\rangle)) \geq \cdots \geq \chi_D(f_{\widehat{B}}(\langle x, 2\tilde{p}(|x|)\rangle)).$$

By (3.9) of Wagner's lemma follows

$$x \in A \iff g\left(f_{\widehat{B}}(\langle x, 1\rangle), f_{\widehat{B}}(\langle x, 2\rangle), \ldots, f_{\widehat{B}}(\langle x, 2\tilde{p}(|x|)\rangle)\right) \in E$$

for a function $g \in$ FP. Therefore, the composition of function $f_{\widehat{B}}$ with function $g$ yields a polynomial-time reduction from $A$ to $E$. Hence $E$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard.

▌ (Lemma 3.2.16)

[10]Wagner states hardness for $\mathrm{P}_{bf}^{\mathrm{NP}}$, a class which is now known to be equal to $\mathrm{P}_{\|}^{\mathrm{NP}}$.

## 3.3 $\mathrm{P}_{\|}^{\mathrm{NP}}$-Completeness of the Minimum Vertex Cover Comparison Problem

In this section, we prove that it is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete to compare the sizes of minimum vertex covers for two given graphs. To this end, we give a reduction from the satisfiability compare problem defined in the previous section. We get this reduction by an easy modification of the classic reduction by Garey and Johnson [GJ79]. To make the exposition simpler, we make use of a weighted version of the vertex cover problem as an intermediate step.

A *vertex weighted graph* is a graph $G$ with a weight function $c : V(G) \to \mathbb{N}$ for its vertices. For any $V' \subseteq V(G)$, let $c(V') = \sum_{v \in V'} c(v)$. We define

$$mwvc(G) = \min\{c(V') \mid V' \text{ is a vertex cover of } G\}.$$

**Lemma 3.3.1** *There is a polynomial-time computable function $f$ such that for every satisfiable 3-CNF formula $F$, $f(F) = \langle G, k\rangle$, where $G$ is a vertex weighted graph and $k$ an integer, such that*

1. $mwvc(G) = k - \text{max-}\mathbb{1}(F)$.

2. *Every vertex $v$ has weight $c(v) \leq ||V||$.*

**Proof** We use the well-known reduction from `3SAT` to `Vertex Cover` given by Garey and Johnson [GJ79]. That reduction has the useful property that the structure of each solutions for the vertex cover problem reflect closely a satisfying assignment of the corresponding satisfiability problem.

Let $F$ be an arbitrary satisfiable 3-CNF formula. Let $F$ contain $n$ variables $u_1, u_2, \ldots, u_n$ and $m$ clauses $c_1, c_2, \ldots, c_m$. We can assume that $F$ has only clauses with *exactly* 3 literals per clause. If $F$ contains clauses with fewer than 3 literals, then we fill them up by replicating literals.[11]

Let $f(F)$ be the graph $G$ given by the reduction of Garey and Johnson [GJ79]. For the sake of self-containment we repeat their construction here.

The graph $G$ consists of two components (called "truth-setting" and "satisfaction testing") and edges communicating between them.

**Truth-setting components:** Each variable $u_i$ $(1 \leq i \leq n)$ is assigned a component $T_i = (V_i, E_i)$ with

$$\begin{aligned} V_i &=_{df} \{u_i, \neg u_i\} \\ E_i &=_{df} \{\{u_i, \neg u_i\}\}. \end{aligned}$$

[11]The 3-CNF formulas in [GJ79] are assumed to have exactly three literals per clause. The reduction works also if a literal occurs more than once in a clause.

**Satisfaction testing components:** Each clause $c_j$ $(1 \leq j \leq m)$ is assigned a component $S_j = (V'_j, E'_j)$ with three vertices forming a triangle:

$$\begin{aligned} V'_j &=_{df} \{a_1[j], a_2[j], a_3[j]\} \\ E'_j &=_{df} \{\{a_1[j], a_2[j]\}, \{a_1[j], a_3[j]\}, \{a_2[j], a_3[j]\}\} \end{aligned}$$

**Edges communicating between truth-setting and satisfaction testing components:** For each clause $c_j$ let

$$E''_j =_{df} \{\{a_1[j], x_j\}, \{a_2[j], y_j\}, \{a_3[j], z_j\}\}$$

where $x_j$, $y_j$, and $z_j$ denote the literals in $c_j$.

The graph $G$ is defined by

$$V =_{df} \left(\bigcup_{i=1}^{n} V_i\right) \cup \left(\bigcup_{j=1}^{m} V'_j\right)$$

and

$$E =_{df} \left(\bigcup_{i=1}^{n} E_i\right) \cup \left(\bigcup_{j=1}^{m} E'_j\right) \cup \left(\bigcup_{j=1}^{m} E''_j\right)$$

This ends the construction of $G$ given by [GJ79].

We define the weight function $w$ as follows. All vertices get weight $2m+n+1$, except the vertices corresponding to variables with assignment 0, which get weight $2m+n+2$:

$$\begin{aligned} c(a_1[j]) = c(a_2[j]) = c(a_3[j]) &= 2m+n+1 \quad (0 \leq j \leq m) \\ c(u_i) &= 2m+n+1 \\ c(\neg u_i) &= 2m+n+2 \quad (0 \leq i \leq n) \end{aligned}$$

**Claim 3.1** *Let $W$ be any minimum weight vertex cover of $G$. Then $W$ contains exactly $2m+n$ vertices: exactly two vertices from each satisfaction testing component $V'_j$, $(j = 1, 2, \ldots, m)$ and exactly one vertex from each truth-setting component $V_i$, $(i = 1, 2, \ldots, n)$.*

**Proof** Because $F$ is satisfiable, $G$ has a vertex cover with no more than $2m+n$ vertices. It is easy to see that every set of vertices with more than $2m+n$ vertices has larger weight than any set of vertices with exactly $2m+n$ vertices. On the other hand, every vertex cover $W$ of $G$ must necessarily contain at least two vertices from each $V'_j$ $(j = 1, 2, \ldots, m)$ and at least one vertex from each $V_i$ $(i = 1, 2, \ldots, n)$.

▮ (Claim 3.1)

We continue the proof of Lemma 3.3.1. It is easy to see that the following is true for any minimum weight vertex cover $W$ (see the explanations in [GJ79]):

$$\begin{aligned} \|\{v \in W \mid c(v) = 2m+n+2\}\| &= \min\{|w|_0 \mid F(w) = 1\} \\ &= n - \text{max-}\mathbb{1}(F). \end{aligned}$$

Therefore

$$\begin{aligned} c(W) &= 2m \cdot (2m+n+1) + \text{max-}\mathbb{1}(F) \cdot (2m+n+1) \\ &\quad + (n - \text{max-}\mathbb{1}(F)) \cdot (2m+n+2) \\ &= 2m(2m+n+1) + n(2m+n+1) + n - \text{max-}\mathbb{1}(F) \end{aligned}$$

Hence setting $k = (2m+n+1)(2m+n)+n$, we get the desired identity $mwvc(G) = k - \text{max-}\mathbb{1}(F)$. ▮ (Lemma 3.3.1)

Now it is easy to prove Lemma 3.3.1 also for unweighted graphs. Recall that $\tau(G)$ denotes the size of a minimum vertex cover of $G$.

**Lemma 3.3.2** *There is a polynomial-time computable function $g$ such that for every satisfiable 3-CNF formula $F$, $g(F) = \langle G, k\rangle$, where $G$ is a graph and $k$ an integer, such that $\tau(G) = k - \text{max-}\mathbb{1}(F)$.*

**Proof** Let $f(F) = \langle G', k\rangle$, where $f$ is the function stated in Lemma 3.3.1. Let $c$ be the weight function for $G$. Define $g(F) = \langle G, k\rangle$, where $G$ is defined as follows.

$$\begin{aligned} V(G) &= \bigcup_{u \in V(G')} \{u_1, \ldots, u_{c(u)}\}; \\ E(G) &= \left\{\{(u_i, v_j) \in V(G)\} \mid \{u,v\} \in E(G')\right\}. \end{aligned}$$

It is easy to see that $\tau(G) = mwvc(G')$. ▮ (Lemma 3.3.2)

**Definition 3.3.3** *We define the following decision problems:*

*1.* `Minimum Vertex Cover Compare`

*Instance: Two graphs $G_1$ and $G_2$.*
*Question: Does it hold that $\tau(G_1) \leq \tau(G_2)$?*

*2.* `Minimum Vertex Cover Equality`

*Instance: Two graphs $G_1$ and $G_2$.*
*Question: Does it hold that $\tau(G_1) = \tau(G_2)$?*

The next theorem gives the reduction establishing the $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness of `Minimum Vertex Cover Compare` and `Minimum Vertex Cover Equality`. The reduction has the additional property that we have never $\tau(G_1) < \tau(G_2)$. This property of the reduction will be needed in Chapter 6.

**Theorem 3.3.4** *For any set $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$, there exists a polynomial-time computable function $f$ such that for each $x \in \Sigma^*$, $f(x) = \langle G_1, G_2 \rangle$ is a pair graphs and*

$$\begin{aligned} x \in A &\implies \tau(G_1) = \tau(G_2); \\ x \notin A &\implies \tau(G_1) > \tau(G_2). \end{aligned}$$

**Proof** Let $A \in \mathrm{P}_{\|}^{\mathrm{NP}}$. Let $F_1$ and $F_2$ be the 3-CNF formulas stated in Theorem 3.2.6. Let $g(F_1) = \langle H_2, k_2 \rangle$ and $g(F_2) = \langle H_1, k_1 \rangle$, where $g$ is the reduction stated in Lemma 3.3.2. Suppose $k_1 \leq k_2$. Add $k_2 - k_1$ new isolated edges to $H_1$ and obtain $G_1$. Set $G_2 = H_2$. Then we get

$$\begin{aligned} \tau(G_1) &= k_2 - \text{max-}\mathbb{1}(F_2), \text{ and} \\ \tau(G_2) &= k_2 - \text{max-}\mathbb{1}(F_1). \end{aligned}$$

Hence

$$\begin{aligned} \text{max-}\mathbb{1}(F_1) &= \text{max-}\mathbb{1}(F_2) \implies \tau(G_1) = \tau(G_2); \\ \text{max-}\mathbb{1}(F_1) &> \text{max-}\mathbb{1}(F_2) \implies \tau(G_1) > \tau(G_2). \end{aligned}$$

The case of $k_2 < k_1$ is treated analogously.

▮ (Theorem 3.3.4)

Using binary search, it is easy to prove that `Minimum Vertex Cover Compare` and `Minimum Vertex Cover Equality` are in $\mathrm{P}_{\|}^{\mathrm{NP}}$.[12] Hence from Theorem 3.3.4 we get the following corollary.

**Corollary 3.3.5** `Minimum Vertex Cover Compare` *and* `Minimum Vertex Cover Equality` *are complete in $\mathrm{P}_{\|}^{\mathrm{NP}}$ under polynomial-time many-one reduction.* ▮

Theorem 3.3.4 and Corollary 3.3.5 can also be stated in terms of `Independent Set` and `Clique` by the following well-known facts:

- $V'$ is a vertex cover of a graph $G$ if and only if $V(G) - V'$ is an independent set of $G$.
- $V'$ is an independent set of a graph $G$ if and only if $V'$ is a clique of $G^c$, where $G^c$ is the complement graph of $G$.

[12] Here we use that $\mathrm{P}_{\|}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}[\log]}$.

## 3.4 Digression to Completeness for $\mathrm{P}^{\mathrm{NP}}$

In this section we use the ideas of Sections 3.2 and 3.3 to prove completeness results for the class $\mathrm{P}^{\mathrm{NP}}$.

**Definition 3.4.1** *We define the following functions:* [13]

1. *For every $A \subseteq \Sigma^*$ and $x \in \Sigma^*$,*

$$\text{max-lex}(A, x) = \max\left\{w \in \Sigma^* \mid \langle x, w\rangle \in A\right\}.$$

2. *For every boolean function (circuit or formula) $\phi$,*

$$\begin{aligned}\text{max-lex}(\phi) &= \max\left\{y_1y_2\dots y_n \in \Sigma^* \mid \phi(y_1, y_2, \dots, y_n) = 1\right\};\\ \text{min-lex}(\phi) &= \min\left\{y_1y_2\dots y_n \in \Sigma^* \mid \phi(y_1, y_2, \dots, y_n) = 1\right\}.\end{aligned}$$

**Definition 3.4.2** *We define the following satisfiability problems for boolean formulas:*

1. `MAX LEX 3SAT COMPARE`

   *Instance: Two 3-CNF formulas $F_1$ and $F_2$.*
   *Question: Is the lexicographically maximum satisfying truth assignment for $F_1$ less than or equal to that for $F_2$?*

2. `MAX LEX 3SAT EQUALITY`

   *Instance: Two 3-CNF formulas $F_1$ and $F_2$.*
   *Question: Does the lexicographically maximum satisfying truth assignment for $F_1$ equal the lexicographically maximum satisfying truth assignment for $F_2$?*

Using binary search, it is easy to prove that `MAX LEX 3SAT COMPARE` and `MAX LEX 3SAT EQUALITY` are in $\mathrm{P}^{\mathrm{NP}}$. The $\mathrm{P}^{\mathrm{NP}}$-hardness of these problems is established by the next theorem. Note that the reduction has the additional property that it guarantees $\text{max-lex}(F_1) \geq \text{max-lex}(F_2)$.

[13] If the maximum is taken over the empty set, then the function is undefined.

**Theorem 3.4.3** *For any set $A \in \mathrm{P}^{\mathrm{NP}}$, there exists a polynomial-time computable function $f$ such that for each $x \in \Sigma^*$, $f(x) = \langle F_1, F_2 \rangle$ is a pair of satisfiable 3-CNF formulas, and*

$$\begin{aligned} x \in A &\implies \text{max-lex}(F_1) = \text{max-lex}(F_2); \\ x \notin A &\implies \text{max-lex}(F_1) > \text{max-lex}(F_2). \end{aligned}$$

For the proof of Theorem 3.4.3, we need the following lemma.

**Lemma 3.4.4** *For any set $A \in \mathrm{P}^{\mathrm{NP}}$, there are sets $B_1, B_2 \in \mathrm{P}$ and a polynomial $\tilde{p}$ such that*

1. $x \in A \implies \text{max-lex}(B_1, x) = \text{max-lex}(B_2, x)$;
   $x \notin A \implies \text{max-lex}(B_1, x) > \text{max-lex}(B_2, x)$.
2. $\{w \mid \langle x, w \rangle \in B_i\} \subseteq \Sigma^{\tilde{p}(|x|)}$ *for every* $x \in \Sigma^*$ $(i \in \{1, 2\})$.

**Proof** Let $A$ be an arbitrary set in $\mathrm{P}^{\mathrm{NP}}$. Let NPTM $N$ be as given by Proposition 3.2.2.

Let

$$\begin{aligned} B_1' =_{df} \{\langle x, w \rangle \mid\ & x, w \in \Sigma^* \text{ and } w = 1\,\text{out}_N(x, \rho)\,\text{code}(\rho) \\ & \text{for some path } \rho \text{ of } N(x)\}, \text{ and} \end{aligned}$$

$$\begin{aligned} B_2' =_{df} \{\langle x, w \rangle \mid\ & x, w \in \Sigma^* \text{ and } w = 1\,\text{out}_N(x, \rho)\,\text{code}(\rho) \\ & \text{for some } \textit{accepting} \text{ path } \rho \text{ of } N(x)\} \cup \{\langle x, 0 \rangle\}.^{14} \end{aligned}$$

Let $\tilde{p}$ be a polynomial that is large enough such that for every $x, w \in \Sigma^*$ and $i \in \{1, 2\}$,

$$\langle x, w \rangle \in B_i' \implies |w| \le \tilde{p}(|x|).$$

Clearly, $B_1'$ and $B_2'$ are in P. Define $B_1$ and $B_2$ to be the same as $B_1'$ and $B_2'$, but with the second string of each pair $\langle x, w \rangle$ filled up with zeroes as follows:

$$B_i = \{\langle x, w' \rangle \mid |w'| = \tilde{p}(|x|) \text{ and } (\exists w)[\langle x, w \rangle \in B_i \text{ and } w' = w0^*]\} \;\; (i \in \{1, 2\}).$$

It is easy to see that $B_1$ and $B_2$ satisfy the properties stated in the lemma.

▮ (Lemma 3.4.4)

**Proof of Theorem 3.4.3.** The proof is analogous to (and in fact easier than) the proof of Theorem 3.2.6: First, construct two circuit $C_1$ and $C_2$ such that

$$\begin{aligned} x \in A &\implies \text{max-lex}(C_1) = \text{max-lex}(C_2); \\ x \notin A &\implies \text{max-lex}(C_1) > \text{max-lex}(C_2). \end{aligned}$$

[14] We put $\langle x, 0 \rangle$ in $B_2'$ in order to treat the case that $N(x)$ has no accepting paths at all.

Second, construct 3-CNF formulas $F_1$ and $F_2$ from $C_1$ and $C_2$ as described in Lemma 3.2.10. For the proof of this theorem, no replication of variables $x_i$ is necessary, because the lexicographic ordering ensures that the values of the auxiliary variables $h_i$ are less significant than the values of the variables $x_i$.

∎ (Theorem 3.4.3)

**Corollary 3.4.5** `MAX LEX 3SAT COMPARE` *and* `MAX LEX 3SAT EQUALITY` *are complete in* $P^{NP}$ *under polynomial-time many-one reduction.* ∎

The `Traveling Salesperson` Problem (`TSP`) can be described as follows. We are given $k$ cities, and a $k \times k$ matrix $M$ with nonnegative integer entries that give the distances between the cities. We are interested in the length of a shortest tour, min-tour$(M)$, that visits each city exactly once and returns to the start. More formally,

$$\text{min-tour}(M) = \min \left\{ \sum_{i=1}^{k-1} M[\pi(i), \pi(i+1)] + M[\pi(k), \pi(1)] \,\middle|\, \pi \text{ is a permutation on } [k] \right\}.[15]$$

We define the classical NP-complete decision version of the `Traveling Salesperson` Problem.

**Definition 3.4.6** *(Traveling Salesperson Problem)*

*Traveling Salesperson Problem*

*Instance: A matrix* $M \in \mathbb{N}_{k\times k}$*; an integer* $b$.
*Question: Does it hold that* min-tour$(M) \le b$*?*

We now apply Theorem 3.4.3 for proving the below defined versions of `TSP` to be $P^{NP}$-complete.

**Definition 3.4.7** *We define the following decision problems:*

*1.* `TSP Compare`

*Instance: Two matrices* $M_1, M_2 \in \mathbb{N}_{k\times k}$.
*Question: Does it hold that* min-tour$(M_1) \le$ min-tour$(M_2)$*?*

[15]Recall that $[k]$ stands for $\{1, 2, \ldots, k\}$.

2. `TSP Equality`

   *Instance: Two matrices* $M_1, M_2 \in \mathbb{N}_{k \times k}$.
   *Question: Does it hold that* $\text{min-tour}(M_1) = \text{min-tour}(M_2)$*?*

An *edge weighted graph* is a graph $G$ with a cost function $c : E(G) \to \mathbb{N}$ for its edges. Given a Hamilton cycle $C$ in an edge weighted directed or undirected graph, the cost of $C$ is the sum of the costs of the edges in $C$.

**Definition 3.4.8** *We define the following functions:*

1. *For every directed graph $G$ with weight function $c$,* $\text{min-directed-circuit}(G)$ *denotes the cost of a minimum cost Hamilton cycle in $G$. If $G$ has no Hamilton cycle then* $\text{min-directed-circuit}(G)$ *is undefined.*

2. *For every undirected graph $G$ with weight function $c$,* $\text{min-undirected-circuit}(G)$ *denotes the cost of a minimum cost Hamilton cycle in $G$. If $G$ has no Hamilton cycle then* $\text{min-undirected-circuit}(G)$ *is undefined.*

**Lemma 3.4.9** *There is a polynomial-time computable function $f$ such that for every satisfiable 3-CNF formula $F$, $f(F) = G$, where $G$ is an edge weighted directed graph such that* $\text{min-directed-circuit}(G) = \text{pos}(\text{min-lex}(F))$. [16]

**Proof** We use the reduction from `3SAT` to `Directed Hamilton Cycle` that can be found in the textbook by Schöning [Sch01]. This reduction has (once again) the useful property that the structure of any solution for the Directed Hamilton Cycle problem reflects closely a corresponding satisfying assignment of the satisfiability problem we are reducing from.

Let $F$ be an arbitrary 3-CNF formula with $n$ variables $u_1, u_2, \ldots, u_n$ and $m$ clauses $c_1, c_2, \ldots, c_m$. We can assume that all clauses in $F$ have *exactly* 3 literals. If $F$ contains clauses with fewer than 3 literals, then we fill them up by replicating literals. The reduction also works if a literal occurs more than once in a clause. We refer to the reader to [Sch01] for the reduction from `3SAT` to `Directed Hamilton Cycle`. We do not repeat the reduction here. Let $f(F)$ be the graph $G$ obtained by this reduction. For each variable $u_i$, there is a corresponding vertex $i$ (notation of [Sch01]) in $G$. Each such vertex $i$ has exactly two outgoing arcs: one arc corresponding to occurrences of $u_i$ in $F$, and another arc corresponding to occurrences of $\neg u_i$ in $F$. We set the weight of the arc leaving vertex $i$ that corresponds to occurrences of $u_i$ in $F$ to $2^{n-i}$. All other arcs get weight 0. ▮ (Lemma 3.4.9)

[16] Recall that $\text{pos}(x)$ denotes the lexicographic rank of $x$ among the strings of length $|x|$.

The transition from undirected to directed graphs is easy.

**Lemma 3.4.10** *There is a polynomial-time computable function $f$ such that for every satisfiable 3-CNF formula $F$, $f(F) = G$, where $G$ is an edge weighted undirected graph such that* min-undirected-circuit$(G) = \text{pos}(\text{min-lex}(F))$.

**Proof** Let $H$ be the directed graph obtained according to Lemma 3.4.9. Obtain $f(F)$ from $H$, as in [Sch01], by expanding each vertex of $H$ to three vertices: one vertex for incoming edges, another vertex for outgoing edges, and a third (middle) node to force us to go from ingoing edges to outgoing edges. The weights of the edges are chosen as in Lemma 3.4.9. The correctness of this construction can easily be seen from the explanations given in [Sch01]. ▌ (Lemma 3.4.10)

The same lemma, but with max-lex instead of min-lex is easily obtained.

**Lemma 3.4.11** *There is a polynomial-time computable function $g$ such that for every satisfiable 3-CNF formula $F$ with $n$ variables, $g(F) = G$, where $G$ is an edge weighted undirected graph such that* min-undirected-circuit$(G) = 2^n - 1 - \text{pos}(\text{max-lex}(F))$.

**Proof** Define $F'$ to be the same as $F$, but with each literal negated. It is easy to see that $\text{pos}(\text{min-lex}(F')) = 2^n - 1 - \text{pos}(\text{max-lex}(F))$.

Define $g(F) = f(F')$, where $f$ is the reduction of the previous lemma. Then we have min-undirected-circuit$(G)$ = min-undirected-circuit$(g(F))$ = min-undirected-circuit$(f(F')) = \text{pos}(\text{min-lex}(F')) = 2^n - 1 - \text{pos}(\text{max-lex}(F))$.

▌ (Lemma 3.4.11)

**Lemma 3.4.12** *There is a polynomial-time computable function $f$ such that for every satisfiable 3-CNF formula $F$ with $n$ variables, $f(F) = M$, where $M \in \mathbb{N}_{k\times k}$ for some integer $k$, such that* min-tour$(M) = 2^n - 1 - \text{pos}(\text{max-lex}(F))$.

**Proof** Let $F$ be an arbitrary satisfiable 3-CNF formula. Let $G = g(F)$ be an edge weighted undirected graph with cost function $c$, where $g$ is the reduction of the previous lemma. Let $V(G) = \{v_1, v_2, \ldots, v_k\}$. It is sufficient to define a matrix $M$ such that min-tour$(M)$ = min-undirected-circuit$(G)$. From the construction of $G$ it is clear that min-undirected-circuit$(G) \leq \sum_{i=1}^{n} 2^{n-i} = 2^n - 1$. Hence adding arcs with weight $2^n$ to $G$ does not change min-undirected-circuit$(G)$. Therefore, we can define $M$ by

$$M[i,j] =_{df} \begin{cases} c(\{v_i, v_j\}) & \text{if } \{v_i, v_j\} \in E(G) \\ 2^n & \text{otherwise.} \end{cases}$$

Clearly, min-tour$(M)$ = min-undirected-circuit$(G)$. ▌ (Lemma 3.4.12)

**Theorem 3.4.13** *For any set $A \in \mathrm{P}^{\mathrm{NP}}$, there exists a polynomial-time computable function $h$ such that for each $x \in \Sigma^*$, $h(x) = \langle M_1, M_2 \rangle$, and*

$$\begin{aligned} x \in A &\implies \text{min-tour}(M_1) = \text{min-tour}(M_2); \\ x \notin A &\implies \text{min-tour}(M_1) > \text{min-tour}(M_2). \end{aligned}$$

**Proof** Let $A \in \mathrm{P}^{\mathrm{NP}}$ and $x \in A$. Let the 3-CNF formulas $F_1$ and $F_2$ be determined by $\langle F_1, F_2 \rangle = f(x)$, where $f$ is the function stated in Theorem 3.4.3. W.l.o.g. we can assume that $F_1$ and $F_2$ have the same number of variables. Define $M_1$ and $M_2$ by $M_1 = g(F_2)$ and $M_2 = g(F_1)$, where $g$ is the function stated in Lemma 3.4.12. Clearly,

$$\begin{aligned} \text{max-lex}(F_2) = \text{max-lex}(F_1) &\implies \text{min-tour}(M_1) = \text{min-tour}(M_2); \\ \text{max-lex}(F_2) > \text{max-lex}(F_1) &\implies \text{min-tour}(M_1) > \text{min-tour}(M_2). \end{aligned}$$

Let $h(x) = \langle M_1, M_2 \rangle$. ▮ (Theorem 3.4.13)

Using binary search, it is easy to prove that `TSP Compare` and `TSP Equality` are in $\mathrm{P}^{\mathrm{NP}}$. Hence from Theorem 3.4.13 we get the following corollary.

**Corollary 3.4.14** `TSP Compare` *and* `TSP Equality` *are $\mathrm{P}^{\mathrm{NP}}$-complete.* ▮

# Chapter 4

# Exact Complexity of the Winner Problem for Young Elections

## 4.1 Introduction

More than a decade ago, Bartholdi, Tovey, and Trick [BTT89b, BTT89a, BTT92] initiated the study of electoral systems with respect to their computational properties. In particular, they proved NP hardness lower bounds [BTT89b] for determining the winner in the voting schemes proposed by Dodgson and by Kemeny, and they studied complexity issues related to the problem of manipulating elections by strategic voting [BTT89a, BTT92]. Hemaspaandra, Hemaspaandra, and Rothe [HHR97a] classified both the winner and the ranking problem for Dodgson elections by proving them complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$.

We study complexity issues related to Young and Dodgson elections. In 1977, Young [You77] proposed a voting scheme that extends the Condorcet Principle based on the fewest possible number of voters whose removal makes a given candidate $c$ the Condorcet winner, i.e., $c$ defeats all other candidates by a strict majority of the votes. We prove that both the winner and the ranking problem for Young elections is complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$. To this end, we give a reduction from the problem `Maximum Set Packing Compare`, which we also prove $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.

In Section 4.3, we study a homogeneous variant of Dodgson elections that was introduced by Fishburn [Fis77]. In contrast to the above-mentioned result of Hemaspaandra et al. [HHR97a], we show that both the winner and the ranking problem for Fishburn's homogeneous Dodgson elections can be solved efficiently by a linear program that is based on an integer linear program of Bartholdi et al. [BTT89b].

## 4.2 Complexity of the Winner Problem for Young Elections

### 4.2.1 Some Background from Social Choice Theory

We first give some background from social choice theory. Let $C$ be the set of all candidates (or alternatives). We assume that each voter has strict preferences over the candidates. Formally, the preference order of each voter is antisymmetric, transitive, and complete (i.e., all candidates are ranked by each voter). An election is given by a *preference profile*, a pair $\langle C, V\rangle$ such that $C$ is a set of candidates and $V$ is the multiset of the voters' preference orders on $C$. Note that distinct voters may have the same preferences over the candidates. A *voting scheme* (or *social choice function*, SCF for short) is a rule for how to determine the winner(s) of an election; i.e., an SCF maps any given preference profile to society's aggregate *choice set*, the set of candidates who have won the election. For any SCF $f$ and any preference profile $\langle C, V\rangle$, $f(\langle C, V\rangle)$ denotes the set of winning candidates. For example, the *majority rule* says that a candidate $a$ *defeats* a candidate $b$ if and only if $a$ is preferred to $b$ by a strict majority of the voters. According to the majority rule, an election is won by a candidate who defeats every other candidate. Such a candidate is called the *Condorcet winner*.

In 1785, Marie-Jean-Antoine-Nicolas de Caritat, the Marquis de Condorcet, noted in his seminal essay [Con85] that whenever there are at least three candidates, say $a$, $b$, and $c$, the majority rule may yield cycles. His example consists of the following three voters:

$$\begin{array}{ccccc} a & > & b & > & c, \\ b & > & c & > & a, \\ c & > & a & > & b. \end{array}$$

Thus, $a$ defeats $b$ and $b$ defeats $c$, and yet $c$ defeats $a$. That is, even though each individual voter has a rational (i.e., transitive or non-cyclic) preference order, society may behave irrationally and Condorcet winners do not always exist. This observation is known as the Condorcet Paradox. The *Condorcet principle* says that for each preference profile, the winner of the election is to be determined by the majority rule. An SCF is said to be a *Condorcet SCF* if and only if it respects the Condorcet principle in the sense that the Condorcet winner is elected whenever one exists. Note that Condorcet winners are uniquely determined if they exist.

Many Condorcet SCFs have been proposed in the social choice literature; for an overview of the most central ones, we refer to the work of Fishburn [Fis77]. They extend the Condorcet principle in a way that avoids the troubling feature of the majority rule. In this chapter, we will focus on only two such Condorcet SCFs, the Dodgson voting scheme [Dod76] and the Young voting scheme [You77]. In the next chapter, we will be concerned with Kemeny's voting scheme.

In 1876, Charles L. Dodgson proposed a voting scheme [Dod76] that suggests that we remain most faithful to the Condorcet principle if the election is won by any candidate who is "closest" to being a Condorcet winner. To define "closeness," each candidate $c$ in a given election $\langle C, V\rangle$ is assigned a score, denoted DodgsonScore$(C, c, V)$, which is the smallest number of sequential interchanges of adjacent candidates in the voters' preferences that are needed to make $c$ a Condorcet winner. Here, one interchange means that, in (any) one of the voters, two adjacent candidates are switched. A *Dodgson winner* is any candidate with minimum Dodgson score. Using Dodgson scores, one can also tell who of two given candidates is ranked better according to the Dodgson SCF.

Young's approach to extending the Condorcet principle is reminiscent of Dodgson's approach in that it is also based on altered profiles. Unlike Dodgson, however, Young [You77] suggests that we remain most faithful to the Condorcet principle if the election is won by any candidate who is made a Condorcet winner by *removing the fewest possible number of voters*, instead of doing the fewest possible number of switches in the voters' preferences. For each candidate $c$ in a given preference profile $\langle C, V\rangle$, define YoungScore$(C, c, V)$ to be the size of a largest submultiset of $V$ for which $c$ is a Condorcet winner.[1] A *Young winner* is any candidate with a maximum Young score.

Here is an example. We have the following voter profile $\langle C, V\rangle$ with candidate set $C = \{a, b, c, d\}$ and the multiset of preference rankings $V$ given by

$$\begin{array}{ccccccc} a & > & b & > & c & > & d, \\ a & > & b & > & c & > & d, \\ d & > & a & > & b & > & c, \\ b & > & c & > & d & > & a. \end{array}$$

One can verify that YoungScore$(C, a, V) = 3$: Candidate $a$ is not a Condorcet winner in the given profile, but $a$ is Condorcet winner for the profile that we obtain if we delete the last preference order. Candidate $b$ is Condorcet winner in the profile given by the submultiset of $V$ that contains only the preference order $b > c > d > d$, and it can be verified that there is no larger submultiset of $V$ for which $b$ is a Condorcet winner. Hence YoungScore$(C, b, V) = 1$. Analogously, we obtain YoungScore$(C, d, V) = 1$. Finally, candidate $c$ is not Condorcet winner for any profile that is a submultiset of $V$. Hence YoungScore$(C, c, V) = 0$.

We see that $a$ is the candidate with largest Young score. Therefore $a$ is the winner of the election according to Young's voting scheme.

Homogeneous variants of Dodgson's and Young's voting schemes will be defined in Section 4.3.

[1] We define YoungScore$(C, c, V)$ to be zero in case there does not exist any submultiset of $V$ for which $c$ is a Condorcet winner.

### 4.2.2 Complexity Issues Related to Voting Schemes

To study computational complexity issues related to Dodgson's voting scheme, Bartholdi, Tovey, and Trick defined the decision problems `Dodgson Winner` and `Dodgson Ranking`.

**Definition 4.2.1 ([BTT89b])** *We define the following decision problems:*

1. `Dodgson Winner`

   *Instance: A preference profile $\langle C, V\rangle$ and a designated candidate $c \in C$.*
   *Question: Is $c$ a Dodgson winner of the election? That is, is it true that for all $d \in C$,* $\mathrm{DodgsonScore}(C, c, V) \leq \mathrm{DodgsonScore}(C, d, V)$*?*

2. `Dodgson Ranking`

   *Instance: A preference profile $\langle C, V\rangle$ and two designated candidates $c, d \in C$.*
   *Question: Does $c$ tie-or-defeat $d$ in the election? That is, is it true that* $\mathrm{DodgsonScore}(C, c, V) \leq \mathrm{DodgsonScore}(C, d, V)$*?*

Bartholdi et al. [BTT89b] established an NP-hardness lower bound for both these problems. Their result was optimally improved by Hemaspaandra, Hemaspaandra, and Rothe [HHR97a] who proved that `Dodgson Winner` and `Dodgson Ranking` are complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$, the class of problems solvable in polynomial time with parallel (i.e., truth-table) access to an NP oracle.

As above, we define the corresponding decision problems for Young elections.

**Definition 4.2.2** *We define the following decision problems:*

1. `Young Winner`

   *Instance: A preference profile $\langle C, V\rangle$ and a designated candidate $c \in C$.*
   *Question: Is $c$ a Young winner of the election? That is, is it true that for all $d \in C$,* $\mathrm{YoungScore}(C, c, V) \geq \mathrm{YoungScore}(C, d, V)$*?*

2. `Young Ranking`

   *Instance: A preference profile $\langle C, V\rangle$ and two designated candidates $c, d \in C$.*
   *Question: Does $c$ tie-or-defeat $d$ in the election? That is, is it true that* $\mathrm{YoungScore}(C, c, V) \geq \mathrm{YoungScore}(C, d, V)$*?*

### 4.2.3 Hardness of Determining Young Winners

The main result in this section is that the problems `Young Winner` and `Young Ranking` are complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$. In Theorem 4.2.7 below, we give a reduction from the problem `Maximum Set Packing Compare` that is defined below. For a given family $\mathcal{S}$ of sets, let $\kappa(\mathcal{S})$ be the maximum number of pairwise disjoint sets in $\mathcal{S}$.

**Definition 4.2.3** *We define the following decision problem:*

> `Maximum Set Packing Compare`
>
> *Instance: Two finite sets $B_1$ and $B_2$, and two families $\mathcal{S}_1$ and $\mathcal{S}_2$ of subsets of $B_1$ and $B_2$, respectively.*
> *Question: Does it hold that $\kappa(\mathcal{S}_1) \geq \kappa(\mathcal{S}_2)$?*

To prove that `Maximum Set Packing Compare` is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete, we give a reduction from the problem `Independence Number Compare`.

**Definition 4.2.4** *We define the following decision problem:*

> `Independence Number Compare`
>
> *Instance: Two graphs $G_1$ and $G_2$.*
> *Question: Does it hold that $\alpha(G_1) \geq \alpha(G_2)$?*

As remarked in Section 3.3, Corollary 3.3.5 immediately implies the following proposition.

**Proposition 4.2.5** `Independence Number Compare` *is* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete.*

**Lemma 4.2.6** `Maximum Set Packing Compare` *is* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete.*

**Proof** We give a polynomial-time many-one reduction from the problem `Independence Number Compare` to the problem `Maximum Set Packing Compare`. Let $G_1$ and $G_2$ be two given graphs. For $i \in \{1, 2\}$, define $B_i$ to be the union of the set of edges of $G_i$ and the set of vertices of $G_i$, and define $\mathcal{S}_i$ so as to contain the following sets: For each vertex $v$ of $G_i$, add to $\mathcal{S}_i$ the set of edges incident to $v$ and the vertex $v$ itself. Thus, for each $i \in \{1, 2\}$, we have $\alpha(G_i) = \kappa(\mathcal{S}_i)$, which proves the lemma. ▮ (Lemma 4.2.6)

Now, we prove the main result of this section.

**Theorem 4.2.7** `Young Ranking` *is* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete.*

**Proof** It is easy to see that `Young Ranking` and `Young Winner` are in $\mathrm{P}_{\|}^{\mathrm{NP}}$. To prove the $\mathrm{P}_{\|}^{\mathrm{NP}}$ lower bound, we give a polynomial-time many-one reduction from the problem `Maximum Set Packing Compare`. Let $B_1 = \{x_1, x_2, \ldots, x_m\}$ and $B_2 = \{y_1, y_2, \ldots, y_n\}$ be two given sets, and let $\mathcal{S}_1$ and $\mathcal{S}_2$ be given families of subsets of $B_1$ and $B_2$, respectively. Recall that $\kappa(\mathcal{S}_i)$, for $i \in \{1, 2\}$, is the maximum number of pairwise disjoint sets in $\mathcal{S}_i$; w.l.o.g., we may assume that $\kappa(\mathcal{S}_i) > 2$.

We define a preference profile $\langle C, V\rangle$ such that $c$ and $d$ are designated candidates in $C$, and it holds that:

$$\begin{aligned}\text{YoungScore}(C,c,V) &= 2\cdot\kappa(\mathcal{S}_1)+1; && (4.1)\\ \text{YoungScore}(C,d,V) &= 2\cdot\kappa(\mathcal{S}_2)+1. && (4.2)\end{aligned}$$

Define the set $C$ of candidates as follows:

- create the two designated candidates $c$ and $d$;
- for each element $x_i$ of $B_1$, create a candidate $x_i$;
- for each element $y_i$ of $B_2$, create a candidate $y_i$;
- create two auxiliary candidates, $a$ and $b$.

Define the set $V$ of voters as follows:

- **Voters representing $\boldsymbol{\mathcal{S}_1}$:** For each set $E \in \mathcal{S}_1$, create a single voter $v_E$ as follows:
  - Enumerate $E$ as $\{e_1, e_2, \ldots, e_{\|E\|}\}$ (renaming the candidates $e_i$ chosen from $\{x_1, x_2, \ldots, x_m\}$ for notational convenience), and enumerate its complement $\overline{E} = B_1 - E$ as $\{\overline{e}_1, \overline{e}_2, \ldots, \overline{e}_{m-\|E\|}\}$.
  - To make the preference orders easier to parse, we use

    "$\overrightarrow{E}$" to represent the text string "$e_1 > e_2 > \cdots > e_{\|E\|}$";
    "$\overrightarrow{\overline{E}}$" to represent the text string "$\overline{e}_1 > \overline{e}_2 > \cdots > \overline{e}_{m-\|E\|}$";
    "$\overrightarrow{B_1}$" to represent the text string "$x_1 > x_2 > \cdots > x_m$";
    "$\overrightarrow{B_2}$" to represent the text string "$y_1 > y_2 > \cdots > y_n$".

  - Create one voter $v_E$ with preference order:

$$\overrightarrow{E} > a > c > \overrightarrow{\overline{E}} > \overrightarrow{B_2} > b > d. \tag{4.3}$$

- Additionally, create two voters with preference order:

$$c > \overrightarrow{B_1} > a > \overrightarrow{B_2} > b > d, \tag{4.4}$$

  and create $\|\mathcal{S}_1\| - 1$ voters with preference order:

$$\overrightarrow{B_1} > c > a > \overrightarrow{B_2} > b > d. \tag{4.5}$$

- **Voters representing $\mathcal{S}_2$:** The case of $\mathcal{S}_2$ is treated analogously with the roles of respectively $\mathcal{S}_1$, $B_1$, $x_i$, $c$, $a$, $E$, $e_j$, and $\overline{e}_k$ interchanged with $\mathcal{S}_2$, $B_2$, $y_i$, $d$, $b$, $F$, $f_j$, and $\overline{f}_k$. More precisely, for each set $F \in \mathcal{S}_2$, create a single voter $v_F$ as follows:
  - Enumerate $F$ as $\{f_1, f_2, \dots, f_{\|F\|}\}$ (renaming the candidates $f_j$ chosen from $\{y_1, y_2, \dots, y_n\}$ for notational convenience), and enumerate its complement $\overline{F} = B_1 - F$ as $\{\overline{f}_1, \overline{f}_2, \dots, \overline{f}_{n-\|F\|}\}$.
  - To make the preference orders easier to parse, we use

$$\begin{array}{lll} \text{“}\overrightarrow{F}\text{”} & \text{to represent the text string} & \text{“}f_1 > f_2 > \cdots > f_{\|F\|}\text{”;} \\ \text{“}\overrightarrow{\overline{F}}\text{”} & \text{to represent the text string} & \text{“}\overline{f}_1 > \overline{f}_2 > \cdots > \overline{f}_{n-\|F\|}\text{”.} \end{array}$$

  - Create one voter $v_F$ with preference order:

$$\overrightarrow{F} > b > d > \overrightarrow{\overline{F}} > \overrightarrow{B_1} > a > c. \tag{4.6}$$

- Additionally, create two voters with preference order:

$$d > \overrightarrow{B_2} > b > \overrightarrow{B_1} > a > c, \tag{4.7}$$

  and create $\|\mathcal{S}_2\| - 1$ voters with preference order:

$$\overrightarrow{B_2} > d > b > \overrightarrow{B_1} > a > c. \tag{4.8}$$

We now prove Equation (4.1): $\text{YoungScore}(C, c, V) = 2 \cdot \kappa(\mathcal{S}_1) + 1$.

Let $E_1, E_2, \dots, E_{\kappa(\mathcal{S}_1)} \in \mathcal{S}_1$ be $\kappa(\mathcal{S}_1)$ pairwise disjoint subsets of $B_1$. Consider the following submultiset $\widehat{V}$ of the voters $V$. $\widehat{V}$ consists of:

- every voter $v_{E_i}$ corresponding to the set $E_i$, where $1 \leq i \leq \kappa(\mathcal{S}_1)$;
- the two voters given in Equation (4.4);
- $\kappa(\mathcal{S}_1) - 1$ voters of the form given in Equation (4.5).

Then, $\|\widehat{V}\| = 2 \cdot \kappa(\mathcal{S}_1) + 1$. Note that a strict majority of the voters in $\widehat{V}$ prefer $c$ over any other candidate, and thus $c$ is a Condorcet winner in $\langle C, \widehat{V}\rangle$. Hence,

$$\text{YoungScore}(C, c, V) \geq 2 \cdot \kappa(\mathcal{S}_1) + 1.$$

Conversely, to prove that $\text{YoungScore}(C, c, V) \leq 2 \cdot \kappa(\mathcal{S}_1) + 1$, we need the following lemma.

**Lemma 4.2.8** *For any $\lambda$ with $3 < \lambda \leq \|\mathcal{S}_1\| + 1$, let $V_\lambda$ be any submultiset of $V$ such that $V_\lambda$ contains exactly $\lambda$ voters of the form (4.4) or (4.5) and $c$ is a Condorcet winner in $\langle C, V_\lambda \rangle$. Then, $V_\lambda$ contains exactly $\lambda - 1$ voters of the form (4.3) and no voters of the form (4.6), (4.7), or (4.8). Moreover, the $\lambda - 1$ voters of the form (4.3) in $V_\lambda$ represent pairwise disjoint sets from $\mathcal{S}_1$.*

**Proof of Lemma** 4.2.8 Let $V_\lambda$ for fixed $\lambda$ be given as above. Consider the submultiset of $V_\lambda$ that consists of the $\lambda$ voters of the form (4.4) or (4.5). Every candidate $x_i$, $1 \leq i \leq m$, is preferred to $c$ by the at least $\lambda - 2$ voters of the form (4.5). Since $c$ is a Condorcet winner in $\langle C, V_\lambda \rangle$, there exist, for every $x_i$, at least $\lambda - 1 > 2$ voters in $V_\lambda$ who prefer $c$ to $x_i$. By construction, these voters must be of the form (4.3) or (4.4). Since there are at most two voters of the form (4.4), there exists at least one voter of the form (4.3), say $\tilde{v}$. Since the voters of the form (4.3) represent $\mathcal{S}_1$, which contains only nonempty sets, there exists some candidate $x_j$ who is preferred to $c$ by $\tilde{v}$. In particular, $c$ must outpoll $x_j$ in $\langle C, V_\lambda \rangle$ and thus needs more than $(\lambda - 2) + 1$ votes of the form (4.3) or (4.4). There are at most two voters of the form (4.4); hence, $c$ must be preferred by at least $\lambda - 2$ voters of the form (4.3) that are distinct from $\tilde{v}$. Summing up, $V_\lambda$ contains at least $\lambda - 1$ voters of the form (4.3).

On the other hand, since $c$ is a Condorcet winner in $\langle C, V_\lambda \rangle$, $c$ must in particular outpoll $a$, who is not preferred to $c$ by the $\lambda$ voters of the form (4.4) or (4.5) and who is preferred to $c$ by all other voters. Hence, $V_\lambda$ may contain at most $\lambda - 1$ voters of the form (4.3), (4.6), (4.7), or (4.8). It follows that $V_\lambda$ contains exactly $\lambda - 1$ voters of the form (4.3) and no voters of the form (4.6), (4.7), or (4.8).

For a contradiction, suppose that there is a candidate $x_j$ who is preferred to $c$ by more than one voter of the form (4.3) in $V_\lambda$. Then,

- $c$ is preferred to $x_j$ by at most two voters of the form (4.4) and by at most $(\lambda - 1) - 2 = \lambda - 3$ voters of the form (4.3);
- $x_j$ is preferred to $c$ by at least $\lambda - 2$ voters of the form (4.5) and by at least two voters of the form (4.3).

Since $c$ thus has at most $\lambda - 1$ votes and $x_j$ has at least $\lambda$ votes in $V_\lambda$, $c$ is not a Condorcet winner in $\langle C, V_\lambda \rangle$, a contradiction. Thus, every candidate $x_i$, $1 \leq i \leq m$, is preferred to $c$ by at most one voter of the form (4.3) in $V_\lambda$, which means that the $\lambda - 1$ voters of the form (4.3) in $V_\lambda$ represent pairwise disjoint sets from $\mathcal{S}_1$. ∎ (Lemma 4.2.8)

To continue the proof of Theorem 4.2.7, let $k = \text{YoungScore}(C, c, V)$. Let $\widehat{V} \subseteq V$ be a submultiset of size $k$ such that $c$ is a Condorcet winner in $\langle C, \widehat{V} \rangle$. Suppose that there are exactly $\lambda \leq \|\mathcal{S}_1\| + 1$ voters of the form (4.4) or (4.5) in $\widehat{V}$. Since $c$, the Condorcet winner of $\langle C, \widehat{V} \rangle$, must in particular outpoll $a$, we have $\lambda \geq \left\lceil \frac{k+1}{2} \right\rceil$. By our assumption that $\kappa(\mathcal{S}_1) > 2$, it follows from $k \geq 2 \cdot \kappa(\mathcal{S}_1) + 1$ that $\lambda > 3$. Lemma 4.2.8 then implies that there are exactly $\lambda - 1$ voters of the form (4.3) in $\widehat{V}$, which represent pairwise disjoint sets from $\mathcal{S}_1$, and $\widehat{V}$ contains no voters of the form (4.6), (4.7), or (4.8). Hence, $k = 2 \cdot \lambda - 1$ is odd, and $\frac{k-1}{2} = \lambda - 1 \leq \kappa(\mathcal{S}_1)$, which proves Equation (4.1).

Equation (4.2) can be proved analogously. Thus, we have

$$\kappa(\mathcal{S}_1) \geq \kappa(\mathcal{S}_2) \quad \text{if and only if} \quad \text{YoungScore}(C, c, V) \geq \text{YoungScore}(C, d, V).$$

This completes the proof of Theorem 4.2.7. ▮ (Theorem 4.2.7)

**Theorem 4.2.9** `Young Winner` *is* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete.*

**Proof** To prove the theorem, we modify the reduction from Theorem 4.2.7 to a reduction from the problem `Maximum Set Packing Compare` to the problem `Young Winner` as follows. Let $\langle C, V\rangle$ be the preference profile constructed in the proof of Theorem 4.2.7 with the designated candidates $c$ and $d$. We alter this profile such that all other candidates do worse than $c$ and $d$.

From $\langle C, V\rangle$, we construct a new preference profile $\langle D, W\rangle$. To define the new set $D$ of candidates, replace every candidate $g \in C$ except $c$ and $d$ by $\|V\|$ candidates $g^1, g^2, \dots, g^{\|V\|}$.

To define the new voter set $W$, replace each occurrence of candidate $g$ in the $i$th voter of $V$ by the text string:

$$g^{i \mod \|V\|} > g^{i+1 \mod \|V\|} > g^{i+2 \mod \|V\|} > \cdots > g^{i+\|V\|-1 \mod \|V\|}.$$

Let $V'$ be any submultiset of $V$, and let $W'$ be the submultiset of $W$ corresponding to $V'$. It is easy to see that $c$ is a Condorcet winner in $V'$ if and only if $c$ is a Condorcet winner in $W'$. Thus, the change from $\langle C, V\rangle$ to $\langle D, W\rangle$ does not alter the Young score of $c$ and $d$. On the other hand, the Young score of any other candidate now is at most 1. Thus, there is no candidate $h$ with $\text{YoungScore}(D, h, W) > \text{YoungScore}(D, c, W)$ or $\text{YoungScore}(D, h, W) > \text{YoungScore}(D, d, W)$. Hence, $\kappa(\mathcal{S}_1) \geq \kappa(\mathcal{S}_2)$ if and only if $c$ is a winner of the election $\langle D, W\rangle$. ▮ (Theorem 4.2.9)

## 4.3 Homogeneous Young and Dodgson Voting Schemes

Social choice theorists have studied many "reasonable" properties that any "fair" election procedure arguably should satisfy, including very natural properties such as nondictatorship, monotonicity, the Pareto Principle, and independence of irrelevant alternatives. One of the most notable results in this regard is Arrow's famous Impossibility Theorem [Arr63] stating that the just-mentioned four properties are logically inconsistent, and thus no "fair" voting scheme can exist.

In this section, we are concerned with another quite natural property, the homogeneity of voting schemes (see [Fis77, You77]).

**Definition 4.3.1** *A voting scheme $f$ is said to be* homogeneous *if and only if for each preference profile $\langle C, V\rangle$ and for all positive integers $q$, it holds that*

$$f(\langle C, V\rangle) = f(\langle C, qV\rangle),$$

*where $qV$ denotes $V$ replicated $q$ times.*

Homogeneity means that splitting each voter $v \in V$ into $q$ voters, each of whom has the same preference order as $v$, yields exactly the same choice set of winning candidates.

Fishburn [Fis77] showed that neither the Dodgson nor the Young voting schemes are homogeneous. For the Dodgson SCF, he presented a counterexample with seven voters and eight candidates; for the Young SCF, he modified a preference profile constructed by Young with 37 voters and five candidates. Fishburn [Fis77] provided the following limit device in order to define homogeneous variants of the Dodgson and Young SCFs. For example, the Dodgson scheme can be made homogeneous by defining from the function DodgsonScore for each preference profile $\langle C, V\rangle$ and designated candidate $c \in C$ the function

$$\text{DodgsonScore}^*(C, c, V) = \lim_{q\to\infty} \frac{\text{DodgsonScore}(C, c, qV)}{q}.$$

The resulting SCF is denoted by Dodgson$^*$ SCF, and the corresponding winner and ranking problems are denoted by `Dodgson`$^*$ `Winner` and `Dodgson`$^*$ `Ranking`.

**Example 4.3.2 (Fishburn [Fis77])** *We provide here Fishburn's example* [Fis77] *showing that the original Dodgson voting scheme is not homogeneous. Consider the preference profile $\langle C, V\rangle$, where $C$ consists of the eight candidates $a_1$, $a_2$, …, $a_7$, and $c$, and $V$ consists of the following preference orders:*

$$a_1 > a_2 > a_3 > a_4 > c > a_5 > a_6 > a_7,$$
$$a_7 > a_1 > a_2 > a_3 > c > a_4 > a_5 > a_6,$$
$$a_6 > a_7 > a_1 > a_2 > c > a_3 > a_4 > a_5,$$
$$a_5 > a_6 > a_7 > a_1 > c > a_2 > a_3 > a_4,$$
$$a_4 > a_5 > a_6 > a_7 > c > a_1 > a_2 > a_3,$$
$$a_3 > a_4 > a_5 > a_6 > c > a_7 > a_1 > a_2,$$
$$a_2 > a_3 > a_4 > a_5 > c > a_6 > a_7 > a_1.$$

*One can verify that* DodgsonScore$(C, c, V) = 7$ *and* DodgsonScore$(C, a_i, V) = 6$, *for each $i$. Thus, according to the original Dodgson scheme, the choice set of winning candidates in $\langle C, V\rangle$ is $\{a_i \mid 1 \leq i \leq 7\}$. However,* DodgsonScore$^*(C, c, V) =$ 3.5 *and* DodgsonScore$^*(C, a_i, V) = 4.5$, *for each $i$, which implies that, according to the original Dodgson scheme and for a large enough $q$, the choice set of winning candidates in $\langle C, qV\rangle$ is $\{c\}$. Hence, the original Dodgson voting scheme is not homogeneous.*

Analogously, the Young voting scheme defined in Section 4.2.2 can be made homogeneous by defining YoungScore$^*$. Young [You77] showed that the corresponding problem `Young`$^*$ `Winner` can be solved by a linear program of polynomial size. Hence, the problem `Young`$^*$ `Winner` is efficiently solvable, since linear programs can be solved in polynomial time [Kha79], see also [Kar84]. Inspired by Young's work, we establish an analogous result for the problems `Dodgson`$^*$ `Winner` and `Dodgson`$^*$ `Ranking` below. Theorem 4.3.3 should be contrasted with the known result [HHR97a] that `Dodgson Winner` and `Dodgson Ranking` are complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$.

**Theorem 4.3.3** `Dodgson`$^*$ `Winner` *and* `Dodgson`$^*$ `Ranking` *can be solved in polynomial time.*

**Proof** Bartholdi, Tovey, and Trick [BTT89b] provided an integer linear program for determining the Dodgson score of a given candidate $c$. They noted that if the number of candidates is fixed, then the winner problem for Dodgson elections (in the inhomogeneous case defined in Section 4.2.2) can be solved in polynomial time using the algorithm of Lenstra [Len83].

Based on their integer linear program, we provide a linear program for computing DodgsonScore$^*(C, c, V)$ for a given preference profile $\langle C, V\rangle$ and a given candidate $c$. Since linear programms are polynomial-time solvable [Kha79], it follows that the problems `Dodgson`$^*$ `Winner` and `Dodgson`$^*$ `Ranking` can be solved in polynomial time, even if the number of candidates is not prespecified.

Let a profile $\langle C, V\rangle$ and a candidate $c \in C$ be given, and let $V = \{v_1, v_2, \ldots, v_n\}$. Our linear program has the variables $x_{i,j}$, and constants $e_{i,j,k}$, and $w_k$, where $1 \leq i \leq n$, $1 \leq j \leq \|C\| - 1$, and $k \in C - \{c\}$. The constants are obtained from the profile $\langle C, V\rangle$ as follows:

- For given $i$, $j$, and $k$, set $e_{i,j,k} = 1$ if the result of moving $c$ upwards by $j$ positions in the preference order of voter $v_i$ is that $c$ gains one additional vote against candidate $k$, and set $e_{i,j,k} = 0$ otherwise.
- For any candidate $k$ other than $c$, the constant $w_k$ gives the number of voters who prefer $c$ over $k$.

DodgsonScore$^*(C, c, V)$ is the value of the linear program

$$\min \sum_{i,j} j \cdot x_{i,j} \tag{4.9}$$

subject to the constraints:

(1) $\sum_j x_{i,j} = 1$ for each voter $v_i$;

(2) $\sum_{i,j} e_{i,j,k} \cdot x_{i,j} + w_k > \frac{n}{2}$ for each candidate $k \in C - \{c\}$;

(3) $0 \leq x_{i,j} \leq 1$ for each $i$ and $j$.

The variables and constraints can be interpreted as follows:

1. For given $i$ and $j$, the variable $x_{i,j}$ is a rational number in the interval $[0,1]$ (by the set of constraints (3)) that gives the percentage $\frac{v^q_{i,j}}{q}$, where $q$ is the least common multiple of the denominators in all $x_{i,j}$, and $v^q_{i,j}$ is the number of voters among the $q$ replicants of voter $v_i$ in which $c$ is moved upwards by $j$ positions.

2. The set of constraints (2) ensures that $c$ becomes a Condorcet winner.

3. The set of constraints (1) ensures that $v^q_{i,j}$, summed over all possible positions $j$, equals the number $q$ of all replicants of voter $v_i$.

The objective is to minimize the number of switches needed to make $c$ a Condorcet winner. For the homogeneous case of Dodgson elections, the linear program (4.9) tells us how many times we have to replicate each voter $v_i$ (namely, $q$ times) and in how many of the replicants of each voter $v_i$ the given candidate $c$ has to be moved upwards by how many positions in order to achieve this objective.

∎ (Theorem 4.3.3)

# Chapter 5

# The Complexity of Kemeny Elections

## 5.1 Introduction

In this chapter, we investigate the complexity of determining the winner of Kemeny's voting scheme. Kemeny's voting scheme, which will be described in Section 5.2, was introduced by Kemeny [Kem59] and specified by Levenglick [Lev75]. Young and Levenglick [YL78] showed that Kemeny's voting scheme is the unique Condorcet voting scheme that is neutral (symmetric in its treatment of candidates) and consistent. A voting scheme is called consistent if it is consistent over disjoint voter set union: Whenever two subsets $V_1$ and $V_2$ of the voter set $V$ with $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$, voting separately, elect the same set of winners according to the voting scheme, then the set of winners obtained by applying the voting scheme to the *whole* set of voters yields this same set of winners. Bartholdi et al. [BTT89b] proved that determining the winner in Kemeny's voting scheme is NP-hard. The exact complexity of Kemeny's voting scheme however remained an open problem. We show that the winner problem for Kemeny's voting scheme is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.

## 5.2 Kemeny's Voting Scheme

Like Dodgson's and Young's voting schemes, Kemeny's voting scheme is a preferential voting system . Each voter casts his or her vote by ranking all the candidates in order of preference. We allow the voters to be indifferent between candidates (ties).[1] For example, a voter may rank candidates $a$, $b$, $c$, $d$, and $e$ by

[1]This follows Kemeny's original definition [Kem59]. It should be noted that there is no consensus about this in the literature. To avoid confusion, we will use the term preference *ranking* here, rather than preference *order*. It follows immediately from our proofs that our complexity results go through if we do not allow ties in preference rankings.

the preference ranking $a > b = c > d > e$. Candidate $a$ is the favorite, and $e$ is the least favorite candidate in this ranking. Candidates $b$ and $c$ are considered to be of equal desirability, i.e., they are tied. A preference ranking without ties is called a strict preference ranking. We identify each voter with its preference ranking, and we will view the set of voters as a multiset of preference rankings.

Kemeny defined the outcome of an election as the collection of preference rankings that are "closest" to the preference rankings of the voters. Such a preference ranking is called a *Kemeny consensus*. A candidate is a *winner* of the election if it is a preferred candidate in a Kemeny consensus.

There are different ways to define closeness. For Kemeny elections, the goal is to minimize the *Kemeny score*: the sum of the distances to the preference rankings of the voters.

For each pair $P$, $Q$ of preference rankings, we define the distance

$$\mathrm{dist}(P, Q) = \sum_{\{c,d\}} d_{P,Q}(c, d),$$

where the sum is taken over all unordered pairs $\{c, d\}$ of candidates, and

$$d_{P,Q}(c,d) \;=\; \begin{cases} 0 & \text{if } P \text{ and } Q \text{ agree on } c \text{ and } d \\ 1 & \text{if one of } P \text{ or } Q \text{ has a preference among } c \text{ and } d \text{ and the} \\ & \text{other has not} \\ 2 & \text{if } P \text{ and } Q \text{ strictly disagree on } c \text{ and } d. \end{cases}$$

Given a set of candidates $C$ and a multiset of preference rankings $V$ on $C$, we define the following three Kemeny score functions.

- For every preference ranking $P$ on $C$,

$$\mathrm{KemenyScore}(C, P, V) = \sum_{Q \in V} \mathrm{dist}(P, Q).$$

- For every candidate $c \in C$,

$$\begin{aligned}\mathrm{KemenyScore}(C, c, V) = \min\{&\mathrm{KemenyScore}(C, P, V) \mid P \text{ is a preference} \\ &\text{ranking on } C, \text{ and } c \text{ is a preferred candidate in } P\}.\end{aligned}$$

- $\mathrm{KemenyScore}(C, V)$

  $= \min\{\mathrm{KemenyScore}(C, P, V) \mid P \text{ is a preference ranking on } C\}.$

**Definition 5.2.1** *We define the following decision problems related to Kemeny elections. (Compare with similar definitions for Dodgson's and Young's voting schemes in Chapter 4.)*

1. `Kemeny Score`

   *Instance: A set of candidates $C$; a multiset $V$ of preference rankings on $C$; a positive integer $k$.*
   *Question: Is* $\mathrm{KemenyScore}(C, V) \leq k$*?*

2. `Candidate Kemeny Score`

   *Instance: A set of candidates $C$; a multiset $V$ of preference rankings on $C$; a candidate $c \in C$; a positive integer $k$.*
   *Question: Is* $\mathrm{KemenyScore}(C, c, V) \leq k$*?*

3. `Kemeny Winner`

   *Instance: A set of candidates $C$; a multiset $V$ of preference rankings on $C$; a candidate $c \in C$.*
   *Question: Is there some Kemeny consensus $P$ in which no candidate is strictly preferred to $c$? Equivalently, is* $\mathrm{KemenyScore}(C, c, V) \leq \mathrm{KemenyScore}(C, d, V)$ *for all $d \in C$?*

4. `Kemeny Ranking`

   *Instance: A set of candidates $C$; a multiset $V$ of preference rankings on $C$; two distinguished candidates $c, d \in C$.*
   *Question: Does $c$ tie-or-defeat $d$ in the election? That is, is* $\mathrm{KemenyScore}(C, c, V) \leq \mathrm{KemenyScore}(C, d, V)$*?*

Bartholdi, Tovey, and Trick [BTT89b] showed that `Kemeny Score` is NP-complete and that `Kemeny Winner` and `Kemeny Ranking` are NP-hard. We now state the main result of this chapter.

**Theorem 5.2.2** `Kemeny Winner` *and* `Kemeny Ranking` *are* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete.*

## 5.3 The Completeness Proof

Now we prove the main result of this chapter, namely that `Kemeny Winner` and `Kemeny Ranking` are complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$. It is easy to show that `Kemeny Winner` and `Kemeny Ranking` are in $\mathrm{P}_{\|}^{\mathrm{NP}}$. The $\mathrm{P}_{\|}^{\mathrm{NP}}$ algorithms will use the set

Candidate Kemeny Score as an oracle. Note that Candidate Kemeny Score is clearly in NP. Let $C$ be a set of candidates and $V$ a multiset of preference rankings on $C$. For every $c \in C$, $\text{KemenyScore}(C, c, V) \leq ||V|| \cdot ||C||^2$, so we can in polynomial time in parallel query all tuples $\langle C, V, c, k\rangle$ to Candidate Kemeny Score for all $c \in C$ and all $k \leq ||V|| \cdot ||C||^2$. With the answers to all these queries in hand, we know $\text{KemenyScore}(C, c, V)$ for each candidate $c \in C$, since this is the smallest $k$ such that $\langle C, V, c, k\rangle \in$ Candidate Kemeny Score. From the Kemeny scores for the candidates, it is trivial to verify that a certain candidate is a winner (has the smallest Kemeny score) or that a certain candidate ties-or-defeats another (does not have a higher Kemeny score).

It remains to show that Kemeny Winner and Kemeny Ranking are hard for $\mathrm{P}_{||}^{\mathrm{NP}}$. The hardness proof consists of two parts. In Section 5.3.1 we give a reduction from the problem Feedback Arc Set Member to Kemeny Ranking and Kemeny Winner. The $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness of Feedback Arc Set Member is shown in Section 5.3.2.

## 5.3.1 The Reduction from the Feedback Arc Set Member Problem

In [BTT89b], NP-hardness for Kemeny Score is proved by a reduction from the NP-complete digraph problem Feedback Arc Set, which will be defined below. Our approach to prove $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness for Kemeny Winner is the following. Define a version Feedback Arc Set Member of Feedback Arc Set that is $\mathrm{P}_{||}^{\mathrm{NP}}$-complete and adapt the reduction from Feedback Arc Set to Kemeny Score so that it becomes a reduction from Feedback Arc Set Member to Kemeny Winner. We will then use this reduction to obtain a reduction from Feedback Arc Set Member to Kemeny Ranking.

We will start by defining Feedback Arc Set and Feedback Arc Set Member.

**Definition 5.3.1** *1. A* feedback arc set *(FAS) for a digraph $G$ is a set of arcs of $G$ that includes at least one arc from every cycle in $G$.*

*2.* Feedback Arc Set $= \{\langle G, k\rangle \mid$ *$G$ is a digraph, $k$ a positive integer, and $G$ has a feedback arc set of size at most $k$*$\}$.

*3.* Feedback Arc Set Member $= \{\langle G, v\rangle \mid$ *$G$ is an irreflexive and antisymmetric digraph, $v$ is a vertex of $G$, and some minimum size feedback arc set of $G$ contains all arcs entering $v$*$\}$.

We will now prove that Feedback Arc Set Member $\leq_{\mathrm{m}}^{\mathrm{p}}$ Kemeny Winner.

**Lemma 5.3.2** Feedback Arc Set Member $\leq_{\mathrm{m}}^{\mathrm{p}}$ Kemeny Winner.

**Proof** We will modify the reduction from Feedback Arc Set to Kemeny Score from [BTT89b] to construct a reduction $f$ from Feedback Arc Set Member to

`Kemeny Winner`. Suppose that we are given $\langle G, \widehat{c}\rangle$, where $G = \langle C, A\rangle$ is an irreflexive and antisymmetric digraph, and $\widehat{c} \in C$.

Due to a note by McGarvey [McG53], we can interpret $G$ as an election. We can in polynomial time compute an election $g(G) = \langle C, V\rangle$, where $V$ is a multiset of *strict* preference rankings on $C$ such that the number of voters is even and $[c \to d] \in A$ iff $||V||/2 + 1$ voters prefer $c$ to $d$ (and $||V||/2 - 1$ voters prefer $d$ to $c$), and $c$ and $d$ are unconnected iff exactly half of the voters prefer $c$ to $d$ (and exactly half of the voters prefer $d$ to $c$): For each arc $[\alpha_1 \to \alpha_2] \in A$ we create one voter with preference ranking $\alpha_1 > \alpha_2 > c_1 > c_2 > \cdots > c_{n-2}$ and one voter with preference ranking $c_{n-2} > \cdots > c_2 > c_1 > \alpha_1 > \alpha_2$, where $c_1, \ldots, c_{n-2}$ are the remaining candidates.

Let $f(\langle G, \widehat{c}\rangle) = \langle g(G), \widehat{c}\rangle = \langle C, V, \widehat{c}\rangle$. We have to show that $\langle G, \widehat{c}\rangle \in$ `Feedback Arc Set Member` if and only if $\langle C, V, \widehat{c}\rangle \in$ `Kemeny Winner`.

Lemma 3 in [BTT89b] can easily be strengthened such that the particular Kemeny winner is preserved. We get the following lemma.

**Lemma 5.3.3 ([BTT89b])** *Given a set of candidates $C$ and a multiset $V$ of* strict *preference rankings on $C$. If $c \in C$ is a Kemeny winner, then there exists a* strict *Kemeny consensus*[2] *such that $c$ is the preferred candidate in the consensus.* ∎

Our election $\langle C, V\rangle$ consists only of strict preference rankings. That justifies the definition of the following score function.

- $\text{StrictKemenyScore}(C, c, V) = \min\{\text{KemenyScore}(C, P, V) \mid P$ is a *strict* preference ranking on $C$, and $c$ is the preferred candidate in $P\}$.

Lemma 5.3.3 implies that the winners of $\langle C, V\rangle$ are the candidates with smallest StrictKemenyScore, because $\text{StrictKemenyScore}(C, c, V) = \text{KemenyScore}(C, c, V)$ for all Kemeny winners $c$.

We define the following functions in analogy to KemenyScore.

- For every strict preference ranking $P$ on $C$,

$$\text{Disagree}(G, P) = ||\{[c \to d] \mid [c \to d] \in A \text{ and } P \text{ prefers } d \text{ to } c\}||.$$

- For every candidate $c \in C$,

$$\text{Disagree}(G, c) = \min\{\text{Disagree}(G, P) \mid P \text{ is a strict preference ranking on } C, \text{ and } c \text{ is the preferred candidate in } P\}.$$

[2]A *strict* Kemeny consensus is a Kemeny consensus that is a *strict* preference ranking.

The following claim is implicit in [BTT89b].

**Claim 5.1** *Let $g(G) = \langle C, V\rangle$. Then the following is true for a function* FixedCost *that depends neither on $P$ nor on $c$.*

1. *For each strict preference ranking $P$,*

$$\text{KemenyScore}(C, P, V) = \text{FixedCost}(G) + 4\,\text{Disagree}(G, P).$$

2. *For any candidate $c \in C$,*

$$\text{StrictKemenyScore}(C, c, V) = \mathit{FixedCost}(G) + 4\,\text{Disagree}(G, c).$$

**Proof**

1. According to the definitions,

$$\text{KemenyScore}(C, P, V) = \sum_{Q\in V} \text{dist}(P, Q) = \sum_{\{c,d\}} \sum_{Q\in V} d_{P,Q}(c, d).$$

Given an unordered pair $\{c, d\}$, it is easy to see that for every strict preference ranking $P$,

$$\sum_{Q\in V} d_{P,Q}(c, d) = 2 \cdot \begin{cases} ||V||/2 - 1 \text{ if } [c \to d] \in A \text{ and } P \text{ prefers } c \text{ to } d, \\ ||V||/2 + 1 \text{ if } [c \to d] \in A \text{ and } P \text{ prefers } d \text{ to } c, \\ ||V||/2 - 1 \text{ if } [d \to c] \in A \text{ and } P \text{ prefers } d \text{ to } c, \\ ||V||/2 + 1 \text{ if } [d \to c] \in A \text{ and } P \text{ prefers } c \text{ to } d, \\ ||V||/2 \qquad \text{if } [c \to d] \notin A \text{ and } [d \to c] \notin A. \end{cases}$$

Hence

$$\begin{aligned} &\text{KemenyScore}(C, P, V) \\ &\quad = 2 \cdot \big(||V||/2 \cdot ||\, \big\{\{x, y\} \,\big|\, x \neq y, [x \to y] \notin A \text{ and } [y \to x] \notin A\big\}\, || \\ &\qquad + (||V||/2 - 1)\, ||A|| + 2\,\text{Disagree}(G, P)) \\ &\quad = \text{FixedCost}(G) + 4\,\text{Disagree}(G, P). \end{aligned}$$

Clearly, FixedCost$(G)$ does not depend on $P$.

2. By definition,

$$\begin{aligned} &\text{StrictKemenyScore}(C, c, V) \\ &\quad = \min\{\text{KemenyScore}(C, P, V) \,\big|\, P \text{ is a } \textit{strict} \text{ preference ranking on } C, \\ &\qquad\qquad \text{and } c \text{ is the preferred candidate in } P\} \\ &\quad = \min\{\text{FixedCost}(G) + 4\,\text{Disagree}(G, P) \,\big|\, P \text{ is a } \textit{strict} \text{ preference} \\ &\qquad\qquad \text{ranking on } C, \text{ and } c \text{ is the preferred candidate in } P\}, \end{aligned}$$

due to Claim 5.1(1). By the definition of Disagree$(G, c)$, it follows that

$$\text{StrictKemenyScore}(C, c, V) = \text{FixedCost}(G) + 4\,\text{Disagree}(G, c).$$

Hence Claim 5.1 is proved. ▮ (Claim 5.1)

**Claim 5.2**

$$\text{Disagree}(G,c) = \min\left\{||F|| \,\middle|\, F \textit{ is a FAS of } G \textit{ containing all arcs entering } c\right\}.$$

**Proof** To prove that the left-hand side is less than or equal to the right-hand side, suppose that $F$ is a FAS of $G$ that contains all arcs entering $c$. Let $\widehat{G}$ be the digraph that is obtained from $G$ if we throw away all arcs that belong to $F$. Then $\widehat{G}$ does not contain cycles and $\widehat{G}$ does not contain any arcs entering $c$. Order $C$, the set of vertices of $\widehat{G}$, as $c_1, c_2, \ldots, c_\ell$, where $\ell = ||C||$, such that $c_1 = c$ and for all arcs $[c_i \to c_j]$ in $\widehat{G}$, $i < j$. This is possible because $\widehat{G}$ is cycle free; therefore its transitive closure is a partial order, any total extension of which agrees with $\widehat{G}$. Now consider the preference ranking $P = c_1 > c_2 > \cdots > c_{\ell-2} > c_{\ell-1} > c_\ell$. Clearly, Disagree$(\widehat{G}, P) = 0$. Since $G$ consists of $\widehat{G}$ plus $||F||$ extra arcs, it follows that Disagree$(G,P) \leq ||F||$, and therefore Disagree$(G,c) \leq ||F||$.

To prove that the left-hand side is greater than or equal to the right-hand side, let $\widehat{P} = c_1 > c_2 > \cdots > c_{\ell-2} > c_{\ell-1} > c_\ell$ be a preference ranking on $C$ with $c_1 = c$ and Disagree$(G, \widehat{P})$ = Disagree$(G,c)$. Let $\widehat{G}$ be the graph we obtain if we delete from $G$ the Disagree$(G,c)$ arcs in $\widehat{F} = \{[c_i \to c_j] \mid [c_i \to c_j] \in G \text{ and } i > j\}$ (the arcs disagreeing with $\widehat{P}$). Graph $\widehat{G}$ does not contain cycles, is obtained from $G$ by removing Disagree$(G,c)$ arcs, and does not contain any arcs entering $c$ (since arcs entering $c$ would disagree with $\widehat{P}$). The arc set $\widehat{F}$ is a FAS for $G$. Moreover, $\widehat{F}$ contains all arcs entering $c$, because $G$ with the arcs in $\widehat{F}$ removed has no arcs entering $c$.

▮ (Claim 5.2)

Using Lemma 5.3.3, and Claims 5.1, and 5.2, it is easy to prove that the following statements are equivalent.

(i) Candidate $\widehat{c}$ is a Kemeny winner of the election $\langle C, V\rangle$.

(ii) Candidate $\widehat{c}$ has smallest KemenyScore$(C, \widehat{c}, V)$.
(from the definition of Kemeny winner)

(iii) Candidate $\widehat{c}$ has smallest StrictKemenyScore$(C, \widehat{c}, V)$.
(by Lemma 5.3.3)

(iv) Candidate $\widehat{c}$ has smallest Disagree$(G, \widehat{c})$.
(by Claim 5.1(2))

(v) Candidate $\widehat{c}$ has smallest

$$\min\{||F|| \mid F \text{ is a FAS of } G \text{ containing all arcs entering } \widehat{c}\}.$$

(by Claim 5.2)

(vi) There is a minimum size FAS of $G$ containing all arcs entering $\widehat{c}$.

To see (v) $\rightarrow$ (vi), note that for every FAS of $G$ there exists a vertex $v$ such that $F$ contains all arcs entering $v$. That concludes the proof of Lemma 5.3.2.[3]

▮ (Lemma 5.3.2)

We can use the reduction of the previous lemma to obtain a reduction from `Feedback Arc Set Member` to `Kemeny Ranking`. The main idea is to add a special candidate $\widehat{d}$ that is always a winner of the election. This way, $\widehat{c}$ is a winner of the election if and only if $\widehat{c}$'s KemenyScore is not greater than $\widehat{d}$'s KemenyScore.

**Lemma 5.3.4** `Feedback Arc Set Member` $\leq^{\mathrm{p}}_{\mathrm{m}}$ `Kemeny Ranking`.

**Proof** Suppose that $x = \langle G, \widehat{c}\rangle$, with $G$ an irreflexive, antisymmetric digraph and $\widehat{c} \in V(G)$. It is easy to see that the following hold.

1. $G$ has a minimum size feedback arc set that contains all vertices entering $\widehat{c}$ if and only if $(G \cup \langle\{\widehat{d}\}, \emptyset\rangle)$ has a minimum size feedback arc set that contains all vertices entering $\widehat{c}$.

2. Any minimum size feedback arc set of $(G \cup \langle\{\widehat{d}\}, \emptyset\rangle)$ contains all arcs entering $\widehat{d}$ (since there are no arcs entering $\widehat{d}$).

Let $\langle C, V\rangle = g(G \cup \langle\{\widehat{d}\}, \emptyset\rangle)$ where $g$ is defined as in the proof of Lemma 5.3.2. Then $\langle C, V, \widehat{c}\rangle = f(G \cup \langle\{\widehat{d}\}, \emptyset\rangle, \widehat{c})$ and $\langle C, V, \widehat{d}\rangle = f(G \cup \langle\{\widehat{d}\}, \emptyset\rangle, \widehat{d})$, where $f$ is the reduction from `Feedback Arc Set Member` to `Kemeny Winner` from Lemma 5.3.2. From the observations above, and the fact that $f$ is a reduction from `Feedback Arc Set Member` to `Kemeny Winner`, it follows that:

1. $G$ has a minimum size feedback arc set that contains all vertices entering $\widehat{c}$ if and only if $\widehat{c}$ is a winner of $\langle C, V\rangle$.

2. $\widehat{d}$ is a winner of $\langle C, V\rangle$.

It follows immediately that $G$ has a minimum size feedback arc set that contains all vertices entering $\widehat{c}$ if and only if $\text{KemenyScore}(C, \widehat{c}, V) \leq \text{KemenyScore}(C, \widehat{d}, V)$.

▮ (Lemma 5.3.4)

### 5.3.2 The Hardness of the Feedback Arc Set Member Problem

In order to conclude from Lemmas 5.3.2 and 5.3.4 that `Kemeny Winner` and `Kemeny Ranking` are $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-hard, we still need to show that

[3]At this point, it is easy to see that Kemeny's voting scheme is also $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-complete if we disallow ties in all preference rankings (cf. footnote 1).

`Feedback Arc Set Member` is $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard. Karp [Kar72] proved that `Feedback Arc Set` is NP-hard by reducing `Vertex Cover` to it. We will follow the same approach as in Section 5.3.1: We will define a $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete version `Vertex Cover Member` of `Vertex Cover`, and we will reduce `Vertex Cover Member` to `Feedback Arc Set Member`.

**Definition 5.3.5** `Vertex Cover Member` $= \{\langle G, v\rangle \mid$ *$G$ is a graph, $v$ is a vertex of $G$, and some minimum size vertex cover of $G$ contains $v$*$\}$.

**Lemma 5.3.6** `Vertex Cover Member` $\leq_{\mathrm{m}}^{\mathrm{p}}$ `Feedback Arc Set Member`.

**Proof** We will use Karp's reduction from `Vertex Cover` to `Feedback Arc Set` [Kar72]. Given an (undirected) graph $G$, define digraph $H = \langle W, A\rangle$ as follows.

- $W = \{v, v' \mid v \in V(G)\}$, and
- $A = \{[v \to v'] \mid v \in V(G)\} \cup \{[v' \to w], [w' \to v] \mid \{v, w\} \in E(G)\}$,

where $v'$ is a duplicate of $v$ for each vertex $v$. From [Kar72], we know that $G$ contains a vertex cover of size at most $k$ if and only if $H$ contains a feedback arc set of size at most $k$. Note that this implies that the minimum size of a vertex cover for $G$ is the same as the minimum size of a feedback arc set for $H$.

We modify Karp's reduction to obtain a reduction from `Vertex Cover Member` to `Feedback Arc Set Member` as follows. For $G$ a graph, and $\widehat{v} \in V(G)$, let $f(\langle G, \widehat{v}\rangle) = \langle H, \widehat{v}'\rangle$, where $H$ is the digraph from Karp's reduction defined above. Note that $H$ is irreflexive and antisymmetric and that $f$ is computable in polynomial time. It remains to show that $\langle G, \widehat{v}\rangle \in$ `Vertex Cover Member` if and only if $\langle H, \widehat{v}'\rangle \in$ `Feedback Arc Set Member`. This follows almost immediately from the following lemma.

**Lemma 5.3.7** *$G$ has a vertex cover of size at most $k$ containing $\widehat{v}$ if and only if $H$ has a feedback arc set of size at most $k$ containing $[\widehat{v} \to \widehat{v}']$.*

This completes the proof of Lemma 5.3.6, since $\langle G, \widehat{v}\rangle \in$ `Vertex Cover Member` if and only if $G$ has a minimum size vertex cover containing $\widehat{v}$. By Lemma 5.3.7 and the properties of Karp's reduction, this holds if and only if $H$ has a minimum size feedback arc set containing $[\widehat{v} \to \widehat{v}']$. This last step is equivalent to $\langle H, \widehat{v}'\rangle \in$ `Feedback Arc Set Member`, since $[\widehat{v} \to \widehat{v}']$ is the only arc entering $\widehat{v}'$.

▮ (Lemma 5.3.6)

**Proof of Lemma 5.3.7.** The proof follows from inspection of Karp's proof [Kar72]. We include the proof for the sake of completeness. First suppose that $V' \subseteq V(G)$ is a vertex cover for $G$ such that $||V'|| = k$ and $\widehat{v} \in V'$. We claim that $\{[v \to v'] \mid v \in V'\}$ is a feedback arc set for $H$. Note that the size of this set is $k$, and that $[\widehat{v} \to \widehat{v}']$ is in this set.

Suppose for a contradiction that $\langle W, A \setminus \{[v \to v'] \mid v \in V'\}\rangle$ contains a cycle. By construction of $H$, this cycle is of the following form for some $k \geq 2$, $v_i \neq v_j$ for all $i \neq j$, and all $v_i$ not in $V'$.

$$v_1 \to v_1' \to v_2 \to v_2' \to \cdots \to v_k \to v_k' \to v_1$$

But in $G$, this implies that $\{v_1, v_2\} \in E$, while $v_1, v_2 \notin V'$. This contradicts the fact that $V'$ is a vertex cover for $G$.

For the converse, suppose that $H$ has a feedback arc set $A'$ of size $k$ that contains $[\widehat{v} \to \widehat{v}']$. We claim that

$$V' = \{v \in V \mid \exists w \in W : [v \to w] \in A' \text{ or } [v' \to w] \in A'\}$$

forms a vertex cover. Note that $||V'|| \leq k$ and that $\widehat{v} \in V'$.

Suppose for a contradiction that $\{v_1, v_2\} \in E$ and $v_1, v_2 \notin V'$. Then (by definition)

$$[v_1 \to v_1'], [v_1' \to v_2], [v_2 \to v_2'], [v_2' \to v_1] \in A \setminus A'$$

so that we have a cycle in $\langle W, A \setminus A'\rangle$, which contradicts our assumption that $A'$ is a feedback arc set. ▮ (Lemma 5.3.7)

It remains to show that `Vertex Cover Member` is $\mathrm{P}_{||}^{\mathrm{NP}}$-hard. From Corollary 3.3.5 we know that `Minimum Vertex Cover Compare` is $\mathrm{P}_{||}^{\mathrm{NP}}$-complete. We give a reduction from `Minimum Vertex Cover Compare` to `Vertex Cover Member`.

**Lemma 5.3.8** `Minimum Vertex Cover Compare` $\leq_{\mathrm{m}}^{\mathrm{p}}$ `Vertex Cover Member`.

**Proof** Let $G$ and $H$ be graphs such that $||V(H)|| = ||V(G)||$. Let $v$ and $w$ be two new vertices and let $F = (G \cup \langle\{v\}, \emptyset\rangle) \bowtie (H \cup \langle\{w\}, \emptyset\rangle)$, where $\bowtie$ is the join operation and $\cup$ stands for the disjoint union of graphs. We claim that the following holds: $\langle G, H\rangle \in$ `Minimum Vertex Cover Compare` if and only if $F$ has a minimum vertex cover that contains $w$. Our reduction will map $\langle G, H\rangle$ to $\langle F, w\rangle$.

Recall that $\tau(G)$ denotes the size of a minimum vertex cover of $G$. Note that $V$ is a vertex cover of $F$ if and only if one of the two following statements holds.

1. $V$ contains all vertices of $(G \cup \langle\{v\}, \emptyset\rangle)$ and $V$ contains a vertex cover of $(H \cup \langle\{w\}, \emptyset\rangle)$. The smallest vertex cover of this form is of size $||V(G)|| + 1 + \tau(H \cup \langle\{w\}, \emptyset\rangle) = ||V(H)|| + 1 + \tau(H)$. Note that $w$ is not part of the smallest vertex cover of this type.

2. $V$ contains all vertices of $(H \cup \langle\{w\}, \emptyset\rangle)$ and $V$ contains and a vertex cover of $(G \cup \langle\{v\}, \emptyset\rangle)$. The smallest vertex cover of this form is of size $||V(H)|| + 1 + \tau(G \cup \langle\{v\}, \emptyset\rangle) = ||V(H)|| + 1 + \tau(G)$. Note that that $w$ is always part of a vertex cover of this type.

It follows that $w$ is an element of a minimum size vertex cover of $F$ if and only if there is a minimum size vertex cover of type 2 described above. This is the case if and only if $\tau(G) \leq \tau(H)$. ▮ (Lemma 5.3.8)

Lemmas 5.3.8, 5.3.6, 5.3.2, and 5.3.4 immediately imply that `Kemeny Winner` and `Kemeny Ranking` are $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard. This completes the proof of Theorem 5.2.2. In addition, these lemmas and the upper bounds for `Kemeny Winner` and `Kemeny Ranking` proved at the start of this section show that the intermediate graph problems used are also complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$.

**Corollary 5.3.9** `Feedback Arc Set Member` *and* `Vertex Cover Member` *are complete for* $\mathrm{P}_{\|}^{\mathrm{NP}}$*.*

# Chapter 6

# Recognizing When Vertex Cover Heuristics Can Do Well

## 6.1 Introduction

To cope with the intractability that appears to be inherent to the minimum vertex cover problem, various heuristics for finding minimum vertex covers have been proposed. Two of the most prominent such heuristics are the *edge deletion heuristic* and the *maximum-degree greedy heuristic*, see, e.g., [PS82, Pap94]. These algorithms run in linear time and, depending on the structure of the given input graph, may find a minimum vertex cover, or may provide a good approximation of the optimal solution.

It is common to evaluate heuristics for optimization problems by analyzing their worst-case ratio for approximating the optimal solution. In this regard, the two heuristics considered behave quite differently: The edge deletion heuristic always approximates the size of a minimum vertex cover within a factor of 2 and thus achieves essentially the best approximation ratio known, whereas the maximum-degree greedy heuristic, in the worst case, can have an approximation ratio as bad as logarithmic in the input size. The latter result follows from the early analysis of the approximation behavior of the greedy algorithm for the minimum set cover problem that was done by Johnson [Joh74], Lovász [Lov75], and Chvátal [Chv79] (who studied the weighted version of minimum set cover). Note that the vertex cover problem can be seen as the special case of the set cover problem, restricted so that each element occurs in exactly two sets. The currently best polynomial-time approximation algorithm for the minimum vertex cover problem by Karakostas [Kar04] achieves an approximation ratio of $2 - \Theta\left(\frac{1}{\sqrt{\log n}}\right)$, where $n$ is the number of vertices. Dinur and Safra [DS02] proved that the minimum vertex cover problem cannot be approximated to within any factor smaller than $10\sqrt{5} - 21 \approx 1.36067$, unless P = NP.

We study the problem of recognizing those input graphs for which either of the two heuristics can approximate the size of a minimum vertex cover within a constant factor of $r$, where $r \geq 1$ is a fixed rational number. Let $\mathcal{S}_r^{\mathrm{ED}}$ and $\mathcal{S}_r^{\mathrm{MDG}}$, respectively, denote this recognition problem for the edge deletion heuristic and for the maximum-degree greedy heuristic. Our main results are:

- For each rational number $r$ with $1 \leq r < 2$, $\mathcal{S}_r^{\mathrm{ED}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete (see Theorem 6.3.2).
- For each rational number $r \geq 1$, $\mathcal{S}_r^{\mathrm{MDG}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete (see Theorems 6.4.3 and 6.4.4).

This type of recognition problem was investigated for other problems and other heuristics as well. Bodlaender, Thilikos, and Yamazaki [BTY97] defined and studied the analogous problem for the independent set problem and the minimum-degree greedy heuristic, which they denoted by $\mathcal{S}_r$. They proved that $\mathcal{S}_r$ is coNP-hard and belongs to $\mathrm{P}^{\mathrm{NP}}$. Closing the gap between these lower and upper bounds, Hemaspaandra and Rothe [HR98] proved that $\mathcal{S}_r$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.

Like in previous chapters, we obtain $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness by a reduction from the `Minimum Vertex Cover Compare` problem (see Section 3.3). Also, we show that the vertex cover problem, restricted to those input graphs for which the heuristics considered can find an optimal solution, remains NP-hard. We then lift these NP lower bounds to $\mathrm{P}_{\|}^{\mathrm{NP}}$ lower bounds that prove our main results. This lifting requires a padding technique such that the given approximation ratio $r$ is precisely met. In particular, to achieve $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness of $\mathcal{S}_r^{\mathrm{MDG}}$ for each rational number $r \geq 1$, we modify a construction by Papadimitriou and Steiglitz [PS82] that they use to analyze the worst-case approximation behavior of the maximum-degree greedy heuristic.

## 6.2 Two Heuristics for the Vertex Cover Problem

We consider the following two heuristics (see, e.g., [PS82, Pap94]) for finding a minimum vertex cover of a given graph:

**Edge Deletion Heuristic (ED):** Given a graph $G$, the algorithm outputs a vertex cover $C$ of $G$. Initially, $C$ is the empty set. Nondeterministically choose an edge $\{u, v\} \in E(G)$, add both $u$ and $v$ to $C$, and delete $u$, $v$, and all edges incident to $u$ and $v$ from $G$. Repeat until there is no edge left in $G$.

**Maximum-Degree Greedy Heuristic (MDG):** Given a graph $G$, the algorithm outputs a vertex cover $C$ of $G$. Initially, $C$ is the empty set. Nondeterministically choose a vertex $v \in V(G)$ of maximum degree, add $v$ to $C$, and delete $v$ and all edges incident to $v$ from $G$. Repeat until there is no edge left in $G$.

As mentioned in the introduction, these two heuristics have a quite different approximation behavior. While the worst-case ratio of the MDG algorithm is logarithmic in the input size [Pap94, Joh74], the ED algorithm always approximates the optimal solution within a factor of 2. Thus, despite its extreme simplicity, the edge deletion heuristic achieves essentially[1] the best approximation ratio known for finding minimum vertex covers [Pap94].

The central question raised in this chapter is: How hard is it to determine for which graphs $G$ either of these two heuristics can approximate the minimum vertex cover of $G$ within a factor of $r$, for a given rational number $r \geq 1$? Let *min-ed*$(G)$ (respectively, *min-mdg*$(G)$ ) denote the minimum size of the output set of the ED algorithm (respectively, of the MDG algorithm) on input $G$, where the minimum is taken over all possible sequences of nondeterministic choices the algorithms can make. For any fixed rational $r \geq 1$, $\mathcal{S}_r^{\mathrm{ED}}$ (respectively, $\mathcal{S}_r^{\mathrm{MDG}}$) is the class of graphs for which ED (respectively, MDG) can output a vertex cover of size at most $r$ times the size of a minimum vertex cover. Formally,

$$\begin{aligned} \mathcal{S}_r^{\mathrm{ED}} &= \{G \mid G \text{ is a graph and } \mathit{min\text{-}ed}(G) \leq r \cdot \tau(G)\}; \\ \mathcal{S}_r^{\mathrm{MDG}} &= \{G \mid G \text{ is a graph and } \mathit{min\text{-}mdg}(G) \leq r \cdot \tau(G)\}. \end{aligned}$$

We will prove that for each fixed rational number $r$ with $1 \leq r < 2$, $\mathcal{S}_r^{\mathrm{ED}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete, and that for each fixed rational number $r \geq 1$, $\mathcal{S}_r^{\mathrm{MDG}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete. To this end, we give reductions from the $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete `Minimum Vertex Cover Compare` problem. Here we need the additional useful property of the reduction given by Theorem 3.3.4.

## 6.3 The Edge Deletion Heuristic

Lemma 6.3.1 below states that the vertex cover problem restricted to graphs in $\mathcal{S}_1^{\mathrm{ED}}$ is NP-hard. The reduction $g$ from Lemma 6.3.1 will be used in the proof of the main result of this section, Theorem 6.3.2. Define the problem

$$\texttt{VC-}\mathcal{S}_1^{\mathrm{ED}} = \{\langle G, k\rangle \mid G \in \mathcal{S}_1^{\mathrm{ED}} \text{ and } k \in \mathbb{N}^+ \text{ and } \tau(G) \leq k\}.$$

**Lemma 6.3.1** *There exists a polynomial-time many-one reduction $g$ from* `Vertex Cover` *to* `VC-`$\mathcal{S}_1^{\mathrm{ED}}$ *transforming any given graph $G$ into a graph $H \in \mathcal{S}_1^{\mathrm{ED}}$ such that*

$$\tau(H) = 2(\tau(G) + \|V(G)\|). \tag{6.1}$$

*Hence,* `VC-`$\mathcal{S}_1^{\mathrm{ED}}$ *is* NP*-hard.*[2]

[1]See [Kar04].

[2]Theorem 6.3.2 will imply that `VC-`$\mathcal{S}_1^{\mathrm{ED}}$ in fact is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.

**Proof** Given any graph $G$, we construct the graph $H \in \mathcal{S}_1^{\mathrm{ED}}$ as follows. For each vertex $v \in V(G)$, create a component $G_v$ that is defined by the vertex set $V(G_v) = \{v_1, v_2, v_3, v_4\}$ and the edge set $E(G_v) = \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_1, v_3\}\}$.

Define the graph $H$ by joining every pair of components that correspond to adjacent vertices of $G$:

$$\begin{aligned} V(H) &= \bigcup_{v \in V(G)} V(G_v); \\ E(H) &= \{\{a_i, b_j\} \mid \{a, b\} \in E(G) \text{ and } i, j \in \{1,2,3,4\}\} \cup \bigcup_{v \in V(G)} E(G_v). \end{aligned}$$

We now prove Equation (6.1). Let $C$ be a minimum vertex cover of $G$, i.e., $\tau(G) = \|C\|$. Construct a vertex cover $D$ of $H$ as follows. For each vertex $v \in C$, add $v_1, v_2, v_3$, and $v_4$ to $D$; and for each vertex $w \in V(G) - C$, add $w_1$ and $w_3$ to $D$. Hence,

$$\|D\| = 2(\|C\| + \|V(G)\|).$$

Since $mvc(H) \leq \|D\|$, it follows that

$$\tau(H) \leq 2(\tau(G) + \|V(G)\|).$$

Conversely, let $D$ be a minimum vertex cover of $H$, i.e., $\tau(H) = \|D\|$. Then, it holds that:

- for each edge $\{u, v\} \in E(G)$, $V(G_u) \subseteq D$ or $V(G_v) \subseteq D$;
- for each vertex $v \in V(G)$, $\|D \cap V(G_v)\| \geq 2$.

Hence,

$$\begin{aligned} \|D\| &\geq 4 \cdot \tau(G) + 2(\|V(G)\| - \tau(G)) \\ &= 2(\tau(G) + \|V(G)\|). \end{aligned}$$

It follows that

$$\tau(H) \geq 2(\tau(G) + \|V(G)\|),$$

which proves Equation (6.1).

It remains to prove that $H \in \mathcal{S}_1^{\mathrm{ED}}$. Let $C$ be a minimum vertex cover of $G$. The edge deletion algorithm can find a vertex cover of $H$ as follows. For every vertex $v \in C$, choose the edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$. For the remaining vertices $w \in V(G) - C$, choose the edge $\{w_1, w_3\}$. Thus, *min-ed*$(H) = 2(\tau(G) + \|V(G)\|)$. By Equation (6.1), *min-ed*$(H) = \tau(H)$, so $H \in \mathcal{S}_1^{\mathrm{ED}}$. ▮ (Lemma 6.3.1)

**Theorem 6.3.2** *For each rational number $r$ with $1 \leq r < 2$, $\mathcal{S}_r^{\mathrm{ED}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.*

**Proof** It is easy to see that $\mathcal{S}_r^{\mathrm{ED}}$ is in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$. To prove $\mathrm{P}_{\parallel}^{\mathrm{NP}}$-hardness, let $A$ be an arbitrary set in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$, and let $f$ be the reduction from $A$ to `Minimum Vertex Cover Compare` stated in Theorem 3.3.4. Fix any rational number $r$ with $1 \leq r < 2$, and let $\ell$ and $m$ be integers such that $r = \ell/m$. Note that $1 \leq m \leq \ell < 2m$.

For any string $x \in \Sigma^*$, let $f(x) = \langle G_2, G_1 \rangle$. Since we can add isolated vertices to any graph $G$ without altering $\tau(G)$, we may without loss of generality assume that $\|V(G_1)\| = \|V(G_2)\|$. Let $g$ be the reduction from Lemma 6.3.1 that transforms any given graph $G$ into a graph $H \in \mathcal{S}_1^{\mathrm{ED}}$ such that Equation (6.1) holds. Let $H_1 = g(G_1)$ and $H_2 = g(G_2)$. Thus, both $H_1$ and $H_2$ are in $\mathcal{S}_1^{\mathrm{ED}}$, and for $i \in \{1, 2\}$, we have $\tau(H_i) = 2(\tau(G_i) + \|V(G_i)\|)$.

We will define a graph $\widehat{H}$ and an integer $k \geq 0$ such that:

$$\begin{aligned} \textit{min-ed}(\widehat{H}) &= r(m \cdot \tau(H_2) + 2km); && (6.2) \\ \tau(\widehat{H}) &= m \cdot \tau(H_1) + 2km. && (6.3) \end{aligned}$$

The reduction mapping any given string $x$ (via the pair $\langle G_2, G_1 \rangle$ obtained according to Theorem 3.3.4 and via the pair $\langle H_2, H_1 \rangle$ obtained according to Lemma 6.3.1) to the graph $\widehat{H}$ such that Equations (6.2) and (6.3) are satisfied will establish that $A \leq_{\mathrm{m}}^{\mathrm{p}} \mathcal{S}_r^{\mathrm{ED}}$. In particular, from these equations, we have that:

- $\tau(H_2) = \tau(H_1)$ implies $\textit{min-ed}(\widehat{H}) = r \cdot \tau(\widehat{H})$, and
- $\tau(H_2) > \tau(H_1)$ implies $\textit{min-ed}(\widehat{H}) > r \cdot \tau(\widehat{H})$.

Note that, due to Theorem 3.3.4, $\tau(H_2) \geq \tau(H_1)$.

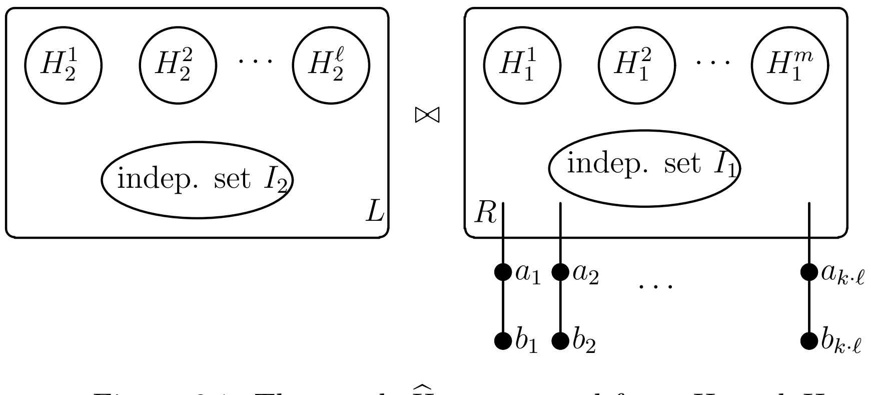

Figure 6.1: The graph $\widehat{H}$ constructed from $H_1$ and $H_2$.

Look at Figure 6.1 for the construction of $\widehat{H}$ from $H_1$ and $H_2$. The graph $\widehat{H}$ consists of two subgraphs, $L$ and $R$, that are joined by the join operation $\bowtie$, plus some additional vertices and edges that are connected to $R$. Formally, let $H_1^1, H_1^2, \ldots, H_1^m$ be $m$ pairwise disjoint copies of $H_1$, and let $H_2^1, H_2^2, \ldots, H_2^\ell$ be $\ell$ pairwise disjoint copies of $H_2$. Let $k = \ell\|V(H_2)\| + m\|V(H_1)\|$. Let $I_1$ and $I_2$ be independent sets such that $L$ contains exactly $k(2m-\ell)$ vertices and $R$ exactly $k\ell$ vertices. (This is possible, because $k(2m-\ell)-\ell\|V(H_2)\|$ is not negative, since $2m-\ell \geq 1$, and $k\ell - m\|V(H_1)\|$ is not negative, since $\ell \geq 1$.) Let $e_i = \{a_i, b_i\}$ $(1 \leq i \leq k\ell)$ be additional edges. Every vertex $a_i$ is adjacent to exactly one vertex in $R$, and each vertex in $R$ is adjacent to exactly one vertex $a_i$. The vertices $a_i$ and $b_i$ are not adjacent to any other vertices.

1. We first determine *min-ed*$(\widehat{H})$. Let $\widehat{E}$ be a fixed minimum-size output set of the ED algorithm on input $\widehat{H}$, i.e., *min-ed*$(\widehat{H}) = \|\widehat{E}\|$. Since $\widehat{E}$ is a vertex cover of $\widehat{H}$, $\widehat{E}$ must contain $a_i$ or $b_i$ for each $i \in \{1, \ldots, k\ell\}$. Since the ED-algorithm can delete only edges, and $\widehat{E}$ is a minimum-size output set, it follows that $\widehat{E}$ contains all vertices $a_i$, all vertices from $R$, and no vertex $b_i$.

   Let $C_L$ be a minimum-size output set of the ED-algorithm on input $L$. By construction of $L$, $\|C_L\| = \ell \cdot$ *min-ed*$(H_2)$. Thus, since $H_2 \in \mathcal{S}_1^{\mathrm{ED}}$, $\|C_L\| = \ell \cdot \tau(H_2)$.

   Define $\widehat{E}' = V(R) \cup C_L \cup \bigcup_{i=1}^{k\ell}\{a_i\}$. It is easy to see that $\widehat{E}'$ is a minimum-size output set of the ED algorithm on input $\widehat{H}$. Hence,

   $$\begin{aligned} \textit{min-ed}(\widehat{H}) &= 2k\ell + \ell \cdot \tau(H_2) \\ &= r(2km + m \cdot \tau(H_2)). \end{aligned}$$

   This proves Equation (6.2).

2. We now determine $\tau(\widehat{H})$. Let $\widehat{C}$ be a fixed minimum vertex cover of $\widehat{H}$, i.e., $\tau(\widehat{H}) = \|\widehat{C}\|$. Distinguish the following two cases.

   **Case 1:** $V(R) \subseteq \widehat{C}$. In this case, $\widehat{C}$ contains all vertices from $R$, at least one of $a_i$ or $b_i$ for each $i$, $1 \leq i \leq k\ell$, and a minimum vertex cover of $L$. Hence,

   $$\tau(\widehat{H}) = 2k\ell + \ell \cdot \tau(H_2).$$

   **Case 2:** $V(L) \subseteq \widehat{C}$. In this case, $\widehat{C}$ contains all vertices from $L$, a minimum vertex cover of $R$, and exactly one of $a_i$ or $b_i$ for each $i$, $1 \leq i \leq k\ell$. Hence,

   $$\begin{aligned} \tau(\widehat{H}) &= k(2m-\ell) + k\ell + m \cdot \tau(H_1) \\ &= 2km + m \cdot \tau(H_1). \end{aligned}$$

   Since $\tau(H_1) \leq \tau(H_2)$, $m \leq \ell$, and $2km \leq 2k\ell$, it follows that

   $$\tau(\widehat{H}) = 2km + m \cdot \tau(H_1).$$

   This proves Equation (6.3).

▌ (Theorem 6.3.2)

## 6.4 The Maximum-Degree Greedy Heuristic

Lemma 6.4.1 below states that the vertex cover problem restricted to graphs in $\mathcal{S}_1^{\mathrm{MDG}}$ is NP-hard. The proof of Lemma 6.4.1 is reminiscent of a proof by Bodlaender et al. [BTY97, Thm. 4], who show that the independent set problem restricted to graphs for which the minimum-degree greedy heuristic can find an optimal solution is NP-hard. The reduction $g$ from Lemma 6.4.1 will be used in the proof of the main result of this section, Theorem 6.4.4. Define the problem

$$\texttt{VC-}\mathcal{S}_1^{\mathrm{MDG}} = \{\langle G, k\rangle \mid G \in \mathcal{S}_1^{\mathrm{MDG}} \text{ and } k \in \mathbb{N}^+ \text{ and } \tau(G) \leq k\}.$$

Recall that $\Delta(G)$ denotes the maximum degree of the vertices in $G$.

**Lemma 6.4.1** *There is a polynomial-time many-one reduction $g$ from* `Vertex Cover` *to* $\texttt{VC-}\mathcal{S}_1^{\mathrm{MDG}}$ *transforming any given graph $G$ into a graph $H \in \mathcal{S}_1^{\mathrm{MDG}}$ such that*

$$\tau(H) = \tau(G) + \|E(G)\|(\Delta(G)+1). \tag{6.4}$$

*Hence,* $\texttt{VC-}\mathcal{S}_1^{\mathrm{MDG}}$ *is* NP*-hard.*[3]

**Proof** Given any graph $G$, we construct the graph $H \in \mathcal{S}_1^{\mathrm{MDG}}$ as follows. We replace each edge of $G$ by a gadget that contains a complete bipartite graph of size $2(\Delta(G)+1)$. Formally, $H$ is defined by:

$$\begin{aligned} V(H) &= V(G) \cup \bigcup_{e=\{u,v\}\in E(G)} \{u_i^e \mid 1 \leq i \leq \Delta(G)+1\} \cup \{v_i^e \mid 1 \leq i \leq \Delta(G)+1\}; \\ E(H) &= \bigcup_{e=\{u,v\}\in E(G)} \left(\{\{u_i^e, v_j^e\} \mid 1 \leq i,j \leq \Delta(G)+1\} \cup \{\{u, u_1^e\}\} \cup \{\{v, v_1^e\}\}\right). \end{aligned}$$

We now prove Equation (6.4). Let $C$ be a minimum vertex cover of $G$, i.e., $\tau(G) = \|C\|$. Note that $\{u,v\} \cap C \neq \emptyset$ for each edge $\{u,v\}$ in $E(G)$. Construct a vertex cover $D$ of $H$ as follows:

- $D$ contains all vertices from $C$.
- For every edge $e = \{u,v\}$ in $E(G)$, add to $D$:
  - either all vertices $u_i^e$, $1 \leq i \leq \Delta(G)+1$, if $u \notin C$ or if both $u$ and $v$ are in $C$;
  - or all vertices $v_i^e$, $1 \leq i \leq \Delta(G)+1$, if $v \notin C$.

It follows that

$$\tau(H) \leq \tau(G) + \|E(G)\|(\Delta(G)+1).$$

[3]Theorem 6.4.3 will imply that $\texttt{VC-}\mathcal{S}_1^{\mathrm{MDG}}$ in fact is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.

Conversely, let $D$ be a minimum vertex cover of $H$, i.e., $\tau(H) = \|D\|$. Construct a vertex cover $C$ of $G$ as follows. Initially, set $C = D$. Let $e = \{u, v\}$ be any fixed edge in $E(G)$. Suppose that at least one vertex from $\{u, v\}$ is in $D$. Since $D$ is a vertex cover of $H$, it contains at least $\Delta(G) + 1$ of the vertices $u_i^e$ and $v_i^e$, $1 \leq i \leq \Delta(G) + 1$, that correspond to the edge $e$. Remove any $\Delta(G) + 1$ such vertices from $C$. Suppose now that neither $u$ nor $v$ is in $D$. Since $D$ is a vertex cover of $H$, it contains at least $\Delta(G) + 2$ of the vertices $u_i^e$ and $v_i^e$, $1 \leq i \leq \Delta(G) + 1$, that correspond to the edge $e$. Remove any $\Delta(G) + 2$ such vertices from $C$, and add to $C$ one of $u$ or $v$ instead. Since the set $C$ thus obtained is a vertex cover of $G$, we have

$$\tau(H) \geq \tau(G) + \|E(G)\|(\Delta(G) + 1),$$

which proves Equation (6.4).

It remains to prove that $H \in \mathcal{S}_1^{\mathrm{MDG}}$. Let $C$ be a minimum vertex cover of $G$. The maximum-degree greedy algorithm can find a vertex cover of $H$ as follows. For every edge $e = \{u, v\}$ in $E(G)$, the MDG algorithm on input $H$ can choose:

- either all vertices $u_i^e$, $1 \leq i \leq \Delta(G) + 1$, if $u \notin C$ or if both $u$ and $v$ are in $C$;
- or all vertices $v_i^e$, $1 \leq i \leq \Delta(G) + 1$, if $v \notin C$.

Note that the MDG heuristic can always do so, since every vertex in $V(G)$ has degree at most $\Delta(G)$. Subsequently, all vertices that are not in $C$ are isolated. Thus, the MDG algorithm can now choose all vertices from $C$. Hence,

$$\textit{min-mdg}(H) = \tau(G) + \|E(G)\|(\Delta(G) + 1).$$

By Equation (6.4), *min-mdg*$(H) = \tau(H)$, so $H \in \mathcal{S}_1^{\mathrm{MDG}}$. ▮ (Lemma 6.4.1)

Lemma 6.4.2 below will be used in the proof of Theorem 6.4.4. The construction of the graph $G$ in this lemma is a modification of a construction given by Papadimitriou and Steiglitz [PS82, p. 408, Fig. 17-3], which shows that the worst-case approximation ratio of the MDG heuristic can be as bad as logarithmic in the input size, and so grows unboundedly. Similar constructions for achieving the worst-case approximation behavior of the greedy heuristic solving the more general minimum set cover problem were given by Johnson [Joh74], Lovász [Lov75], and Chvátal [Chv79].

**Lemma 6.4.2** *For all positive integers $n_1$, $n_2$, $\delta \geq 6$, and $\mu$ satisfying*

$$\mu(\ln \mu - 2\ln(\delta + 2) - 1) \geq n_1 + n_2, \tag{6.5}$$

*there exists a bipartite graph $G$ with the following properties:*

1. *$V(G) = V \cup \tilde{V}$ such that $V \cap \tilde{V} = \emptyset$ and both $V$ and $\tilde{V}$ are independent sets, where*

- *$V = \{u_1, u_2, \ldots, u_{n_1}, w_1, w_2, \ldots, w_\mu, z_1, z_2, \ldots z_{n_2}\}$ and*
- *$\tilde{V} = \{\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{n_1}, \tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_\mu\}$.*

*2. $\{\{u_i, \tilde{u}_i\} \mid 1 \le i \le n_1\} \cup \{\{w_i, \tilde{w}_i\} \mid 1 \le i \le \mu\} \subseteq E(G)$.*

*3. Every vertex $\tilde{u}_i$, where $1 \le i \le n_1$, has degree one.*

*4. For each induced subgraph $S$ of $G$ that can be obtained by deleting vertices from $V$ such that $V \cap V(S) \neq \emptyset$, it holds that*

$$\max_{v \in V \cap V(S)} \deg_S(v) > \max_{v \in \tilde{V}} \deg_S(v) + \delta.$$

**Proof** Let the constants $n_1$, $n_2$, $\delta$, and $\mu$ be given such that Equation (6.5) is satisfied. We describe the construction of the graph $G$. As stated in the lemma, the vertex set of $G$ is given by $V(G) = V \cup \tilde{V}$, where $V$ and $\tilde{V}$ are two disjoint independent sets.

Rename the vertices of $V$ by $V = \{\alpha_1, \alpha_2, \ldots, \alpha_{n_1+\mu+n_2}\}$.

Let $\tilde{W} = \{\tilde{w}_1, \tilde{w}_2, \tilde{w}_3 \ldots, \tilde{w}_\mu\}$. The edge set of $G$ is defined as follows:

- Create the edges $\{u_i, \tilde{u}_i\}$ for each $i$ with $1 \le i \le n_1$ and the edges $\{w_j, \tilde{w}_j\}$ for each $j$ with $1 \le j \le \mu$.
- Partition $\tilde{W}$ into $\lfloor \frac{\mu}{\delta+3} \rfloor$ disjoint sets $\tilde{W}_1^{\delta+3}, \tilde{W}_2^{\delta+3}, \ldots, \tilde{W}^{\delta+3}_{\lfloor \frac{\mu}{\delta+3} \rfloor}$ of size $\delta + 3$ each, possibly leaving out some vertices from $\tilde{W}$ and taking care that no vertex in $\tilde{W}_i^{\delta+3}$ already is connected with $\alpha_i$, $1 \le i \le \lfloor \frac{\mu}{\delta+3} \rfloor$. For each $i$ with $1 \le i \le \lfloor \frac{\mu}{\delta+3} \rfloor$, connect $\alpha_i$ with each vertex in $\tilde{W}_i^{\delta+3}$ by an edge.
- Partition $\tilde{W}$ into $\lfloor \frac{\mu}{\delta+4} \rfloor$ disjoint sets $\tilde{W}_1^{\delta+4}, \tilde{W}_2^{\delta+4}, \ldots, \tilde{W}^{\delta+4}_{\lfloor \frac{\mu}{\delta+4} \rfloor}$ of size $\delta + 4$ each, possibly leaving out some vertices from $\tilde{W}$ and taking care that no vertex in $\tilde{W}_i^{\delta+4}$ already is connected with $\alpha_{\lfloor \frac{\mu}{\delta+3} \rfloor + i}$, $1 \le i \le \lfloor \frac{\mu}{\delta+4} \rfloor$. For each $i$ with $1 \le i \le \lfloor \frac{\mu}{\delta+4} \rfloor$, connect $\alpha_{\lfloor \frac{\mu}{\delta+3} \rfloor + i}$ with each vertex in $\tilde{W}_i^{\delta+4}$ by an edge.
- Partition $\tilde{W}$ into $\lfloor \frac{\mu}{\delta+5} \rfloor$ disjoint sets $\tilde{W}_1^{\delta+5}, \tilde{W}_2^{\delta+5}, \ldots, \tilde{W}^{\delta+5}_{\lfloor \frac{\mu}{\delta+5} \rfloor}$ of size $\delta + 5$ each, possibly leaving out some vertices from $\tilde{W}$ and taking care that no vertex in $\tilde{W}_i^{\delta+5}$ already is connected with $\alpha_{\lfloor \frac{\mu}{\delta+3} \rfloor + \lfloor \frac{\mu}{\delta+4} \rfloor + i}$, $1 \le i \le \lfloor \frac{\mu}{\delta+5} \rfloor$. For each $i$ with $1 \le i \le \lfloor \frac{\mu}{\delta+5} \rfloor$, connect $\alpha_{\lfloor \frac{\mu}{\delta+3} \rfloor + \lfloor \frac{\mu}{\delta+4} \rfloor + i}$ with each vertex in $\tilde{W}_i^{\delta+5}$ by an edge.
- Continue in this way until all vertices $\alpha_i$ are connected with vertices in $\tilde{W}$.

The construction is possible, since Equation (6.5) implies

$$\left\lfloor\frac{\mu}{\delta+3}\right\rfloor + \left\lfloor\frac{\mu}{\delta+4}\right\rfloor + \cdots + \left\lfloor\frac{\mu}{\mu-1}\right\rfloor \geq n_1 + \mu + n_2, \tag{6.6}$$

and thus there are enough vertices in $\tilde{W}$. To see why, note that

$$\begin{aligned}
&\left\lfloor\frac{\mu}{\delta+3}\right\rfloor + \left\lfloor\frac{\mu}{\delta+4}\right\rfloor + \cdots + \left\lfloor\frac{\mu}{\mu-1}\right\rfloor \\
&= \left\lfloor\frac{\mu}{1}\right\rfloor + \left\lfloor\frac{\mu}{2}\right\rfloor + \cdots + \left\lfloor\frac{\mu}{\mu}\right\rfloor - 1 - \left(\left\lfloor\frac{\mu}{1}\right\rfloor + \left\lfloor\frac{\mu}{2}\right\rfloor + \cdots + \left\lfloor\frac{\mu}{\delta+2}\right\rfloor\right) \\
&\geq \frac{\mu}{1} + \frac{\mu}{2} + \cdots + \frac{\mu}{\mu} - \mu - \left(\frac{\mu}{1} + \frac{\mu}{2} + \cdots + \frac{\mu}{\delta+2}\right) \\
&= \mu H_\mu - \mu - \mu H_{\delta+2} \\
&\geq \mu \ln \mu - \mu - \mu \ln(\delta+2) - \mu \\
&\geq \mu \ln \mu - \mu \ln(\delta+2) - 2\mu \\
&\geq \mu \ln \mu - \mu \ln(\delta+2) - \mu \ln(\delta+2) \qquad (6.7) \\
&= \mu \ln \mu - 2\mu \ln(\delta+2).
\end{aligned}$$

Here, $H_k$ denotes the $k$th harmonic number, which is defined by $H_k = \sum_{i=1}^{k} \frac{1}{i}$. It is well known that for all $k$, $\ln k \leq H_k \leq \ln k + 1$ (see, e.g., Graham, Knuth, and Patashnik [GKP94]). Equation (6.7) holds because $\delta \geq 6$ and hence $\ln(\delta+2) \geq 2$.

It is evident from the construction that $G$ has all required properties. In particular, to see why Property 4 holds, let $S$ be any induced subgraph of $G$ that can be obtained by deleting vertices from $V$ such that $V \cap V(S) \neq \emptyset$. Let $y_S = \max_{v \in V \cap V(S)} \deg_S(v)$. By construction, $S$ can have only edges of the form $\{u_i, \tilde{u}_i\}$ or $\{w_j, \tilde{w}_j\}$ or edges that are added during the stages $\delta+3, \delta+4, \ldots, y_S$, where $\delta + i$ denotes the stage in which $\tilde{W}$ is partitioned into subsets of size $\delta + i$. It follows that

$$\max_{v \in \tilde{V}} \deg_S(v) \leq 1 + y_S - (\delta+3) + 1 = y_S - \delta - 1 < y_S - \delta,$$

which proves the lemma. ▮ (Lemma 6.4.2)

**Theorem 6.4.3** $\mathcal{S}_1^{\mathrm{MDG}}$ *is* $\mathrm{P}_{\parallel}^{\mathrm{NP}}$*-complete.*

**Proof** It is easy to see that $\mathcal{S}_1^{\mathrm{MDG}}$ is in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$. To prove $\mathrm{P}_{\parallel}^{\mathrm{NP}}$-hardness of $\mathcal{S}_1^{\mathrm{MDG}}$, let $A$ be an arbitrary set in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$, and let $f$ be the reduction from $A$ to `Minimum Vertex Cover Compare` stated in Theorem 3.3.4. For any string $x \in \Sigma^*$, let $f(x) = \langle G_2, G_1 \rangle$.

We will define a graph $\widehat{G}$ and an integer $q \geq 0$ such that:

$$\textit{min-mdg}(\widehat{G}) = \tau(G_2) + q; \tag{6.8}$$

$$\tau(\widehat{G}) = \tau(G_1) + q. \tag{6.9}$$

The reduction mapping any given string $x$ (via the pair $\langle G_2, G_1 \rangle$ obtained according to Theorem 3.3.4) to the graph $\widehat{G}$ such that Equations (6.8) and (6.9) are satisfied will establish that $X \leq_{\mathrm{m}}^{\mathrm{p}} \mathcal{S}_1^{\mathrm{MDG}}$. In particular, from these equations, we have that:

- $\tau(G_2) = \tau(G_1)$ implies $\mathit{min\text{-}mdg}(\widehat{G}) = \tau(\widehat{G})$, and
- $\tau(G_2) > \tau(G_1)$ implies $\mathit{min\text{-}mdg}(\widehat{G}) > \tau(\widehat{G})$.

Note that, due to Theorem 3.3.4, $\tau(G_2) \geq \tau(G_1)$.

We now describe the construction of $\widehat{G}$. Let $g$ be the reduction from Lemma 6.4.1 and let $H_2 = g(G_2)$. Thus, $H_2$ is in $\mathcal{S}_1^{\mathrm{MDG}}$ and, by Equation (6.4),

$$\tau(H_2) = \tau(G_2) + \|E(G_2)\|(\Delta(G_2) + 1). \tag{6.10}$$

Since one can add isolated vertices to any graph $G$ without affecting the values of $\tau(G)$ or $\mathit{min\text{-}mdg}(G)$, we may without loss of generality assume that

$$\|V(H_2)\| = \|V(G_1)\| + \|E(G_2)\|(\Delta(G_2) + 1). \tag{6.11}$$

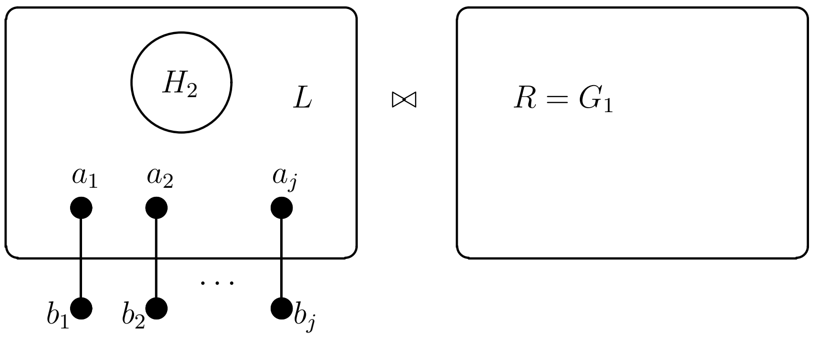

Figure 6.2: The graph $\widehat{G}$ constructed from $G_1$ and $H_2$.

Look at Figure 6.2 for the construction of $\widehat{G}$ from $G_1$ and $H_2$. The graph $\widehat{G}$ consists of two subgraphs, $L$ and $R$, that are joined by the join operation, plus some additional vertices and edges that are connected to $L$. Formally, choose $2j$ new vertices $a_i$ and $b_i$, $1 \leq i \leq j$, where $j$ is a fixed integer large enough such that the degree of each vertex in $R$ is larger than the maximum degree of the vertices in $L$. Note that the degree of each vertex in $R$ must remain larger than the degree of any vertex in $L$ even after some vertices have been removed from $R$.

Let $B$ be the bipartite matching with the vertex set

$$V(B) = \{a_i \mid 1 \leq i \leq j\} \cup \{b_i \mid 1 \leq i \leq j\}$$

and the edge set $E(B) = \{\{a_i, b_i\} \mid 1 \leq i \leq j\}$. Let $R = G_1$, and let $L$ be the graph with the vertex set $V(L) = \{a_i \mid 1 \leq i \leq j\} \cup V(H_2)$ and the edge set

$E(L) = E(H_2)$. The graph $\widehat{G}$ is defined by forming the join $L \bowtie R$, i.e., there are edges connecting each vertex of $L$ with each vertex of $R$, plus attaching the vertices $b_i$, $1 \leq i \leq j$, to $L$ by adding the $j$ edges from $E(B)$.

We first consider *min-mdg*$(\widehat{G})$. By our choice of $j$, each vertex in $R$ has a degree larger than the degree of any vertex not in $R$. Hence, on input $\widehat{G}$, the MDG algorithm first deletes all vertices from $R$. Subsequently, it can find a minimum vertex cover of $H_2$, which has size $\tau(G_2) + \|E(G_2)\|(\Delta(G_2)+1)$ by Equation (6.10), and eventually it can choose, say, the vertices $a_i$, $1 \leq i \leq j$, to cover the edges of $B$. Hence,

$$\begin{aligned} \textit{min-mdg}(\widehat{G}) &= \|V(G_1)\| + \tau(G_2) + \|E(G_2)\|(\Delta(G_2)+1) + j \\ &\overset{(6.11)}{=} \tau(G_2) + \|V(H_2)\| + j. \end{aligned}$$

We now consider $\tau(\widehat{G})$. Since every vertex cover of $\widehat{G}$ must contain all vertices of $L$ or all vertices of $R$ to cover the edges connecting $L$ and $R$, it follows from Equations (6.10) and (6.11) that:

$$\begin{aligned} \tau(\widehat{G}) &= \min\{\|V(G_1)\| + \tau(H_2) + j,\ \|V(H_2)\| + j + \tau(G_1)\} \\ &= \min\{\tau(G_2) + \|V(H_2)\| + j,\ \tau(G_1) + \|V(H_2)\| + j\}. \end{aligned}$$

Since $\tau(G_2) \geq \tau(G_1)$, it follows that

$$\tau(\widehat{G}) = \tau(G_1) + \|V(H_2)\| + j.$$

Hence, setting $q = \|V(H_2)\| + j$, Equations (6.8) and (6.9) are satisfied, which completes the proof that $\mathcal{S}_1^{\mathrm{MDG}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete. ▮ (Theorem 6.4.3)

**Theorem 6.4.4** *For each rational number $r > 1$, $\mathcal{S}_r^{\mathrm{MDG}}$ is $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.*

**Proof** Fix any rational number $r = \ell/m$, where $\ell$ and $m$ are integers with $1 \leq m < \ell$. Without loss of generality, we may assume that $\gcd(\ell - m, m) = 1$, where $\gcd(a, b)$ denotes the greatest common divisor of the integers $a$ and $b$. It is easy to see that $\mathcal{S}_r^{\mathrm{MDG}}$ is in $\mathrm{P}_{\|}^{\mathrm{NP}}$. To prove $\mathrm{P}_{\|}^{\mathrm{NP}}$-hardness of $\mathcal{S}_r^{\mathrm{MDG}}$, let $A$ be an arbitrary set in $\mathrm{P}_{\|}^{\mathrm{NP}}$, and let $f$ be the reduction from $A$ to `Minimum Vertex Cover Compare` stated in Theorem 3.3.4. For any string $x \in \Sigma^*$, let $f(x) = \langle G_2, G_1 \rangle$. Note that $\tau(G_2) \geq \tau(G_1)$.

We will define a graph $\widehat{G}_r$ and integers $p, q \geq 0$ such that:

$$\begin{aligned} \textit{min-mdg}(\widehat{G}_r) &= r(p \cdot \tau(G_2) + q); && (6.12) \\ \tau(\widehat{G}_r) &= p \cdot \tau(G_1) + q. && (6.13) \end{aligned}$$

The reduction mapping any given string $x$ (via the pair $\langle G_2, G_1 \rangle$ obtained according to Theorem 3.3.4) to the graph $\widehat{G}_r$ such that Equations (6.12) and (6.13) are satisfied will establish that $A \leq_{\mathrm{m}}^{\mathrm{p}} \mathcal{S}_r^{\mathrm{MDG}}$. In particular, from these equations, we have that:

- $\tau(G_2) = \tau(G_1)$ implies *min-mdg*$(\widehat{G}_r) = r \cdot \tau(\widehat{G}_r)$, and
- $\tau(G_2) > \tau(G_1)$ implies *min-mdg*$(\widehat{G}_r) > r \cdot \tau(\widehat{G}_r)$.

We now describe the construction of $\widehat{G}_r$:

- Let $g$ be the reduction from Lemma 6.4.1 and let $H_2 = g(G_2)$. Thus, $H_2 \in \mathcal{S}_1^{\mathrm{MDG}}$ and Equation (6.10) holds:

$$\tau(H_2) = \tau(G_2) + \|E(G_2)\|(\Delta(G_2) + 1).$$

- Let $G_1^1, G_1^2, \ldots, G_1^m$ be $m$ pairwise disjoint copies of $G_1$, and let $H_2^1$, $H_2^2$, …, $H_2^\ell$ be $\ell$ pairwise disjoint copies of $H_2$.
- Let $\tilde{U} = \bigcup_{i=1}^{\ell} H_2^i$ be the disjoint union of these copies of $H_2$, and rename the vertices of $\tilde{U}$ by $V(\tilde{U}) = \{\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{\ell\cdot\|V(H_2)\|}\}$.
- Let $Z = \bigcup_{i=1}^{m} G_1^i$ be the disjoint union of these copies of $G_1$, and rename the vertices of $Z$ by $V(Z) = \{z_1, z_2, \ldots, z_{m\cdot\|V(G_1)\|}\}$.
- To apply Lemma 6.4.2, choose $n_1 = \ell \cdot \|V(H_2)\|$, $n_2 \geq m \cdot \|V(G_1)\|$, and $\delta = \max\{6, \Delta(H_2) + 1\}$, where the exact value of $n_2$ will be specified below. Choose the constant $\mu$ so as to satisfy Equation (6.5):

$$\mu(\ln \mu - 2\ln(\delta + 2) - 1) \geq n_2 + n_1.$$

- Given the constants $n_1$, $n_2$, $\delta$, and $\mu$, define $\widehat{G}_r$ to be the bipartite graph $G$ from Lemma 6.4.2 extended by the edges between the $\tilde{u}_i$ vertices that were added above to represent the structure of the copies of $H_2$, and extended by the edges between the $z_j$ vertices that were added above to represent the structure of the copies of $G_1$. That is, unlike $G$, the graph $\widehat{G}_r$ is no longer a bipartite graph. Formally, the vertex set of $\widehat{G}_r$ is given by

$$\begin{array}{rcll} V(\widehat{G}_r) & = & V(G) = V \cup \tilde{V}, & \text{where} \\ V & = & \{u_1, u_2, \ldots, u_{n_1}, w_1, w_2, \ldots, w_\mu, z_1, z_2, \ldots z_{n_2}\} & \text{and} \\ \tilde{V} & = & \{\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{n_1}, \tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_\mu\}, & \end{array}$$

  and the edge set of $\widehat{G}_r$ is given by $E(\widehat{G}_r) = E(G) \cup E(\tilde{U}) \cup E(Z)$, where $E(G)$ is constructed as in the proof of Lemma 6.4.2.

This completes the construction of $\widehat{G}_r$. We now prove Equations (6.12) and (6.13).

1. We first consider *min-mdg*$(\widehat{G}_r)$. By construction, for each vertex $v$ in $\tilde{V}$, we have
$$\deg_{\widehat{G}_r}(v) \leq \deg_G(v) + \Delta(H_2) < \deg_G(v) + \delta. \tag{6.14}$$
Let $S$ be any induced subgraph of $\widehat{G}_r$ that can be obtained by deleting vertices from $V$ such that $V \cap V(S) \neq \emptyset$. Property 4 of Lemma 6.4.2 and Equation (6.14) imply that
$$\max_{v \in V \cap V(S)} \deg_S(v) > \max_{v \in \tilde{V}} \deg_S(v).$$
Hence, on input $\widehat{G}_r$, the MDG algorithm starts by choosing the $n_1 + \mu + n_2$ vertices from $V$, which isolates each vertex $\tilde{w}_i \in \tilde{V}$ and leaves $\ell$ isolated copies of $H_2$. Subsequently, since $H_2 \in \mathcal{S}_1^{\mathrm{MDG}}$, the MDG algorithm can choose a minimum vertex cover in each of these $\ell$ copies of $H_2$. By Equation (6.10),
$$\tau(H_2) = \tau(G_2) + \|E(G_2)\|(\Delta(G_2) + 1),$$
and hence,
$$\textit{min-mdg}(\widehat{G}_r) = n_1 + \mu + n_2 + \ell(\tau(G_2) + \|E(G_2)\|(\Delta(G_2) + 1)).$$

2. We now consider $\tau(\widehat{G}_r)$. Define the set $C = \tilde{V} \cup D$, where $D$ with $\|D\| = m \cdot \tau(G_1)$ is a minimum vertex cover of $Z$. It is obvious from the construction of $\widehat{G}_r$ that $C$ is a minimum vertex cover of $\widehat{G}_r$. Hence,
$$\tau(\widehat{G}_r) = n_1 + \mu + m \cdot \tau(G_1).$$

To complete the proof, we have to choose $n_2 \geq m \cdot \|V(G_1)\|$ such that Equations (6.12) and (6.13) are satisfied for suitable integers $p$ and $q$. Setting $p = m$ and $q = n_1 + \mu$ and requiring
$$n_1 + n_2 + \mu + \ell \cdot \|E(G_2)\|(\Delta(G_2) + 1) = r(n_1 + \mu) \tag{6.15}$$
or, equivalently,
$$m \cdot n_2 + m \cdot \ell \cdot \|E(G_2)\|(\Delta(G_2) + 1) = (\ell - m)n_1 + (\ell - m)\mu \tag{6.16}$$
satisfies Equations (6.12) and (6.13).

Our assumption that $\gcd(\ell - m, m) = 1$ implies that the equation
$$m \cdot n_2' + 1 = (\ell - m)\mu' \tag{6.17}$$
has integer solutions. Clearly, the set of solutions of Equation (6.17) depends only on the fixed rational number $r = \ell/m$. Fix one such solution $(n_2', \mu')$. Multiplying this solution with $m \cdot \ell \cdot \|E(G_2)\|(\Delta(G_2) + 1) - (\ell - m)n_1$, we obtain an integer

solution $(\widehat{n_2}, \widehat{\mu})$ for Equation (6.16). For every $k \in \mathbb{Z}$, $(n_2, \mu)$ is a solution of Equation (6.16), where

$$n_2 = \widehat{n_2} + k(\ell - m), \tag{6.18}$$

$$\mu = \widehat{\mu} + km. \tag{6.19}$$

Choosing $k$ large enough, we can make sure that (a) $n_2$ and $\mu$ are positive integers,[4] (b) $n_2 \geq m \cdot \|V(G_1)\|$, and (c) $n_2$ and $\mu$ satisfy Equation (6.5) for given $n_1$ and $\delta$. It is easy to see that $k$ can be small enough such that $n_2$ and $\mu$ are polynomially bounded in the size of the input of the reduction being described. This completes the proof of the theorem. ▮ (Theorem 6.4.4)

[4]Recall that $\ell - m > 0$ and $m > 0$.

# Chapter 7

# Counting Class Separations

In this chapter, we are concerned with some open problems of an influential paper by Fenner, Fortnow, and Kurtz [FFK94]. First, we give partial answers to the following questions by relativization:

1. Does it hold that $\mathrm{WPP}^{\mathrm{SPP}} = \mathrm{WPP}$?

2. Is WPP closed under polynomial-time Turing reductions?

3. Are the similar appearing classes LWPP and WPP equal?

Second, we prove that LWPP and WPP are not uniformly gap-definable.

Our proofs combine the well-known polynomial encoding technique with a new combinatorial property of low-degree multilinear polynomials (Key Lemma 7.3.2).

## 7.1 Counting Classes

We study complexity classes that are based on counting the number of accepting and rejecting computation paths of NPTMs. Valiant [Val79] introduced the famous class #P, which is the set of all functions that can be defined by the number of accepting paths of some NPTM. He proved several natural problems complete for #P, for example the problem of computing the permanent of a zero-one matrix. Fenner, Fortnow, and Kurtz [FFK94] generalized #P to GapP and developed a theory of gap-definable counting classes.[1] The class GapP is the set of functions that can be defined by the difference (the "*gap*") between the number of accepting and rejecting paths of an NPTM.

[1]Gupta [Gup95] defined independently the same class under the name $\mathcal{Z}\#\mathrm{P}$.

**Definition 7.1.1** *For any oracle NPTM $N$ and $A \subseteq \Sigma^*$, we define the following functions:*

1. *For every $x \in \Sigma^*$, $\#\mathrm{acc}_{N^A}(x)$ denotes the number of accepting paths of $N^A(x)$.*

2. *For every $x \in \Sigma^*$, $\#\mathrm{rej}_{N^A}(x)$ denotes the number of rejecting paths of $N^A(x)$.*

3. *For every $x \in \Sigma^*$, $\mathrm{gap}_{N^A}(x) = \#\mathrm{acc}_{N^A}(x) - \#\mathrm{rej}_{N^A}(x)$.*

We can now formally define the function classes #P and GapP.

**Definition 7.1.2** *1.* [Val79] $\#\mathrm{P} = \{g \mid (\exists\, \mathrm{NPTM}\, N)[g = \#\mathrm{acc}_N]\}$.

*2.* [FFK94, Gup95] $\mathrm{GapP} = \{g \mid (\exists\, \mathrm{NPTM}\, N)[g = \mathrm{gap}_N]\}$.

It is easy to show that every #P function is also a GapP function [FFK94].

Several important languages classes, including PP, $\oplus$P, $\mathrm{C}_=\mathrm{P}$, and UP, can be conveniently defined in terms of GapP or #P functions. Valiant [Val76] introduced the class UP of all NP languages that can be decided by NPTMs that never have more than one accepting path. With the help of #P functions we can define:

**Definition 7.1.3 ([Val76])** *The class* UP *is the set of all languages $L$ such that for some function $f \in \#\mathrm{P}$ and every $x \in \Sigma^*$:*

- $x \in L \Longrightarrow f(x) = 1.$
- $x \notin L \Longrightarrow f(x) = 0.$

We get the counting class SPP if we allow $f$ to be any function in the larger class GapP:

**Definition 7.1.4 ([FFK94, Gup95, OH93])** *The class* SPP *is the set of all languages $L$ such that for some function $f \in \mathrm{GapP}$ and every $x \in \Sigma^*$:*

- $x \in L \Longrightarrow f(x) = 1.$
- $x \notin L \Longrightarrow f(x) = 0.$

This class was independently introduced by Fenner, Fortnow, and Kurtz [FFK94] (under the name SPP), Gupta [Gup95] (under the name $\mathcal{Z}$UP), and Ogiwara and Hemachandra [OH93] (under the name XP). It can be considered as gap analog of the class UP. Fenner et al. [FFK94] also introduced the following generalizations of SPP.

**Definition 7.1.5 ([FFK94])**

1. *The class* WPP *is the set of all languages $L$ such that for some functions $f \in$ GapP and $g \in$ FP with $0 \notin \text{range}(g)$, and every $x \in \Sigma^*$:*
   - $x \in L \Longrightarrow f(x) = g(x)$.
   - $x \notin L \Longrightarrow f(x) = 0$.
2. *The class* LWPP *is the set of all languages $L$ such that for some functions $f \in$ GapP and $g \in$ FP with $0 \notin \text{range}(g)$, and every $x \in \Sigma^*$:*
   - $x \in L \Longrightarrow f(x) = g(0^{|x|})$.
   - $x \notin L \Longrightarrow f(x) = 0$.

LWPP is the restricted version of WPP where the function $g$ depends only on the *length* of $x$.

From the definitions, we get the following inclusion relation among the above defined classes:

$$\text{UP} \subseteq \text{SPP} \subseteq \text{LWPP} \subseteq \text{WPP}.$$

Although obviously UP $\subseteq$ NP, it is not clear whether SPP $\subseteq$ NP is true.

The class SPP is known to contain an important natural problem—the graph isomorphism problem [AK02]. Arvind and Vinodchandran [AV97] and Vinodchandran [Vin04] showed that many group-theoretic computational problems are in SPP or LWPP.

## 7.2 Preliminaries

For any set $X$ of variables, and for any multivariate polynomial $p \in \mathbb{Z}[X]$, $\deg(p)$ denotes the total degree of $p$. Polynomials bounding the running time of machines are monotonically increasing.

Recall that a computation path of an oracle NPTM $N$ encodes a complete valid computation of $N$ relative to some oracle, i.e., is a sequence of configurations including query strings and answers from the oracle. Given a computation path $\rho$, let $\text{sign}(\rho) = +1$ if $\rho$ is an accepting path, and let $\text{sign}(\rho) = -1$ if $\rho$ is a rejecting path.

In our proofs, we use an encoding of nondeterministic polynomial-time oracle Turing machines defined in terms of multilinear polynomials with integer coefficients over variables representing the oracle strings queried by the machine. See, e.g., the paper by de Graaf and Valiant [dGV02] for a similar approach. The formal description of the polynomial encoding is given below.

**Definition 7.2.1** *Let $N^{(\cdot)}(x)$ be a nondeterministic polynomial-time oracle Turing machine with running time $t(.)$ and input $x \in \Sigma^*$. Let $x_1, x_2, \dots, x_m$ be an enumeration of all strings in $\Sigma^*$ up to length $t(|x|)$.*

*A* polynomial encoding *of $N^{(\cdot)}(x)$ is a multilinear polynomial $p \in \mathbb{Z}[y_1, y_2, \dots, y_m]$ defined as follows: Call a computation path $\rho$* valid *if $\rho$ is a computation path of $N^D(x)$ for some oracle $D \subseteq \Sigma^*$. Let $x_{i_1}, x_{i_2}, \dots, x_{i_\ell}$ be the distinct queries along a valid computation path $\rho$. Create a monomial $\mathrm{mono}(\rho)$ that is the product of terms $z_{i_k}$, $k \in \{1, 2, \dots \ell\}$, where $z_{i_k} = y_{i_k}$ if $x_{i_k}$ is answered "yes" and $z_{i_k} = (1 - y_{i_k})$ if $x_{i_k}$ is answered "no" along $\rho$. Define*

$$p(y_1, y_2, \dots, y_m) = \sum_{\rho:\rho \text{ is valid}} \mathrm{sign}(\rho) \cdot \mathrm{mono}(\rho)$$

The next proposition states that the multilinear polynomial $p$ has low total degree, and contains all the necessary information about $N^{(\cdot)}(x)$ to yield the value $\mathrm{gap}_{N^{\mathcal{B}}}(x)$ for every oracle $\mathcal{B} \subseteq \Sigma^*$.

**Proposition 7.2.2** *The just defined polynomial $p(y_1, y_2, \dots, y_m)$ has the following properties:*

1. $\deg(p) \le t(|x|)$, *and*
2. *for all $\mathcal{B} \subseteq \Sigma^*$, $p(\chi_{\mathcal{B}}(x_1), \chi_{\mathcal{B}}(x_2), \dots, \chi_{\mathcal{B}}(x_m)) = \mathrm{gap}_{N^{\mathcal{B}}}(x)$.*

Lemma 7.2.3 states a variant of the prime number theorem. We will need it in our oracle constructions to estimate the number of primes between two integers.

**Lemma 7.2.3 ([RS62])** *For every $n \ge 17$, the number of primes less than or equal to $n$, $\pi(n)$, satisfies*

$$n/\ln n < \pi(n) < 1.25506\, n/\ln n.$$

## 7.3 The Key Lemma

In this section, we prove Lemma 7.3.2, which will be the main tool for the oracle constructions in the following sections.

First, we need the following lemma.

**Lemma 7.3.1** *Let $N, p \in \mathbb{N}$, where $1 < p \le N/2$. Let $s \in \mathbb{Q}[y_1, y_2, \dots, y_N]$ be a multilinear polynomial with rational coefficients, where each monomial has exactly $p-1$ different variables. Suppose that for some $val \in \mathbb{Q}$, it holds that $s(y_1, \dots, y_N) = val$ for every $y_1, y_2 \dots, y_N \in \{0, 1\}$ with $\sum_{i=1}^N y_i = p$. Then each monomial in $s(y_1, \dots, y_N)$ has the same rational coefficient, i.e.,*

$$s(y_1, y_2, \dots, y_N) = \sum_{1 \le i_1 < i_2 < \cdots < i_{p-1} \le N} (val/p) \cdot y_{i_1} y_{i_2} \cdots y_{i_{p-1}}.$$

**Proof of Lemma 7.3.1** Assume that the hypothesis of the lemma is true. For each $1 \leq i \leq N$, we identify variable $y_i$ by its index $i$ and identify a monomial $\prod_{j=1}^{k} y_{i_j}$ by the set of indices $\{i_1, \ldots, i_k\}$. Let $\mathcal{A}$ denote the collection of all subsets of $\{1, \ldots, N\}$ of size $p$ and let $\mathcal{B}$ denote the collection of all subsets of $\{1, \ldots, N\}$ of size $p-1$. W.l.o.g, we assume that the elements of $\mathcal{A}$ and $\mathcal{B}$ are ordered in an arbitrary but fixed manner. We use $m = \binom{N}{p}$ to denote the size of $\mathcal{A}$ and $n = \binom{N}{p-1}$ to denote the size of $\mathcal{B}$. Let $A_i$, $1 \leq i \leq m$, denote the $i$th element of $\mathcal{A}$, and $B_j$, $1 \leq j \leq n$, denote the $j$th element of $\mathcal{B}$. For a 2-dimensional matrix $M_{m\times n}$, let $\text{Row}(M, i)$, $1 \leq i \leq m$, denote the $i$th row of $M$.

The condition $s(y_1, \ldots, y_N) = val$ for every $y_1, \ldots, y_N \in \{0, 1\}$ with $\sum_{i=1}^{N} y_i = p$, as given in the hypothesis, can be expressed in terms of a matrix equation $M_{m\times n}X_{n\times 1} = b_{m\times 1}$. Here $M_{m\times n}$ is a 0-1 matrix whose $(i, j)$ entry, $M[i, j]$, is one if $A_i \supseteq B_j$ and is zero otherwise, $X_{n\times 1}$ is a column vector with the $j$th entry, $X[j]$, is a variable that denotes the coefficient of monomial $B_j$, and $b_{m\times 1}$ is a column vector with each entry $b_i$, $1 \leq i \leq m$, equals $val$. By assigning all coefficients $X[j]$ the value $val/p$, we obtain clearly a solution for the system of equations. Hence it is sufficient to prove that the solution is unique, i.e., $\text{rank}(M) = n$. We show that it is possible to express each canonical vector $e_i = [0, \ldots, 0, 1, 0 \ldots, 0]$, $1 \leq i \leq n$, as a linear combination of row vectors in $M$. W.l.o.g, we show that for vector $e_1 = [1, 0, \ldots, 0]$.

Form a matrix $\widehat{M}_{p\times n}$ in the following way. Row $k$, $1 \leq k \leq p$, of $\widehat{M}$ is the sum of all rows $i$ in $M$ with $||A_i \cap B_1|| = p - k$. Note that there is at least one row $i$ with $||A_i \cap B_1|| = p - k$. This follows from $||\{1, \ldots, N\} - B_1|| \geq p$, which is true because of the condition $p \leq N/2$.

**Claim 7.1** *The matrix $\widehat{M}$ has the following properties. For every row $k$ ($1 \leq k \leq p$),*

1. *$\widehat{M}[k, j_1] = \widehat{M}[k, j_2]$ whenever $||B_{j_1} \cap B_1|| = ||B_{j_2} \cap B_1||$,*
2. *$\widehat{M}[k, j] \neq 0$ for all $j$ with $||B_j \cap B_1|| = p - k$,*
3. *$\widehat{M}[k, j] = 0$ for all $j$ with $||B_j \cap B_1|| > p - k$.*

**Proof** To see (1), note that for fixed $k$, the cardinality of the set $\{A_i \in \mathcal{A} \,\big|\, ||A_i \cap B_1|| = p - k \wedge A_i \supseteq B_j\}$ depends only on the number of elements of $B_j \in \mathcal{B}$ that are also in $B_1$. Hence for every $j_1$ and $j_2$, with $||B_{j_1} \cap B_1|| = ||B_{j_2} \cap B_1||$, it holds that

$$\begin{aligned} &||\{A_i \in \mathcal{A} \,\big|\, ||A_i \cap B_1|| = p - k \wedge A_i \supseteq B_{j_1}\}|| \\ &\qquad\qquad = ||\{A_i \in \mathcal{A} \,\big|\, ||A_i \cap B_1|| = p - k \wedge A_i \supseteq B_{j_2}\}||. \end{aligned}$$

Statement (1) follows immediately. For the proof of (2), we have to verify that for every $k$ and $j$,

$$\mathcal{S} \overset{df}{=} \{A_i \in \mathcal{A} \,\big|\, ||A_i \cap B_1|| = p - k \wedge A_i \supseteq B_j\} \neq \emptyset \text{ if } ||B_j \cap B_1|| = p - k.$$

It is easy to see that, if $||B_j \cap B_1|| = p - k$ then $\mathcal{S}$ has as element any set $B_j \cup \{g\}$, where $g \notin B_1$. Finally, to show (3), note that for every $k$ and $j$,

$$||\{A_i \in \mathcal{A} \,\big|\, ||A_i \cap B_1|| = p - k \wedge A_i \supseteq B_j\}|| = 0 \text{ if } ||B_j \cap B_1|| \geq p - k + 1.$$

since $||B_j \cap B_1|| \geq p - k + 1$ and $A_i \supseteq B_j$ implies that $||A_i \cap B_1|| \geq p - k + 1$.
▮ (Claim 7.1)

To complete the proof of the lemma, we show that the structure of the matrix $\widehat{M}$ stated in Claim 7.1 implies that $e_1$ can be expressed as a linear combination of row vectors of $\widehat{M}$, and hence also as linear combination of row vectors of $M$. We construct a matrix $M'_{p\times p}$ from $\widehat{M}_{p\times n}$, which will turn out to have full rank. From Claim 7.1(1), we know that column $j_1$ and column $j_2$ of $\widehat{M}$ are equal whenever $||B_{j_1} \cap B_1|| = ||B_{j_2} \cap B_1||$. Thus it makes sense to define a matrix $M'$ eliminating all these duplicate columns from $\widehat{M}$. We define column $\ell$, $1 \leq \ell \leq p$, of $M'$ to equal column $j$ of $\widehat{M}$ for some $j$ with $||B_j \cap B_1|| = p - \ell$. Note that column 1 of $M'$ corresponds uniquely to column 1 of $\widehat{M}$. Claim 7.1(3) implies that the matrix $M'$ is an upper triangular matrix, and from Claim 7.1(2), it follows that all diagonal elements in $M'$ are $\neq 0$. Hence $M'$ has full rank. In particular, row vector $[1, 0, \ldots, 0]_{1\times p}$ can be written as a linear combination of rows in $M'$. Suppose

$$[1, 0, \ldots, 0]_{1\times p} = \sum_{k=1}^{p} c_k \cdot \mathrm{Row}(M', k).$$

for $c_1, \ldots, c_k \in \mathbb{Q}$. Then

$$e_1 = [1, 0, \ldots, 0]_{1\times n} = \sum_{k=1}^{p} c_k \cdot \mathrm{Row}(\widehat{M}, k),$$

because column 1 of $\widehat{M}$ equals column 1 of $M'$, and all other columns of $\widehat{M}$ equal a column $j$ in $M'$ with $p \geq j > 1$. ▮ (Lemma 7.3.1)

**Key Lemma 7.3.2** *Let $N, p \in \mathbb{N}$ be such that $p$ is a prime and $p \leq N/2$. Let $s \in \mathbb{Z}[y_1, y_2, \ldots, y_N]$ be a multilinear polynomial with total degree* $\deg(s) < p$. *If for some $val \in \mathbb{Z}$, it holds that*

1. *$s(0, 0, \ldots, 0) = 0$, and*
2. *$s(y_1, y_2, \ldots, y_N) = val$, for every $y_1, y_2, \ldots, y_N \in \{0, 1\}$ with $\sum_{i=1}^{N} y_i = p$,*

*then $p \mid val$.*

**Proof of Lemma 7.3.2** We transform $s(y_1, \ldots, y_N)$ to a multilinear polynomial $s'(y_1, \ldots, y_N)$ with the following properties:

**(1)** All monomials in $s'$ have exactly $p - 1$ different variables.

**(2)** $s'(y_1, \ldots, y_N) = s(y_1, \ldots, y_N) = val \in \mathbb{Z}$, for all $y_1, \ldots, y_N \in \{0, 1\}$ with $\sum_{i=1}^{N} y_i = p$.

**(3)** The coefficients of the monomials in $s'$ have the form $a/b$, where $a, b \in \mathbb{Z}$, and $p \nmid b$.

Since $s(0, \ldots, 0) = 0$, the polynomial $s$ has no monomial with degree 0. Let $t(y_1, \ldots, y_N) = a \prod_{i \in A} y_i$ be a monomial, where $A \subseteq \{1, \ldots, N\}$ and $1 \leq ||A|| = \ell < p - 1$. Define the multilinear polynomial $u_t$ by

$$u_t(y_1, \ldots, y_N) = \sum_{\substack{B \subseteq \{1, \ldots N\} - A, \\ ||A \cup B|| = p-1}} \left( \frac{a}{p - \ell} \prod_{i \in A \cup B} y_i \right). \tag{7.1}$$

**Claim 7.2** *For all $y_1, \ldots, y_N \in \{0, 1\}$ with $\sum_{i=1}^{N} y_i = p$, $u_t(y_1, \ldots, y_N) = t(y_1, \ldots, y_N)$.*

**Proof** Let $y_1, \ldots y_N \in \{0, 1\}$ be such that $\sum_{i=1}^{N} y_i = p$. Depending on the choice in the selection of $y_1, \ldots, y_N$, we have two cases.

**Case 1:** $t(y_1, \ldots, y_N) = 0$.
Then, clearly $u_t(y_1, \ldots, y_N) = t(y_1, \ldots, y_N) = 0$.

**Case 2:** $t(y_1, \ldots, y_N) = a$.
Then $y_i = 1$ for all $i \in A$. Let $D = \{i \,|\, i \in \{1, \ldots, N\} - A \wedge y_i = 1\}$. Clearly, $||D|| = p - \ell$. In the sum on the right hand side of Eq. (7.1), only $B$'s with $B \subseteq D$ contribute a value $\neq 0$. The sets $B$ have always cardinality $p - 1 - \ell$. Hence, there are exactly $\binom{p-\ell}{p-1-\ell} = p - \ell$ sets $B$ contributing the value $a/(p - \ell)$ to the sum.

■ (Claim 7.2)

Transform polynomial $s(y_1, \ldots, y_N)$ to polynomial $s'(y_1, \ldots, y_N)$ by substituting each monomial $t(y_1, \ldots, y_N)$ in $s(y_1, \ldots, y_N)$ of degree $< p-1$ by the corresponding polynomial $u_t(y_1, \ldots, y_N)$. Since $\{a/b \mid a, b \in \mathbb{Z} \land b \neq 0 \land p \nmid b\}$ is closed under addition, it follows that the polynomial $s'(y_1, \ldots, y_N)$ satisfies properties (1), (2), and (3) stated at the beginning of the proof. Lemma 7.3.1 implies that all the coefficients of monomials in $s'(y_1, \ldots, y_N)$ are equal to $val/p$. Thus to match with property (3) of polynomial $s'(y_1, \ldots, y_N)$, $val/p$ must be an integer. It follows that $p \mid val$. ▮ (Lemma 7.3.2)

## 7.4 Is WPP Closed Under Polynomial-Time Turing Reductions?

We now prove that, relative to an oracle, WPP is not closed under polynomial-time Turing reductions. Since LWPP is known to be closed under polynomial-time Turing reductions relative to every oracle, this provides also a relativized world where the similarly defined classes LWPP and WPP are not equal. Prior to this work there was no intuition against the possibility of a relativizable proof for the equality of these classes.

**Theorem 7.4.1** *There exists an oracle $A$ such that* $\mathrm{P}^{\mathrm{WPP}^A} \not\subseteq \mathrm{WPP}^A$.

**Proof** Recall that $\mathrm{pos}(z)$ denotes the number of strings of length $|z|$ that are lexicographically less than $z$. For every set $A \subseteq \Sigma^*$, $w \in \Sigma^*$, and $n \in \mathbb{N}$, we define "Witcount", "Promise" and "Boundary" as follows.

$$\begin{aligned}
&\mathrm{Witcount}(A, w) = ||\{x \in \Sigma^* \mid |x| = |w| \land wx \in A\}||,\\
&\mathrm{Promise}(A, n) \equiv (\forall w \in \Sigma^n)[\mathrm{Witcount}(A, w) = 0 \lor \mathrm{Witcount}(A, w) = \mathrm{pos}(w)] \land\\
&\qquad (\forall w_1, w_2 \in \Sigma^n)[\mathrm{pos}(w_1) \leq \mathrm{pos}(w_2) \land \mathrm{Witcount}(A, w_2) \neq 0 \Rightarrow\\
&\qquad\qquad \mathrm{Witcount}(A, w_1) \neq 0], \text{ and}\\
&\mathrm{Boundary}(A, n) = \max\{\mathrm{pos}(w) \mid |w| = n \land \mathrm{Witcount}(A, w) \neq 0\}.
\end{aligned}$$

For every set $A \subseteq \Sigma^*$, define $L_A$ as follows.

$$L_A = \{0^n \mid \mathrm{Boundary}(A, n) \equiv 1 \pmod{2}\}.$$

Clearly, if $A$ satisfies $\mathrm{Promise}(A, n)$ at each length $n$, then $L_A$ is in $\mathrm{P}^{\mathrm{WPP}^A}$ (using binary search along the strings $w$ with $|w| = n$).

We construct an oracle $A$ such that, for each $n$, $\mathrm{Promise}(A, n)$ is true, and $L_A \notin \mathrm{WPP}^A$. Let $(N_s, M_s, p_s)_{s \geq 1}$ be an enumeration of all triples such that $N_s$ is a nondeterministic polynomial-time oracle Turing machine, $M_s$ is a deterministic polynomial-time oracle transducer, $p_s$ is a polynomial, and the running time of both $N_s$ and $M_s$ is bounded by $p_s$ regardless of the oracle. The oracle $A$ is constructed in stages. In stage $s$, the membership in $A$ of strings of length $2n_s$ is decided, and the initial segment $A_{s-1}$ is extended to $A_s$. Our choice of $n_s$ guarantees that the oracle extension in stage $s$ does not affect the computation in earlier stages. Set $A_0 := \emptyset$ and $n_0 := 17$.

**Stage s where s $\geq$ 1 :** Let $n_s$ be large enough so that the previous stages are not affected and $2^{n_s} > 4n_s^2 p_s(n_s)$. We diagonalize against nondeterministic polynomial-time oracle Turing machine $N_s$ and deterministic polynomial-time oracle transducer $M_s$. Let *val* be the value computed by $M_s^{A_{s-1}}(0^{n_s})$. Because of the condition $0 \notin \text{range}(g)$ in the definition of LWPP, we can assume that $val \neq 0$. Let

$$T = \{w \in \Sigma^{2n_s} \mid M_s^{A_{s-1}}(0^{n_s}) \text{ queries } w\}.$$

($\star$) Choose a set $B$, $B \subseteq \overline{T} \cap \Sigma^{2n_s}$, satisfying $\text{Promise}(B, n_s)$ such that the following holds:

$$\begin{aligned} \text{Boundary}(B, n_s) \equiv 1 \pmod 2 \quad &\wedge \quad gap_{N_s^{A_{s-1} \cup B}}(0^{n_s}) \neq val, \text{ or} \\ \text{Boundary}(B, n_s) \equiv 0 \pmod 2 \quad &\wedge \quad gap_{N_s^{A_{s-1} \cup B}}(0^{n_s}) \neq 0. \end{aligned}$$

Let $A_s := A_{s-1} \cup B$.
**End of Stage s**

Clearly, the construction guarantees that $L_A \notin \text{WPP}^A$. The feasibility of the construction follows from the following claim.

**Claim 7.3** *For each $s \geq 1$, there exists an oracle extension $B$ satisfying ($\star$).*

**Proof** Suppose that in stage $s$ no set $B$ satisfying ($\star$) exists. Then, for every $B \subseteq \overline{T} \cap \Sigma^{2n_s}$ satisfying $\text{Promise}(B, n_s)$, the following hold.

$$\text{Boundary}(B, n_s) \equiv 1 \pmod 2 \implies gap_{N_s^{A_{s-1} \cup B}}(0^{n_s}) = val, \text{ and} \tag{7.2}$$

$$\text{Boundary}(B, n_s) \equiv 0 \pmod 2 \implies gap_{N_s^{A_{s-1} \cup B}}(0^{n_s}) = 0. \tag{7.3}$$

Let

$$U = \{w \in \Sigma^{n_s} \mid \text{pos}(w) \text{ is prime, and } 2^{n_s-2} \leq \text{pos}(w) \leq \frac{3}{2} \cdot 2^{n_s-2}\}.$$

Fix an arbitrary $w \in U$. Choose a set $C_w \subseteq \overline{T} \cap \Sigma^{2n_s}$ satisfying (a) $\text{Promise}(C_w, n_s)$, and (b) $\text{Boundary}(C_w, n_s) = \text{pos}(w) - 1$. Such a set $C_w$ always exists because $2^{n_s} - p_s(n_s) > \frac{3}{2} \cdot 2^{n_s-2}$. Statements (7.2) and (7.3) in particular imply that, for all $D_w \subseteq \overline{T} \cap w\Sigma^{n_s}$, it holds that (note that $\text{pos}(w)$ is odd)

$$\text{Witcount}(D_w, w) = 0 \implies \text{gap}_{N_s^{A_{s-1} \cup C_w \cup D_w}}(0^{n_s}) = 0, \text{ and} \tag{7.4}$$

$$\text{Witcount}(D_w, w) = \text{pos}(w) \implies \text{gap}_{N_s^{A_{s-1} \cup C_w \cup D_w}}(0^{n_s}) = val. \tag{7.5}$$

Let $s'_w \in \mathbb{Z}[y_1, y_2, \ldots, y_m]$ be the polynomial encoding of $N_s^{(\cdot)}(0^{n_s})$. W.l.o.g. assume that $x_1, x_2, \ldots, x_N$ enumerate the strings in $\overline{T} \cap w\Sigma^{n_s}$, and $x_{N+1}, x_{N+2}, \ldots, x_m$ enumerate the remaining strings up to length $p_s(n_s)$. From Proposition 7.2.2, it follows that the polynomial $s'_w(y_1, y_2, \ldots, y_m)$ has the following properties.

1. For all $D_w \subseteq \overline{T} \cap w\Sigma^{n_s}$, it holds that

$$s'_w(\chi_{D_w}(x_1), \chi_{D_w}(x_2), \ldots, \chi_{D_w}(x_N), \chi_{A_{s-1}\cup C_w}(x_{N+1}), \ldots, \chi_{A_{s-1}\cup C_w}(x_m)) = \mathrm{gap}_{N_s^{A_{s-1}\cup C_w \cup D_w}}(0^{n_s}). \quad (7.6)$$

2. $\deg(s'_w) \leq p_s(n_s) < 2^{n_s-2}/n_s^2 < \mathrm{pos}(w) < N/2$.

Note that the sets $A_{s-1} \cup C_w$ and $\overline{T} \cap w\Sigma^{n_s}$ are disjoint. The values $\chi_{A_{s-1}\cup C_w}(x_{N+1}), \ldots, \chi_{A_{s-1}\cup C_w}(x_m)$ do not depend on $D_w$. Define the new polynomial $s_w(y_1, y_2, \ldots, y_N)$ that has these values fixed:

$$s_w(y_1, y_2, \ldots, y_N) = s'_w(y_1, y_2, \ldots, y_N, \chi_{A_{s-1}\cup C_w}(x_{N+1}), \ldots, \chi_{A_{s-1}\cup C_w}(x_m)).$$

Hence $s_w$ satisfies

1. For all $D_w \subseteq \overline{T} \cap w\Sigma^{n_s}$, it holds that

$$s_w(\chi_{D_w}(x_1), \chi_{D_w}(x_2), \ldots, \chi_{D_w}(x_N)) = \mathrm{gap}_{N_s^{A_{s-1}\cup C_w \cup D_w}}(0^{n_s}). \quad (7.7)$$

2. $\deg(s_w) \leq \deg(s'_w) < N/2$.

Statements (7.4) and (7.5) respectively imply that

- $s_w(0, 0, \ldots, 0) = 0$, and
- for all $z_1, z_2, \ldots, z_N \in \{0, 1\}$ such that $\sum_{i=1}^{N} z_i = \mathrm{pos}(w)$, we have $s_w(z_1, z_2, \ldots, z_N) = val$.

It follows from Lemma 7.3.2 that $\mathrm{pos}(w) \mid val$.

Therefore, for each $w \in U$, $\mathrm{pos}(w) \mid val$. Hence,

$$val \geq \prod_{w\in U} \mathrm{pos}(w) \geq 2^{||U||} \geq 2^{\pi(\frac{3}{2}\cdot 2^{n_s-2}) - \pi(2^{n_s-2})} \geq 2^{2^{n_s-2}/n_s^2} > 2^{p_s(n_s)},$$

where the fourth inequality follows from Lemma 7.2.3 and the fifth inequality follows because, $2^{n_s} > 4n_s^2 p_s(n_s)$. However, $val \leq 2^{p_s(n_s)}$, because the running time of $M_s^{(\cdot)}(0^{n_s})$ is bounded by $p_s(n_s)$ regardless of the oracle. Thus, for each $s \geq 1$, $A_{s-1}$ can always be extended in stage $s$.

▮ (Claim 7.3 and Theorem 7.4.1)

As an immediate corollary of Theorem 7.4.1, and the fact that LWPP is closed under polynomial-time Turing reductions in all relativized worlds [FFK94], we get the following result.

**Corollary 7.4.2** *There exists an oracle $A$ such that* $\mathrm{WPP}^A \not\subseteq \mathrm{LWPP}^A$.

## 7.5 LWPP And WPP Are Not Uniformly Gap-Definable

Resolving an issue open since Fenner, Fortnow, and Kurtz raised it in 1994 [FFK94], we prove that LWPP as well as WPP are not uniformly gap-definable.

The notion of gap-definability was introduced by Fenner, Fortnow, and Kurtz [FFK94]. A *gap-definable* counting class is a collection of all sets such that, for any set in the class, the membership of a string in the set depends (in a way particular to the class) on the difference (gap) between the number of accepting and rejecting paths produced by some NPTM associated with the set.

**Definition 7.5.1 ([FFK94])** *A class $\mathcal{C}$ is* gap-definable *if there exist disjoint sets $A, R \subseteq \Sigma^* \times \mathbb{Z}$ such that, for any $L \subseteq \Sigma^*$, $L \in \mathcal{C}$ if and only if there exists an* NPTM *$N$ such that for all $x \in \Sigma^*$,*

$$\begin{aligned} x \in L &\implies (x, \mathrm{gap}_N(x)) \in A, \text{ and} \\ x \notin L &\implies (x, \mathrm{gap}_N(x)) \in R. \end{aligned}$$

*The class $\mathcal{C}$ is also denoted by Gap$(A, R)$.*

For relativizable classes, Fenner, Fortnow, and Kurtz [FFK94] introduced two ways of defining gap-definability: uniform and nonuniform. A relativizable class $\mathcal{C}$ is said to be *uniformly gap-definable* if it is gap-definable w.r.t. any oracle with a fixed (independent of the oracle) choice of $A$ and $R$. A relativizable class $\mathcal{C}$ is said to be *nonuniformly gap-definable* if it gap-definable w.r.t. an oracle where the choice of $A$ and $R$ is dependent on the oracle. Thus, the choice of $A$ and $R$ may vary with different oracles in case of nonuniform gap-definability. We now give a definition that expresses the oracle (in)dependence of the pair $(A, R)$ in the notion of gap-definability. In what follows, $(A, R)$ is called an accepting pair if $A, R \subseteq \Sigma^* \times \mathbb{Z}$ and $A \cap R = \emptyset$.

**Definition 7.5.2 ([FFK94])**

1. *A relativizable class $\mathcal{C}$ is* gap-definable relative to an oracle $\mathcal{O}$ with accepting pair $(A, R)$ *if for any $L \subseteq \Sigma^*$, $L \in \mathcal{C}^{\mathcal{O}}$ if and only if there exists an oracle* NPTM *$N$ such that for all $x \in \Sigma^*$,*

$$\begin{aligned} x \in L &\implies (x, \mathrm{gap}_{N^{\mathcal{O}}}(x)) \in A, \text{ and} \\ x \notin L &\implies (x, \mathrm{gap}_{N^{\mathcal{O}}}(x)) \in R. \end{aligned}$$

2. *A relativizable class $\mathcal{C}$ is* uniformly gap-definable with accepting pair $(A, R)$ *if for any oracle $\mathcal{O} \subseteq \Sigma^*$, it holds that $\mathcal{C}$ is gap-definable relative to $\mathcal{O}$ with accepting pair $(A, R)$.*

A class $\mathcal{D}$ is called *low* for a class $\mathcal{C}$ if and only if $\mathcal{C}^{\mathcal{D}} \subseteq \mathcal{C}$. Fenner et al. proved that SPP is low for GapP. This implies that SPP is low for every *uniformly* gap-definable counting class, such as PP, $\mathrm{C_{=}P}$, $\oplus$P, and SPP. It is easy to see that this result holds in every relativized world:

**Theorem 7.5.3 ([FFK94])** *If $\mathcal{C}$ is a uniformly gap-definable class, then for every $\mathcal{O} \subseteq \Sigma^*$, it holds that $\mathcal{C}^{\mathrm{SPP}^{\mathcal{O}}} = \mathcal{C}^{\mathcal{O}}$.*

In Theorem 7.5.5 below, we construct a relativized world in which $\mathrm{UP} \cap \mathrm{coUP}$ is not low for LWPP as well as for WPP. Since $\mathrm{UP} \cap \mathrm{coUP} \subseteq \mathrm{SPP}$ in every relativized world, this also shows that relative to the same oracle, SPP is not low for either of LWPP or WPP. Fenner, Fortnow, and Kurtz [FFK94] proved that both LWPP and WPP are nonuniformly gap-definable. However, they left open the question of whether LWPP and WPP are *uniformly* gap-definable. From Theorems 7.5.3 and 7.5.5, we can conclude that LWPP and WPP are not uniformly gap-definable.

First, we need the following definition.

**Definition 7.5.4**
1. *We will refer to any pair $(N^A, M^A)$, where $N$ is a nondeterministic polynomial-time oracle Turing machine, $M$ is a deterministic polynomial-time oracle transducer and $A \subseteq \Sigma^*$, as an* $\mathrm{LWPP}^A$ *pair.*

2. *If $(N^A, M^A)$ is an* $\mathrm{LWPP}^A$ *pair, then let*
$$L(N^A, M^A) \stackrel{\mathrm{df}}{=} \{x \in \Sigma^* \mid \mathrm{gap}_{N^A}(x) = g(0^{|x|})\},$$
*where $g$ is the function computed by transducer $M^A$.*

3. *We say that an* $\mathrm{LWPP}^A$ *pair $(N^A, M^A)$ is* valid*, if for each $x \in \Sigma^*$, $g(0^{|x|}) \neq 0$ and $\mathrm{gap}_{N^A}(x) \in \{0, g(0^{|x|})\}$, where $g$ is the function computed by transducer $M^A$.*

**Theorem 7.5.5** *There exists an oracle $\mathcal{A}$ such that* $\mathrm{LWPP}^{\mathrm{UP}^{\mathcal{A}} \cap \mathrm{coUP}^{\mathcal{A}}} \not\subseteq \mathrm{WPP}^{\mathcal{A}}$.[2]

**Proof** For any $B \subseteq \Sigma^*$, define the test language $L_B$ by
$$L_B = \{0^n \mid ||B^{=2n}|| \neq 0\}.$$

We put certain constraints on the set $B$ that guarantee $L_B$ to be in $\mathrm{LWPP}^{\mathrm{UP}^B \cap \mathrm{coUP}^B}$. For each $n \in \mathbb{N}$, we say that $B$ satisfies $\mathrm{Constraint}(B, n)$ if the following conditions hold:

**(a)** $B^{=2n+1} = \{0z\}$ for some $z \in \Sigma^{2n}$, and

**(b)** $B^{=2n+1} = \{0z\} \Longrightarrow ||B^{=2n}|| \in \{0, \mathrm{pos}(z)\}$.

**Claim 7.4** *If $B$ satisfies* $\mathrm{Constraint}(B, n)$ *at each length $n$, then $L_B$ is in* $\mathrm{LWPP}^{\mathrm{UP}^B \cap \mathrm{coUP}^B}$.

[2] It is easy to see that $\mathrm{LWPP}^{\mathrm{UP}^{\mathcal{A}} \cap \mathrm{coUP}^{\mathcal{A}}} = \mathrm{LWPP}^{(\mathrm{UP}^{\mathcal{A}} \cap \mathrm{coUP}^{\mathcal{A}}) \oplus \mathcal{A}}$.

**Proof** Let $B$ satisfy Constraint$(B, n)$ for every $n \in \mathbb{N}$. We will define $\mathcal{L} \subseteq \Sigma^*$ and oracle machines $\mathcal{N}$ and $\mathcal{M}$ that satisfy the following: (a) $\mathcal{L} \in \mathrm{UP}^B \cap \mathrm{coUP}^B$, (b) $(\mathcal{N}^{\mathcal{L}\oplus B}, \mathcal{M}^{\mathcal{L}\oplus B})$ is a valid $\mathrm{LWPP}^{\mathcal{L}\oplus B}$ pair, and (c) $L(\mathcal{N}^{\mathcal{L}\oplus B}, \mathcal{M}^{\mathcal{L}\oplus B}) = L_B$. This will show that $L_B$ is in $\mathrm{LWPP}^{\mathrm{UP}^B \cap \mathrm{coUP}^B}$. The set $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = \{x \mid |x| \text{ is odd and } (\exists x')\,[\mathrm{pos}(x) \leq \mathrm{pos}(x') \wedge x' \in B]\}.$$

It is easy to see that if $B$ satisfies Constraint$(B, n)$ for every $n \in \mathbb{N}$, then $\mathcal{L} \in \mathrm{UP}^B \cap \mathrm{coUP}^B$.

Let $\mathcal{N}'$ be a nondeterministic polynomial-time oracle Turing machine that, with access to the oracle $B$, on input $x$,

1. if $x \notin 0^*$ then rejects $x$, and
2. if $x \in 0^*$ then guesses a string $x'$ of length $2|x|$ and accepts $x'$ if and only if $x'$ is in $B$.

Since $\#\mathrm{P} \subseteq \mathrm{GapP}$ in every relativized world, there exists a nondeterministic polynomial-time oracle Turing machine $\mathcal{N}$ such that for all $\mathcal{O} \subseteq \Sigma^*$ and $x \in \Sigma^*$, $\mathrm{gap}_{\mathcal{N}^{\mathcal{O}}}(x) = \#\mathrm{acc}_{\mathcal{N}'^{\mathcal{O}}}(x)$. Finally, we define the deterministic polynomial-time oracle transducer $\mathcal{M}$ that, with access to the oracle $\mathcal{L} \oplus B$, on input $x$,

1. if $x \notin 0^*$ then outputs some nonzero value, say 1, and
2. if $x \in 0^*$ then performs a binary search for the unique string $0w$, where $|w| = 2|x|$, in $B$ by asking queries for the membership of strings of the form $0w'$, where $|w'| = 2|x|$, in $\mathcal{L}$. The machine $\mathcal{M}^{\mathcal{L}\oplus B}(0^n)$ finally outputs $\mathrm{pos}(w)$.

It can easily be verified that $(\mathcal{N}^{\mathcal{L}\oplus B}, \mathcal{M}^{\mathcal{L}\oplus B})$ is a valid $\mathrm{LWPP}^{\mathrm{UP}^B \cap \mathrm{coUP}^B}$ pair and $L(\mathcal{N}^{\mathcal{L}\oplus B}, \mathcal{M}^{\mathcal{L}\oplus B}) = L_B$. Thus the claim follows. ▮ (Claim 7.4)

We construct an oracle $\mathcal{A}$ such that, for each $n$, Constraint$(\mathcal{A}, n)$ is true, and $L_{\mathcal{A}} \notin \mathrm{WPP}^{\mathcal{A}}$. Let $(N_s, M_s, p_s)_{s\geq 1}$ be an enumeration of all triples such that $N_s$ is nondeterministic polynomial-time oracle Turing machine, $M_s$ is a deterministic polynomial-time oracle transducer, $p_s$ is a polynomial, and the running time of both $N_s$ and $M_s$ is bounded by $p_s$ regardless of the oracle. The oracle $\mathcal{A}$ is constructed in stages. In stage $s$, the membership in $\mathcal{A}$ of strings of length $2n_s$ and $2n_s + 1$ are decided and the partial oracle $\mathcal{A}_{s-1}$ is extended to $\mathcal{A}_s$. Our choice of $n_s$ guarantees that the oracle extension in stage $s$ does not affect the computation in earlier stages. Finally $\mathcal{A} := \lim_{n\to\infty} \mathcal{A}_n$. Let $\mathcal{A}_0 := \emptyset$ and $n_0 := 17$.

**Stage $s$, $s \geq 1$:** Choose $n_s$ large enough so that $2^{n_s} > 4n_s^2 p_s(n_s)$ and none of the strings of length $2n_s$ or more is queried by any machine in previous stages. We diagonalize against nondeterministic polynomial-time oracle Turing machine $N_s$ and deterministic polynomial-time oracle transducer $M_s$. Let

$$Y = \{0^{2m+1} \mid m \in \mathbb{N},\ 2m+1 \leq p_s(n_s),\ m \neq n_s,\ \text{and}\ \Sigma^{2m+1} \cap \mathcal{A}_{s-1} = \emptyset\}.$$

Let $val$ be the value computed by $M_s^{\mathcal{A}_{s-1} \cup Y}(0^{n_s})$. Because of the condition $0 \notin \text{range}(g)$ in the definition of LWPP, we can assume that $val$ is nonzero. Let

$$\begin{aligned} S = {} & \{w \mid w \in \Sigma^{2n_s} \text{ and } M_s^{\mathcal{A}_{s-1} \cup Y}(0^{n_s}) \text{ does not query } w\} \\ & \cup \{0w \mid w \in \Sigma^{2n_s} \text{ and } M_s^{\mathcal{A}_{s-1} \cup Y}(0^{n_s}) \text{ does not query } 0w\}. \end{aligned}$$

($\star\star$) Choose $B \subseteq S$ such that Constraint$(B, n_s)$ is true and the following hold.

$$\begin{aligned} ||B^{=2n_s}|| \neq 0 \quad &\wedge \quad \text{gap}_{N_s^{\mathcal{A}_{s-1} \cup Y \cup B}}(0^{n_s}) \neq val, \text{ or} \\ ||B^{=2n_s}|| = 0 \quad &\wedge \quad \text{gap}_{N_s^{\mathcal{A}_{s-1} \cup Y \cup B}}(0^{n_s}) \neq 0. \end{aligned}$$

We will show in Claim 7.5 that there is a set $B$ satisfying ($\star\star$). Let $\mathcal{A}_s := \mathcal{A}_{s-1} \cup Y \cup B$. Continue on to the next stage.
**End of Stage *s***

Clearly, the construction guarantees that Constraint$(\mathcal{A}, n)$ is true at each length $n$ (and hence $L_{\mathcal{A}} \in \text{LWPP}^{\text{UP}^{\mathcal{A}} \cap \text{coUP}^{\mathcal{A}}}$ by Claim 7.4) and $L_{\mathcal{A}} \notin \text{WPP}^{\mathcal{A}}$. Thus, it remains to show that a set $B$ satisfying ($\star\star$) always exists.

**Claim 7.5** *For every $s \geq 1$, there exists a set $B$ satisfying ($\star\star$).*

**Proof** Assume to the contrary that in some stage $s$, no set $B$ satisfying ($\star\star$) exists. Then, for every $B \subseteq S$ such that $B$ satisfies Constraint$(B, n_s)$, the following hold.

$$\begin{aligned} ||B^{=2n_s}|| \neq 0 \quad &\Longrightarrow \quad \text{gap}_{N_s^{\mathcal{A}_{s-1} \cup Y \cup B}}(0^{n_s}) = val, \text{ and} \\ ||B^{=2n_s}|| = 0 \quad &\Longrightarrow \quad \text{gap}_{N_s^{\mathcal{A}_{s-1} \cup Y \cup B}}(0^{n_s}) = 0. \end{aligned}$$

Let $U = \{z \in \Sigma^{2n_s} \mid \text{pos}(z) \text{ is prime},\ 0z \in S,\ \text{and}\ 2^{n_s-2} \leq \text{pos}(z) \leq 2^{n_s-1}\}$.

Fix an arbitrary element $z$ from $U$. Then, for all $C \subseteq \Sigma^{2n_s} \cap S$, it holds that

$$||C|| = \text{pos}(z) \quad \Longrightarrow \quad \text{gap}_{N_s^{\mathcal{A}_{s-1} \cup Y \cup C \cup \{0z\}}}(0^{n_s}) = val, \text{ and} \tag{7.8}$$

$$||C|| = 0 \quad \Longrightarrow \quad \text{gap}_{N_s^{\mathcal{A}_{s-1} \cup Y \cup C \cup \{0z\}}}(0^{n_s}) = 0. \tag{7.9}$$

Let $s'_z \in \mathbb{Z}[y_1, y_2, \ldots, y_m]$ be the polynomial encoding of $N_s^{(\cdot)}(0^{n_s})$. W.l.o.g. assume that $x_1, x_2, \ldots, x_N$ enumerate the strings in $\Sigma^{2n_s} \cap S$, and $x_{N+1}, x_{N+2}, \ldots, x_m$ enumerate the remaining strings up to length $p_s(n_s)$. From Proposition 7.2.2, it follows that the polynomial $s'_z(y_1, y_2, \ldots, y_m)$ has the following properties.

1. For all $C \subseteq \Sigma^{2n_s} \cap S$, it holds that

$$s'_z(\chi_C(x_1), \chi_C(x_2), \ldots, \chi_C(x_N), \chi_{\mathcal{A}_{s-1}\cup Y\cup\{0z\}}(x_{N+1}), \ldots, \chi_{\mathcal{A}_{s-1}\cup Y\cup\{0z\}}(x_m)) = \operatorname{gap}_{N_s^{\mathcal{A}_{s-1}\cup Y\cup C\cup\{0z\}}}(0^{n_s}). \quad (7.10)$$

2. $\deg(s'_z) \leq p_s(n_s) < 2^{n_s-2}/n_s^2 < \operatorname{pos}(z) < N/2$.

Note that the sets $\mathcal{A}_{s-1} \cup Y \cup \{0z\}$ and $\Sigma^{2n_s} \cap S$ are disjoint. The values $\chi_{\mathcal{A}_{s-1}\cup Y\cup\{0z\}}(x_{N+1}), \ldots, \chi_{\mathcal{A}_{s-1}\cup Y\cup\{0z\}}(x_m))$ do not depend on $C$. Define the new polynomial $s_z(y_1, y_2, \ldots, y_N)$ that has these values fixed:

$$s_z(y_1, y_2, \ldots, y_N) = s'_z(y_1, y_2, \ldots, y_N, \chi_{\mathcal{A}_{s-1}\cup Y\cup\{0z\}}(x_{N+1}), \ldots, \chi_{\mathcal{A}_{s-1}\cup Y\cup\{0z\}}(x_m)).$$

Hence $s_z$ satisfies

1. For all $C \subseteq \Sigma^{2n_s} \cap S$, it holds that

$$s_z(\chi_C(x_1), \chi_C(x_2), \ldots, \chi_C(x_N)) = \operatorname{gap}_{N_s^{\mathcal{A}_{s-1}\cup Y\cup C\cup\{0z\}}}(0^{n_s}). \quad (7.11)$$

2. $\deg(s_z) \leq \deg(s'_z) < N/2$.

Statements (7.8) and (7.9) respectively imply that

- for all $y_1, y_2, \ldots, y_N \in \{0, 1\}$ such that $\sum_{i=1}^{N} y_i = \operatorname{pos}(z)$, we have $s_z(y_1, y_2, \ldots, y_N) = val$, and
- $s_z(0, 0, \ldots, 0) = 0$.

It follows from Lemma 7.3.2 that $\operatorname{pos}(z) \mid val$.

Therefore, for each $z \in U$, $\operatorname{pos}(z) \mid val$. Hence by Lemma 7.2.3 and the fact that $2^{n_s} > 4n_s^2 p_s(n_s)$, $val \geq \prod_{z\in U} \operatorname{pos}(z) \geq 2^{||U||} \geq 2^{\pi(2^{n_s-1})-\pi(2^{n_s-2})-p_s(n_s)}$ $\geq 2^{2^{n_s-1}/n_s^2-p_s(n_s)} > 2^{p_s(n_s)}$. However, $M_s^{(\cdot)}(0^{n_s})$ runs in time $p_s(n_s)$ and so $val \leq 2^{p_s(n_s)}$. Thus, we have a contradiction. ▮ (Claim 7.5 and Theorem 7.5.5)

**Corollary 7.5.6** LWPP *and* WPP *are not uniformly gap-definable.*

# Bibliography

[AK02] V. Arvind and P. Kurur. Graph isomorphism is in SPP. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 743–750, Los Alamitos, November 16–19 2002. IEEE Computer Society.

[ALR04] E. Allender, M. Loui, and K. Regan. Complexity theory. In A. Tucker, editor, *Computer Science Handbook*, pages 5/1–5/30. CRC Press, 2004.

[Arr63] K. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, 1951 (revised edition, 1963).

[AV97] V. Arvind and N. Vinodchandran. Solvable black-box group problems are low for PP. *Theoretical Computer Science*, 180(1-2):17–45, 1997.

[BC93] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.

[BH91] S. Buss and L. Hay. On truth-table reducibility to SAT. *Information and Computation*, 91(1):86–102, 1991.

[Bla58] D. Black. *Theory of Committees and Elections*. Cambridge University Press, 1958.

[BTT89a] J. Bartholdi III, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.

[BTT89b] J. Bartholdi III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[BTT92] J. Bartholdi III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical Comput. Modelling*, 16(8/9):27–40, 1992.

[BTY97] H. Bodlaender, D. Thilikos, and K. Yamazaki. It is hard to know when greedy is good for finding independent sets. *Information Processing Letters*, 61:101–106, 1997.

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[CLRS01] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[Con85] M. J. A. N. de Caritat, Marquis de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. 1785. Facsimile reprint of original published in Paris, 1972, by the Imprimerie Royale. English translation appears in I. McLean and A. Urken, *Classics of Social Choice*, University of Michigan Press, 1995, pages 91–112.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158. ACM Press, 1971.

[dGV02] M. de Graaf and P. Valiant. Comparing EQP and $\mathrm{MOD}_{p^k}\mathrm{P}$ using polynomial degree lower bounds. Technical Report quant-ph/0211179, Quantum Physics, 2002.

[Dod76] C. Dodgson. A method of taking votes on more than two issues, 1876. Pamphlet printed by the Clarendon Press, Oxford, and headed "not yet published" (see the discussions in [MU95, Bla58], both of which reprint this paper).

[DS02] I. Dinur and S. Safra. The importance of being biased. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 33–42, 2002. To appear in *Annals of Mathematics*.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[Fis77] P. Fishburn. Condorcet social choice functions. *SIAM Journal on Applied Mathematics*, 33:469–489, 1977.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[GKP94] R. Graham, D. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1994.

[Gup95] S. Gupta. Closure properties and witness reduction. *Journal of Computer and System Sciences*, 50(3):412–432, 1995.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[HHR97a] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

[HHR97b] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Raising NP lower bounds to parallel NP lower bounds. *SIGACT News*, 28(2):2–13, 1997.

[HO02] L. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion.* Springer, 2002.

[HR98] E. Hemaspaandra and J. Rothe. Recognizing when greed can approximate maximum independent sets is complete for parallel access to NP. *Information Processing Letters*, 65(3):151–156, 1998.

[HRS] E. Hemaspaandra, J. Rothe, and H. Spakowski. Recognizing when heuristics can approximate minimum vertex covers is complete for parallel access to NP. *RAIRO Theoretical Informatics and Applications.* To appear.

[HRS02] E. Hemaspaandra, J. Rothe, and H. Spakowski. Recognizing when heuristics can approximate minimum vertex covers is complete for parallel access to NP. In *Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2002)*, pages 258–269. Springer-Verlag *Lecture Notes in Computer Science #2573*, 2002.

[HSV] E. Hemaspaandra, H. Spakowski, and J. Vogel. The complexity of Kemeny elections. *Theoretical Computer Science.* Accepted subject to minor revision.

[Joh74] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[Kad89] J. Kadin. $\mathrm{P}^{\mathrm{NP}[\log n]}$ and sparse Turing-complete sets for NP. *Journal of Computer and System Sciences*, 39(3):282–298, 1989.

[Kar72] R. Karp. Reducibilities among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, 1972.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[Kar04] G. Karakostas. A better approximation ratio for the vertex cover problem. Technical Report 04-084, Electronic Colloquium on Computational Complexity, `http://www.eccc.uni-trier.de/eccc/`, October 2004.

[Kem59] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959.

[Kha79] L. Khachiyan. A polynomial algorithm for linear programming. *Soviet Math. Dokl.*, 20:191–194, 1979.

[Kre88] M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *RAIRO Theoretical Informatics and Applications*, 21:419–435, 1987.

[Len83] H. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[Lev73] L. Levin. Universal search problems. *Problems of Information Transmission*, 9:265–266, 1973.

[Lev75] A. Levenglick. Fair and reasonable election systems. *Behavioral Science*, 20:34–46, 1975.

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[McG53] D. McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, 21:608–610, 1953.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[MU95] I. McLean and A. Urken. *Classics of Social Choice*. University of Michigan Press, 1995.

[OH93] M. Ogiwara and L. Hemachandra. A complexity theory for feasible closure properties. *Journal of Computer and System Sciences*, 46(3):295–325, 1993.

[Pap94] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[PS82] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings 6th GI Conference on Theoretical Computer Science*, pages 269–276. Springer-Verlag *Lecture Notes in Computer Science #145*, 1983.

[Rot05] J. Rothe. *Complexity Theory and Cryptology. An Introduction to Cryptocomplexity.* Springer-Verlag, Berlin, Heidelberg, New York, 2005. To appear.

[RS62] J. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6:64–94, 1962.

[RSV02] J. Rothe, H. Spakowski, and J. Vogel. Exact complexity of exact-four-colorability and of the winner problem for Young elections. In *Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science*, pages 310–322. Kluwer Academic Publishers, August 2002.

[RSV03] J. Rothe, H. Spakowski, and J. Vogel. Exact complexity of the winner problem for Young elections. *Theory of Computing Systems*, 36(4):375–386, 2003.

[Sch01] U. Schöning. *Theoretische Informatik—kurzgefasst.* Spektrum Akademischer Verlag, 2001.

[ST04] H. Spakowski and R. Tripathi. Degree bounds on polynomials and relativization theory. In *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science*, pages 105–118. Kluwer Academic Publishers, August 2004.

[Sto76] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[STT] H. Spakowski, M. Thakur, and R. Tripathi. Quantum and classical complexity classes: Separations, collapses, and closure properties. *Information and Computation.* To appear. Online version: DOI 10.1016/j.ic.2004.10.009.

[STT03] H. Spakowski, M. Thakur, and R. Tripathi. Quantum and classical complexity classes: Separations, collapses, and closure properties. In *Proceedings of the 23rd Conference on FSTTCS*, pages 375–386.

Springer-Verlag *Lecture Notes in Computer Science #2914*, December 2003.

[SV00] H. Spakowski and J. Vogel. $\Theta_2^p$-completeness: A classical approach for new results. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 348–360. Springer-Verlag *Lecture Notes in Computer Science #1974*, December 2000.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5(1):20–23, 1976.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[Vin04] N. Vinodchandran. Counting complexity of solvable black-box group problems. *SIAM Journal on Computing*, 33(4):852–869, 2004.

[Wag87] K. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51(1–2):53–80, 1987.

[Wag90] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[YL78] H. Young and A. Levenglick. A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35:285–300, 1978.

[You77] H. Young. Extending condorcet's rule. *Journal of Economic Theory*, 16:335–353, 1977.

# Index