Partitioning Sparse Graphs Into Triangles

Relations to exact satisfiability and very fast exponential time algorithms

Johan M. M. van Rooij

Marcel E. van Kooten Niekerk

Hans L. Bodlaender

Technical Report UU-CS-2010-005
Januari 2010

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Partitioning Sparse Graphs Into Triangles

Relations to exact satisfiability and very fast exponential time algorithms

Johan M. M. van Rooij Marcel E. van Kooten Niekerk
Hans L. Bodlaender

Department of Information and Computing Sciences, Utrecht University,
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands
jmmrooij@cs.uu.nl, markonie@xs4all.nl, hansb@cs.uu.nl

February 17, 2010

Abstract

We consider the problem of partitioning bounded degree graphs into triangles. We show that this problem is polynomial time solvable on graphs of maximum degree three by giving a linear time algorithm. We also show that this problem becomes $\mathcal{NP}$-complete on graphs of maximum degree four. Moreover, we show that there is no subexponential time algorithm for this problem on maximum degree four graphs unless the Exponential Time Hypothesis fails. However, the partition into triangles problem for graphs of maximum degree at most four is in many cases practically solvable as we give an algorithm for this problem that runs in $\mathcal{O}(1.0222^n)$ time and linear space. We note that the running time of this algorithm does not involve any large hidden polynomial factors.

1 Introduction

In his weblog of February 2009 [9], R. J. Lipton quotes Alan J. Perlis, the first Turing Award winner:

> For every polynomial-time algorithm you have, there is an exponential time algorithm that I would rather run.

His point is simple: if your algorithm runs in n^4 time, then an algorithm that runs in $n2^{n/10}$ time (alternatively denoted as $n1.07178^n$ time) is faster if for example $n = 100$ (this holds for all $n \leq 236$).

The same observation for $\mathcal{NP}$-hard problems in stead of polynomial time solvable problems was made by Woeginger in his well known survey on exact exponential time algorithms [11]. Woeginger considers the fact that algorithms for $\mathcal{NP}$-hard problems with exponential running times may actually lead to practical algorithms: he compares $\mathcal{O}(n^4)$ with $\mathcal{O}(1.01^n)$. We, however, are not aware of any papers on natural[1] $\mathcal{NP}$-hard problems with exponential time algorithms with running times anywhere near $\mathcal{O}(1.01^n)$ without involving huge polynomial factors (either visible, hidden in the notation, or hidden in the decimal rounding of the exponent in the big-$\mathcal{O}$). In this paper, we will give such an algorithm running in time $\mathcal{O}(1.0222^n)$ or $\mathcal{O}(2^{n/31.58})$ for the Partition Into Triangles problem restricted to maximum degree four graphs.

The Partition Into Triangles problem is one of the classical $\mathcal{NP}$-complete problems [4]. In this paper, we study this problem restricted to bounded degree graphs and obtain a series of results. On graphs of maximum degree three, we show that the problem is linear time solvable. On graphs of maximum degree four, we show that there exists a strong and interesting relation

[1] I.e., without making artificial constructions like Independent Set restricted to graphs in which 99% of the vertices have degree at most two.

between Partition Into Triangles and the Exact 3-Satisfiability problem. We exploit this relation is in several ways. First, we use it to show that the Partition Into Triangles problem becomes $\mathcal{NP}$-complete on graphs of maximum degree four. Second, we use it to show that there exists no subexponential time algorithm for Partition Into Triangles on maximum degree four graphs unless the Exponential Time Hypothesis [5] fails. Thus it seems that Partition Into Triangles restricted to graphs of maximum degree four is a hard problem. However, as a third application of the relation to Exact 3-Satisfiability, we give an $\mathcal{O}(1.0222^n)$ time algorithm for this problem by presenting an algorithm for Exact Satisfiability that is specially tailored to inputs obtained from Partition Into Triangles instances of maximum degree four. The running time of this algorithm involves no large hidden polynomial factors which makes it effective in practice. On these instances, it is significantly faster than the result of applying any of the fastest known algorithms for Exact Satisfiability or Exact 3-Satisfiability [3].

On general graphs, the Partition Into Triangles problem can be solved using set partitioning via inclusion-exclusion [2] in $\mathcal{O}(2^n n^{\mathcal{O}(1)})$ time and polynomial space. This can be improved as a side result of two recent papers. Koivisto [7] has given a general covering algorithm that can be used to solve the problem in $\mathcal{O}(1.7693^n)$ time and space. And, Björklund [1] has given a general randomised partitioning algorithm that can be used to solve the problem in $\mathcal{O}(1.496^n)$ time and polynomial space while having a probability of failure which is exponentially small in n. On bounded degree graphs, we are unfamiliar with any results besides Kann who proved that the optimisation variant (cover by maximum number of triangles) is *Max-*$\mathcal{SNP}$-complete on graphs of maximum degree at least six [6].

Our paper is organised as follows. After some preliminaries, we give a simple linear time algorithm for Partition Into Triangles on graphs of maximum degree three in Section 3. Then, we look at the relation between the problem on graphs of maximum degree four and Exact 3-Satisfiability in Section 4. We use this to show $\mathcal{NP}$-completeness of the problem in Section 5. In this section, we also show that there is no subexponential time algorithms for the problem unless the Exponential Time Hypothesis fails. We give our fast exponential time algorithm in Section 6. Finally, some conclusions are given in Section 7.

2 Preliminaries

Let $G = (V, E)$ be a simple n-vertex graph. The *degree* of a vertex $v \in V$ is its number of neighbours in G: $d(v) = |\{u \in V \mid (u, v) \in E\}|$. A *$r$-regular graph* is a graph in which all vertices have degree r; a *cubic graph* is a 3-regular graph. The *(closed) neighbourhood* of a vertex $N[v]$ is the set of vertices at distance at most one from v: $N[v] = \{v\} \cup \{u \in V \mid (u, v) \in E\}$. In this paper, we will use the term *local neighbourhood* of a vertex v referring to the graph induced by $N[v]$, i.e, the graph $H = (N[v], E \cap (N[v] \times N[v]))$ where $N[v]$ is taken in G.

A *triangle* is a collection of three vertices in G in which each pair is joined by an edge. A *triangle partition* of G is a partitioning of V in $n/3$ disjoint subsets such that each such subset forms a triangle. This paper considers the problem Partition Into Triangles: given a graph G, does G have a triangle partition?

A *literal* is a variable x or its negation $\neg x$. We will often use reasoning involving the Exact Satisfiability problem (XSAT). In this problem, we are given a set of variables X and a set of clauses $\mathcal{C}$ containing literals of the set of variables X; we have to decide whether there exist a truth assignment of the variables such that each clause contains *exactly* one literal that is true. The Exact 3-Satisfiability problem (X3SAT) is the Exact Satisfiability problem with the restriction that all clauses have size at most three.

The 3-Satisfiability problem (3SAT) is also used: given a set of variables X and a set of clauses $\mathcal{C}$ of size at most three, does there exist a truth assignment of the variables such that each clause contains *at least* one literal that is true. Unless stated otherwise, any given clause is considered to be a clause of an *exact* (3-)satisfiability problem instance. When there is the possibility of confusion, we denote a 3SAT clause by $\mathrm{SAT}(x, y, z)$ and an X3SAT clause by $\mathrm{XSAT}(x, y, z)$.

Algorithm 1 A linear time algorithm for graphs of maximum degree three.

Input: A graph $G = (V, E)$ of maximum degree three.
Output: A triangle partition T of G or NO if no such partitioning exists.

```
1: if |V| is not a multiple of three then return NO
2: while G is non-empty do
3:     Take any vertex v ∈ V.
4:     if N[v] contains a vertex of degree at most two then
5:         Reduce the graph using Lemma 1. If a triangle is selected, then add it to T.
6:     else if N[v] corresponds to cases 1, 3, or 4 of Figure 1 then
7:         return NO
8:     else //case 2 of Figure 1
9:         Add the triangle in N[v] to T and remove its vertices from G.
10: return T
```

The number of positive literals of a variable $x \in X$ is denoted by $f_+(x)$ and the number of negative literals by $f_-(x)$. The *frequency* $f(x)$ of a variable $x \in X$ is its total number of occurrences in a problem instance: $f(x) = f_+(x) + f_-(x)$. We also use the notation $F(x)$ for the tuple $F(x) = (f_+(x), f_-(x))$. A *unique* variable is a variable of frequency one.

When an algorithm repeatedly branches on an instance of size n obtaining subproblems of sizes $n - r_1, n - r_2, \ldots, n - r_l$, then the algorithm generates at most $\tau(r_1, r_2, \ldots, r_l)^n$ subproblems in total. Here, $\tau(r_1, r_2, \ldots, r_l)$ is called the *branching number*. It equals the smallest positive real root of the equation $1 = x^{-r_1} + x^{-r_2} + \ldots + x^{-r_l}$. When an algorithm has multiple branching rules, then at most τ^n subproblems are generated, where τ is the maximum over the branching numbers of all its branching rules. For more details on branching numbers, see [8].

The Exponential Time Hypothesis (ETH) [5] is the complexity theoretical assumption that there is no algorithm solving the 3SAT problem on n variables in $\mathcal{O}(2^{\epsilon n})$ time, for all $\epsilon > 0$.

3 A Linear Time Algorithm on Graphs of Maximum Degree Three

We begin by considering PARTITION INTO TRIANGLES on graphs of maximum degree three. We will prove that this problem is polynomial time solvable on this graph class by giving a linear time algorithm: Algorithm 1.

Lemma 1 *Let $G = (V, E)$ be an instance of* PARTITION INTO TRIANGLES *restricted to graphs of maximum degree d containing a vertex v of degree at most two. In constant time, we can either decide that G is a* NO*-instance, or we can transform G into an equivalent smaller instance.*

Proof: If v has degree at most one, then this vertex cannot be in any triangle and the instance is a NO-instance. Otherwise, v has degree two; let u, w be the neighbours of v. As G is of constant maximum degree, we can test in constant time whether $(u, w) \in E$. If $(u, w) \in E$, then $\{u, v, w\}$ is

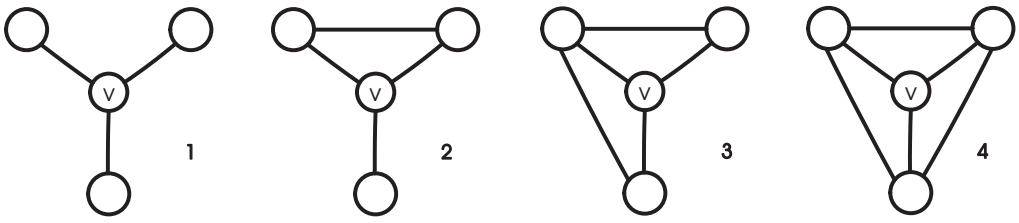

Figure 1: Possible edges within the local neighbourhood of a vertex in a cubic graph.

the unique triangle containing v, and we remove this triangle from G to obtain a smaller equivalent instance. If $(u, w) \notin E$, then v is not part of any triangle, and we again have a No-instance. □

Theorem 1 *Algorithm 1 solves* Partition Into Triangles *on graphs of maximum degree three in linear time.*

Proof: For correctness, we note that the number of vertices must be a multiple three in order to partition G into triangles. Furthermore, correctness of the first case in the main loop follows from Lemma 1. For the other two cases, we observe that any local neighbourhood of v must equal one of the four cases in Figure 1. In case 1, no triangle containing v exists, and, in cases 3 and 4, the fact that G is cubic results in that removing any triangle would lead to vertices of degree at most 1 which can no longer be in a triangle. Hence, these are all No-instances. In case 2, v can only be part of one triangle which Algorithm 1 selects. We conclude that the algorithm is correct.

Each iteration of the main loop requires constant time, since inspecting a neighbourhood in a cubic graph can be done in constant time. In each iteration, Algorithm 1 either terminates, or removes three vertices from G. Hence, there are at most a linear number of iterations. We conclude that Algorithm 1 runs in linear time. □

4 The Relation Between Partition Into Triangles on Graphs of Maximum Degree Four and Exact 3-Satisfiability

When we restrict the Partition Into Triangles problem to graphs of maximum degree four, an interesting relation with Exact 3-Satisfiability emerges. This relation will be the topic of this section.

We will give three lemmas similar to Lemma 1 and Theorem 1 that allow us to either decide that an instances is a No-instance, or that it can be reduced to an equivalent smaller instance. These lemmas will apply to any instance of Partition Into Triangles on maximum degree four graphs unless all vertices in the instance have a local neighbourhood which is identical to one of two possible options. After reducing an instance in this way, connected series of one of the remaining locals neighbourhoods (which we will later call *clouds*) can be interpreted as a variable that can be set to true or false depending on in which of the two possible ways it will be partitioned into triangles. Under this interpretation, the other possible local neighbourhood (which we will later call a *fan*) can be interpreted as a clause of size three in which exactly one variable must be set to true. In this way, remaining instances can be interpreted as an Exact 3-Satisfiability instance.

We will now give the three lemmas. The first one deals with any instance that is not 4-regular.

Lemma 2 *Let $G = (V, E)$ be an instance of* Partition Into Triangles *of maximum degree four containing a vertex v of degree at most three. In constant time, we can either decide that G is a* No*-instance, or we can transform G into an equivalent smaller instance.*

Proof: We can assume that v has degree three: otherwise the result follows by applying Lemma 1.

Similar to in the proof of Theorem 1, the local neighbourhood of v corresponds to one of the four cases in Figure 1. If this neighbourhood corresponds to case 1, then all edges incident to v are not part of any triangle. If this neighbourhood corresponds to case 2, then the edge between v and the bottom vertex is not part of any triangle. In these two cases, we remove these edges and apply Lemma 1 to v which now has degree at most two. If this neighbourhood corresponds to case 4, then, since G is of maximum degree four, selecting any triangle in the solution results in the creation of a vertex of degree at most one: we can conclude that we have a No-instance. The same holds for case 3 unless the vertices a and b (see Figure 2) are of degree four.

In this last case, we reduce the graph in the following way; see Figure 2. Notice that either vertex a or vertex b must be in a triangle with u and v. Because of this, the other vertex from a

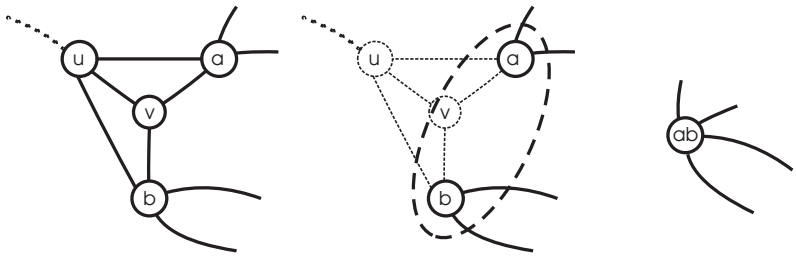

Figure 2: Reducing an instance with a degree three vertex by merging its neighbours.

and b must be in a triangle with its other two neighbours. Therefore, an edge between a neighbour of a and a neighbour of b outside the shown part of the graph cannot be in a triangle in any solution: we remove these if any exist. Next, we merge the vertices a and b to a single vertex and remove both u and v. Notice that the new vertex is part of only two different triangles, and each of these possibilities corresponds to taking one of the two possible triangles containing v in the original graph. Also, no extra triangles are introduced as we have removed the edges between the neighbours of the merged vertices. We conclude that the new smaller instance is equivalent. □

As a result, we can reduce any non 4-regular instance. In a 4-regular graph, a vertex v can have a number of possible local neighbourhoods, all shown in Figure 3. In the next lemmas, we show that we can reduce any instance having a vertex which local neighbourhood does not correspond to cases 2b or 3a in Figure 3. Notice that the numbering corresponds to the number of edges between neighbours of v.

Lemma 3 *Let $G = (V, E)$ be a 4-regular instance of* Partition Into Triangles *containing a vertex v which local neighbourhood is different from cases 2b, 3a and 3b in Figure 3. In constant time, we can either decide that G is a* No*-instance, or we can transform G into an equivalent smaller instance.*

Proof: We inspect the possible local configurations around a vertex of degree four; these are all shown in Figure 3.

If the local neighbourhood of v corresponds to case 0, 1 or 2a, then v is incident to an edge which is not part of any triangle in G since both endpoints do not have a common neighbour. For these cases, we remove the edge and apply Lemma 2 to v, which now has degree three. Furthermore, if the local neighbourhood of v corresponds to case 5 or 6, then we have a No-instances since picking any triangle containing v results in a vertex of degree at most one.

To complete the proof, we consider the remaining two cases: 4a, and 4b.

Case 4a: Consider the edge from the top left vertex to the bottom right vertex. This edge is part of two triangles, one with the centre vertex v and one with the top right vertex. If we would take any of these two triangles in the solution, a vertex of degree one remains. Hence, this edge cannot be part of a triangle in the solution and we can apply Lemma 2 after removing this edge.

Case 4b: Consider one of the four edges in $N[v]$ not incident to v, say the edge between the top two vertices. This edge is part of one or two triangles, one with v, and one with a possible third vertex outside of $N[v]$. Assume that we take the triangle with this edges and v in a solution, then the remaining two vertices will get degree two and thus they can only be in a triangle together and with a common neighbour. Hence, for each of the four edges in $N[v]$, we can remove it if the endpoints of both the edge and the opposite edge (edge between the other two vertices in $N[v] \setminus \{v\}$) have no common neighbour except for v.

We remove these edges. We observe that there is no instance in which all four edges remain. This is so since each of the four corner vertices has only one neighbour outside of $N[v]$, and hence there can be at most two such common neighbours, and if there are two then they must involve the endpoints of opposite edges. Hence, we can apply Lemma 2. □

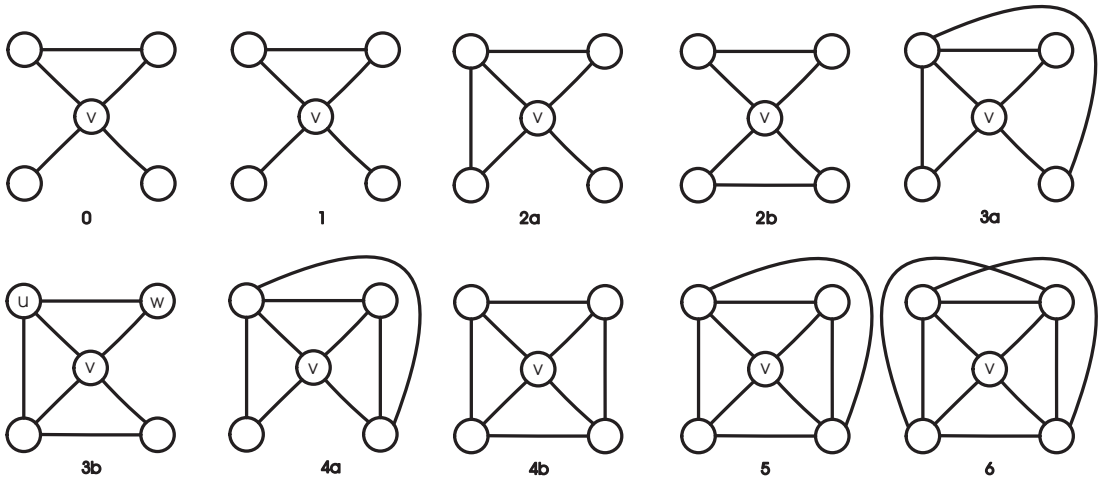

Figure 3: Possible edges within the local neighbourhood of a degree four vertex.

Having reduced the number of possible local neighbourhoods of a vertex in an instance to three, we now remove one more such possibility.

Lemma 4 *Let $G = (V, E)$ be a 4-regular instance of* Partition Into Triangles *in which the local neighbourhood of each vertex corresponds to case 2b, 3a or 3b in Figure 3. Then, the vertices which local neighbourhoods correspond to case 3b form separate connected components in G. If such a connected component exists, then we can either decide that G is a* No*-instance, or we can transform G into an equivalent smaller instance by removing these connected components.*

Proof: Let v be a vertex which local neighbourhood corresponds to case 3b of Figure 3. Let u be the top left vertex in this picture and consider the local neighbourhood of u. This neighbourhood cannot equal case 2b of Figure 3 as it has contains one vertex adjacent to two other vertices in the neighbourhood. The neighbourhood can also not equal case 3a, since v is of degree four and thus cannot have an extra edge to the neighbour of u outside $N[v]$. We conclude that the local neighbourhood of u must equal that of case 3b in Figure 3. Thus, the top two vertices have a common neighbour outside $N[v]$.

We can repeat this argument and apply it to u to conclude that the top right vertex in the picture w also has the same local neighbourhood. This shows that w and the new vertex created in the previous step must have another common neighbour. By repeated application, we conclude that every vertex in the connected component containing v has this local neighbourhood. Moreover, this connected components consists of a circular chain of these configurations as shown in Figure 4.

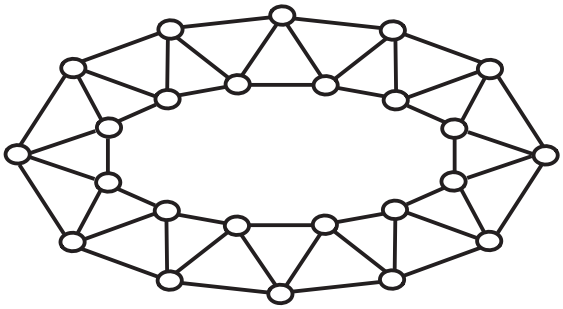

Figure 4: A connected component where each local neighbourhood equals case 3b of Figure 3.

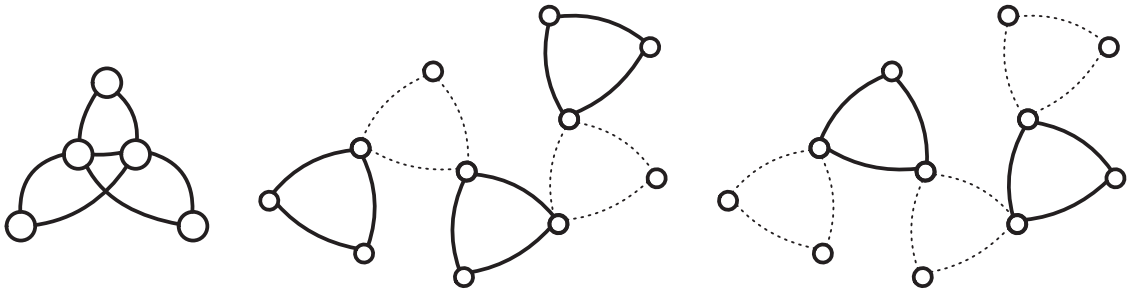

Figure 5: A fan and an example of a cloud and the two ways it can be partitioned into triangles.

It is not hard to see that such a connected component can be partitioned into triangles if and only if the number of vertices in this connected component is a multiple of three. Namely, if we take any triangle in the solution, this will fix other triangles in the solution because vertices will now get degree two. This effect will propagate over the circular chain and result in a triangle partition if and only if the number of vertices is a multiple of three. Therefore, if this number of vertices is a multiple of three, then we can remove it to obtain an equivalent smaller instance, and otherwise we can decide that we have a No-instance. □

We define a *reduced instance* of Partition Into Triangles on maximum degree four graphs to be an instance to which Lemmas 2, 3 and 4 do not apply, i.e., a 4-regular instance in which each local neighbourhood corresponds to case 2a or 3a in Figure 3.

If a vertex in a reduced instance has a neighbourhood corresponding to case 3a in Figure 3, then by definition of this local neighbourhood and because we have a reduced instance, it has one neighbour with this same neighbourhood and it has two neighbours which neighbourhoods correspond to case 2a. We refer to a combination of two vertices corresponding to case 3a in Figure 3 as a *fan*. We will refer to adjacent series of vertices which local neighbourhoods correspond to case 2a as a *cloud* of triangles. See also Figure 5.

Observe how these reduced instances can be partitioned into triangles. In order to partition a fan into triangles, we must select a triangle containing the middle two vertices and exactly one of the three vertices on the boundary. Similarly, in a cloud each triangle is either selected or all its neighbouring (cloud or fan) triangles are selected. Hence, in any partitioning of a cloud into triangles, adjacent triangles will alternate between being selected and not being selected: see Figure 5. As a result, an instance with a cloud that contains a cycle of triangles of odd length is a No-instance since there cannot be such an alternating cycle. Every other cloud has two groups of boundary vertices connecting it to fans: in any solution all fan triangles connected to one group will be selected and all fan triangles connected to the other group will not (see also Figure 5). The only exception to this is the single vertex cloud that directly connects two fans; here the single vertex is in both groups of endpoints.

Now, the relation between Partition Into Triangles on graphs of maximum degree four and Exact 3-Satisfiability emerges. Namely, we can interpret a reduced instance as an X3SAT instance. We interpret a fan as a clause containing three literals represented by its adjacent clouds: exactly one fan triangle must be selected and this choice determines exactly which triangles in the adjacent clouds will be selected. In this way, we interpret a cloud as a variable that can be set to true or false. Both truth assignments correspond to one of the two possible ways to partition the cloud into triangles. The two groups of vertices on the boundary of a cloud then form the positive and the negative literals; these are contained in the clauses represented by adjacent fans. It is not hard to see that this X3SAT interpretation of a reduced instance is satisfiable if and only if the partition into triangles instance has a solution.

Note that an Exact 3-Satisfiability instance that is obtained in this way can have multiple identical clauses. We also have the following property.

Proposition 1 *For any variable x in an X3SAT instance obtained in the above way, the number of positive literals $f_+(x)$ and the number of negative literals $f_-(x)$ differ a multiple of three.*

Proof: Let t_+, t_- be the number of triangles selected within the cloud representing x when x is set to true or false, respectively. A cloud has a fixed number of vertices and for each corresponding truth assignment each vertex is either selected in a triangle or part of a corresponding literal; thus $3t_+ + f_+(x) = 3t_- + f_-(x)$. Hence, $f_+(x) \equiv f_-(x) \pmod 3$. □

The following two propositions show how we can model instances of Exact 3-Satisfiability by reduced instances of Partition Into Triangles of maximum degree four.

Proposition 2 *Any variable x whose number of positive and negative literals differs a multiple of three can be represented by a cloud.*

Proof: Consider a cloud representing a variable, without considering its adjacent fans. Notice that we can increase the number of positive or negative literals of a variable by three in the following way. Take a chain of three triangles connected by common endpoints and identify the loose endpoint of the middle triangle with a vertex representing a literal of the cloud.

Without loss of generality let $f_+(x) > 0$. Starting from the single vertex cloud with $F(x) = (1, 1)$, a single triangle with $F(x) = (3, 0)$, two adjacent triangles with $F(x) = (2, 2)$, or a chain of four triangles with $F(x) = (3, 3)$, we can create any combination $F(x) = (f_+(x), f_-(x))$ given that their difference is a multiple of three. □

Proposition 3 *Any variable x can be represented using $2f(x) - 3$ vertices.*

Proof: See the construction in the proof of Proposition 2. The proposition holds for the initial cases and is maintained every time three triangles are added: this adds six vertices and increases $f(x) = f_+(x) + f_-(x)$ by three. □

We conclude by expressing the relation between Partition Into Triangles on graphs of maximum degree four and Exact 3-Satisfiability in the following theorem.

Theorem 2 *There exist linear time transformations between instances of* Partition Into Triangles *on graphs of maximum degree four and instances of* Exact 3-Satisfiability *in which each variable x has $f_-(x) \equiv f_+(x) \pmod 3$ such that the following holds:*

1. *A given instance has a solution if and only if its transformed instance has a solution.*
2. *An* Exact 3-Satisfiability *instance with variable set X and clause set $\mathcal{C}$ obtained form transforming an n-vertex* Partition Into Triangles *instance of maximum degree four satisfies: $2|\mathcal{C}| + \sum_{x \in X} (2f(x) - 3) \leq n$.*
3. *A* Partition Into Triangles *instance on n vertices obtained form transforming an* Exact 3-Satisfiability *instance in which each variable x has $f_-(x) \equiv f_+(x) \pmod 3$ with variable set X and clauses set $\mathcal{C}$ satisfies: $2|\mathcal{C}| + \sum_{x \in X} (2f(x) - 3) = n$.*

Proof: Given an instance of Partition Into Triangles on graphs of maximum degree four, we can exhaustively apply Lemmas 2, 3 and 4 to obtain an equivalent reduced instance. This can be done in linear time by keeping proper collections of vertices with different degrees and types of neighbourhoods. The resulting reduced instance can be transformed into an equivalent Exact 3-Satisfiability in which each variable x has $f_-(x) \equiv f_+(x) \pmod 3$ as described in the discussion above. It is not hard to see that this last step can be implemented in linear time.

For the reverse transformation, we can use Proposition 2 to construct the required clouds given that the input instance satisfies Property 1. Thereafter, we add fans representing the clauses of the Exact 3-Satisfiability instance. The resulting Partition Into Triangles instances is a reduced instance and is equivalent by the discussion above.

Regarding the size of instance, each fan, and thus each clause, uses two vertices in the Partition Into Triangles instance given that we count the three vertices surrounding it as part of the clouds. By Proposition 3 each variable x uses $2f(x) - 3$ vertices. Hence, the transformation from Exact 3-Satisfiability to Partition Into Triangles satisfies $2|C| + \sum_{x \in X} (2f(x) - 3) = n$. In the reverse transformation, a series of reductions are applied first, and therefore this transformation satisfies $2|C| + \sum_{x \in X} (2f(x) - 3) \leq n$. □

5 Hardness on Graphs of Maximum Degree Four

Having formalised the relation between Partition Into Triangles on graphs of maximum degree four and Exact 3-Satisfiability in the previous section, we are now ready to prove some hardness results. In this section, we will show that the problem Partition Into Triangles on graphs of maximum degree four is $\mathcal{NP}$-complete, and that no subexponential algorithm for this problem exists unless the Exponential Time Hypothesis [5] fails.

Theorem 3 Partition Into Triangles *on graphs of maximum degree four is $\mathcal{NP}$-complete.*

Proof: Clearly, Partition Into Triangles on graphs of maximum degree four is in $\mathcal{NP}$.

For hardness, we reduce from the $\mathcal{NP}$-complete problem Exact 3-Satisfiability [4]. Given an instance of Exact 3-Satisfiability, we can use Theorem 2 if the instance satisfies $f_-(x) \equiv f_+(x) \pmod 3$, for each variable x, to obtain the required equivalent polynomial size Partition Into Triangles on graphs of maximum degree four. If the instance does not satisfy this property, then we can easily force this by making three copies of each clause: every variable x now has that $f_+(x)$ and $f_-(x)$ differ a multiple of three. We complete the proof by noting that this clearly is a polynomial time transformation. □

Theorem 4 *There is no subexponential time algorithm for* Partition Into Triangles *on graphs of maximum degree four unless the Exponential Time Hypothesis fails.*

Proof: Consider an arbitrary 3SAT instance with m clauses. We create an equivalent X3SAT instance with $4m$ clauses by using an equivalence from [10].

$$\mathrm{SAT}(x, y, z) \iff \mathrm{XSAT}(x, v_1, v_2) \wedge \mathrm{XSAT}(y, v_2, v_3) \wedge \mathrm{XSAT}(v_1, v_3, v_4) \wedge \mathrm{XSAT}(\neg z, v_2, v_5)$$

This Exact 3-Satisfiability instance can then be transformed to an instance of the described triangle partitioning problem using the construction in Theorem 3. This construction can triple the number of clauses to $12m$, and thus the total sum of the number of occurrences of the literals will be at most $36m$. By Proposition 3, the variables x can be represented by clouds using less than $2f(x)$ vertices each. In total, this gives at most $94m$ vertices: $72m$ for the variables and another $24m$ for the two vertices of a fan for each clause.

Suppose there exists a subexponential time algorithm for Partition Into Triangles on graphs of maximum degree four running in time $\mathcal{O}(2^{\delta n})$, for all $\delta > 0$. Then, this algorithm solves given 3SAT problems in $\mathcal{O}(2^{\epsilon m})$ for all $\epsilon > 0$ by using the above construction and $\delta = \epsilon/96$. This contradicts the Exponential Time Hypothesis as under this assumption no such algorithm exists by the sparsification lemma [5]. □

6 An Exponential Time Algorithm for Graphs of Maximum Degree Four

In this section, we give a very fast exponential time algorithm solving Partition Into Triangles on graphs of maximum degree four in $\mathcal{O}(1.0222^n)$ time and linear space. The running time does

not involve any large hidden polynomial factors. Actually, our algorithm is an algorithm for Exact Satisfiability that we apply to instances obtained from applying Theorem 2 to the input Partition Into Triangles instance of maximum degree four.

In principle, we could use any known algorithm for Exact 3-Satisfiability or Exact Satisfiability to solve Partition Into Triangles problems of maximum degree four, for example those by Byskov et al. [3]. We do not do so. Instead, we present an algorithm that is specifically tailored to the fact that the input is obtained from an Partition Into Triangles instance. This algorithm had been designed and will be analysed using the number of vertices in a Partition Into Triangles instance used to build the different structures involved in an Exact 3-Satisfiability instance as a measure of progress.

Our algorithm works on Exact Satisfiability instances (not X3SAT). The algorithm is a branch and reduce algorithm. This means that the algorithm exhaustively applies a series of reduction rules. If no such rule applies, then the algorithm branches generating two or more subproblems that are solved recursively. The branching is done in such a way that if any generated subproblem is a Yes-instance, then the original problem is a Yes-instance.

To analyse the resulting algorithm by bounding the number of subproblems generated, we will use the following measure of progress k on instances with variable set X and clause set $\mathcal{C}$.

$$k = 5|X| + \sum_{C \in \mathcal{C}, |C| \geq 3} 2\frac{1}{3}\left(|C| - 3\right)$$

Before justifying this measure, we introduce some standard reduction rules used in many algorithms for XSAT. Besides these reduction rules, we always decide that we have a No-instance if two or more variables in a clause are set to $True$. Also, we set any literals to $False$ if they occur in a clause with a literal set to $True$, and hereafter we remove the clause. After doing so, we remove all literals set to $False$ from the remaining clauses and decide that we have a No-instance if this results in an empty clause.

Below, we let x and y be arbitrary literals (possibly negated), we let C and C' be arbitrary (sub)clauses, and we let Φ be the rest of the current XSAT formula. By $\Phi : a \to b$, we denote the formula Φ with all occurrences of the literal a replaced by b and all occurrences of the literal $\neg a$ by $\neg b$. This notation is extended to sets of variables, for example in $\Phi : C \to False$. The numbers behind the reduction rules represent the minimum reduction in the measure as a result of the reduction.

1.	$C \wedge C \wedge \Phi$	$\Longrightarrow$	$C \wedge \Phi$	(0)
2.	$(x) \wedge \Phi$	$\Longrightarrow$	$\Phi : x \to True$	(-5)
3.	$(x, y) \wedge \Phi$	$\Longrightarrow$	$\Phi : y \to \neg x$	(-5)
4.	$(x, x, C) \wedge \Phi$	$\Longrightarrow$	$C \wedge \Phi : x \to False$	(-5)
5.	$(x, \neg x, C) \wedge \Phi$	$\Longrightarrow$	$\Phi : C \to False$	(-5)
6.	$(x, y, C) \wedge (x, \neg y, C') \wedge \Phi$	$\Longrightarrow$	$(y, C) \wedge (\neg y, C) \wedge \Phi : x \to False$	(-5)
7.	$(x, y, C) \wedge (\neg x, \neg y, C') \wedge \Phi$	$\Longrightarrow$	$\Phi : y \to \neg x;\ C, C' \to False$	(-5)
8.	$C \wedge C' \wedge \Phi$ with $C \subset C'$	$\Longrightarrow$	$C \wedge \Phi : (C' \setminus C) \to False$	(-5)
9.	$(x, C) \wedge (y, C) \wedge \Phi$	$\Longrightarrow$	$(x, C) \wedge \Phi : y \to x$	(-5)
10.	$(x, C) \wedge (C, C') \wedge \Phi$ with $\lvert C\rvert, \lvert C'\rvert \geq 2$	$\Longrightarrow$	$(x, C) \wedge (\neg x, C') \wedge \Phi$	$(-2\frac{1}{3})$
11.	$(x, C) \wedge (\neg x, C') \wedge \Phi$ with $x, \neg x \notin \bigcup \Phi$	$\Longrightarrow$	$(C, C') \wedge \Phi$	$(-2\frac{2}{3})$
12.	If, after application of reduction rules 1-11, Φ contains a variable x and a series of variables $y_1, \ldots, y_l$ only occurring in clauses with x such that every clause that contains x contains exactly one of the variables y_i, then set x to $False$.			(-5)

Lemma 5 *Reduction rules 1-12 are correct and result in the given minimum reductions in the measure k.*

Proof: Reduction rules 1-11 are used in many papers on XSAT, for example [3]; their correctness is evident. For the correctness of rule 12, consider a variable x and a series of variables $y_1, \ldots, y_l$ as in the statement of the reduction rule. Since rules 6 and 7 do not apply, the sign of all literals

of x must be equal, and the same goes for each of the individual variables y_i; without loss of generality, we assume all their literals to be positive. Consider any solution with x set to $True$. Since the y_i occur in clauses with x, they are all set to $False$. Correctness follows because none of the y_i occur in clauses together or in a clause without x, therefore we can replace this assignment by an equivalent one by setting x to $False$ and all the y_i to $True$.

Now, consider the reductions in the measure. In all of the reduction rules except 1, 10 and 11, at least one variable is assigned a value without increasing the size of clauses. Hence, in these cases the measure is reduced by at least 5. Clearly, reduction rule 1 does not increase the size of the measure as it removes a clause. Reduction rule 10 reduces the size of one clause of size at least four by one, and hence reduces the measure by $2\frac{1}{3}$. Finally, reduction rule 11 removes one variable and one possibly large clause of size s reducing the measure by $5 + 2\frac{1}{3}(s-3)$. However, this reduction rule also increases the size of another clause by $s-2$ increasing the measure by $2\frac{1}{3}(s-2)$. Together this leads to a total reduction of $5 - 2\frac{1}{3} = 2\frac{2}{3}$. □

We now justify our measure.

Lemma 6 *Given an instance G of* Partition Into Triangles *of maximum degree four on n vertices, we can either decide that it is a* No*-instance, or we can transform it in polynomial time into an equivalent* Exact Satisfiability *instance of measure k satisfying $k \leq n$.*

Proof: We first apply Theorem 2 to G. If this theorem does not decide that we have a No-instance, it gives us an equivalent Exact 3-Satisfiability instance with variable set X and clause set $\mathcal{C}$ satisfying $2|\mathcal{C}| + \sum_{x \in X} (2f(x) - 3) \leq n$.

Now, we distinguish between two types of variables $x \in X$: variables with $f(x) = 2$ and $F(x) = (1,1)$, and all other variables, which by Property 1 satisfy $f(x) \geq 3$. Let n_2 be the number of variables with $f(x) = 2$, and let $n_{\geq 3}$ be the number of other variables. Then:

$$n \geq 2|C| + \sum_{x \in X} (2f(x) - 3) \geq 2|C| + n_2 + 3n_{\geq 3} = 5n_{\geq 3} + 2\frac{1}{2}n_2$$

where the last equality follows from redistributing the two vertices used by the clauses of size three to the variables: these are given $\frac{2}{3}$ vertices for each occurrence in $\mathcal{C}$.

To this instance, we exhaustively apply reduction rules 1-12. We note that the result of this will not be an instance of X3SAT, but an instance of XSAT instead. The reduction rules, specifically reduction rule 11, will remove all variables x with $f(x) = 2$ and as a result the clauses will increase in size. Clauses increase at most one in size per removed variable, hence:

$$n \geq 5n_{\geq 3} + 2\frac{1}{2}n_2 \geq 5n_{\geq 3} + \sum_{C \in \mathcal{C}, |C| \geq 3} 2\frac{1}{3}\left(|C| - 3\right) = k$$

This proves that $k \leq n$. □

We note that the reductions in the measure proven in Lemma 5 as a result of the reduction rules only apply after first applying Lemma 6: Lemma 6 uses these reductions, specifically those due to reduction rule 11, for its correctness.

The new Exact Satisfiability instance does no longer satisfy Property 1. We notice that in the previous situation Property 1 only held if we counted identical clauses multiple times, and multiple identical clauses have no effect on the satisfiability of an XSAT instance.

Reduction rules 1-12 enforce some new constraints on the resulting XSAT instances. These will be proven in the next lemma. Remind that a *unique* variable is a variable x with $f(x) = 1$.

Lemma 7 *After exhaustively applying reduction rules 1-12 to an* Exact Satisfiability *instance, it satisfies the following properties:*

1. *All clauses have size at least three.*

2. *All variables occur at most once in each clause.*

3. *If variables occur together in multiple clauses, their literals in all clauses have identical signs.*

4. *For any two clauses, each clause contains at least two variables not occurring in the other.*

5. *There are no variables x with $F(x) = (1, 1)$.*

6. *Every clause contains at most one unique variable.*

Proof: (1.) Smaller clauses are removed by reduction rules 2 and 3. (2.) Reduction rule 4 or 5 applies if a clause contains a variable two or more times. (3.) If their literals do not have the same signs, reduction rule 6 or 7 would have been applied. (4.) No clauses are identical by reduction rule 1. Also, no clause is a subclause of another by reduction rule 8. And, if a clause contains only one variable that does not occur in the other clause, reduction rule 9 or 10 is applicable. (5.) By reduction rule 11. (6.) If a clause has more than one unique variable, reduction rule 12 applies. □

What remains is to give a series of lemmas that describe the branching rules of our algorithm. This is what we will do next. Since we first exhaustively apply the reduction rules, we will in each lemma implicitly assume that our reduction rules do not apply, and that directly after the branching all reduction rules are exhaustively applied again. In each lemma, we prove that the described branching has associated branching number at most 1.02220. These branching numbers are computed using the measure k as a measure of progress of the algorithm.

Lemma 8 *If an XSAT instance contains a variable x occurring both as a positive and as a negative literal, then we can either reduce the instance to an equivalent smaller instance, or we can branch on the instance such that the associated branching number is at most* 1.02220.

Proof: Let us first consider branching on a variable x with $f_+(x) \geq 2$ and $f_-(x) \geq 2$, i.e., we have the following situation:

$$(x, C_1) \wedge (x, C_2) \wedge (\neg x, C_1') \wedge (\neg x, C_2') \wedge \Phi$$

We distinguish between different cases: for each of the C_i and C_i', we consider a subccase where this C_i or C_i' has size two and a subcase where the C_i or C_i' has size at least three.

If we set $x \to True$, we obtain the following reductions in the measure. We give these reduction by a series of bullets. Below, we will compute the corresponding sums of the individual reductions for each of the subcases considered.

- 5 for removing x.
- 5 for each literal in C_1 or C_2 as these are set to $False$. Note that by Lemma 7(4), no variable occurs both in C_1 and C_2. If $|C_1| = |C_2| = 2$, then we obtain a reduction of 20, otherwise, we obtain a reduction of at least 25.
- 5 per C_i' with $|C_i'| = 2$ because $\neg x$ is removed from the corresponding clauses resulting in the removal of at least one more variable by reduction rule 3. Notice, that by Lemma 7(3): $(C_1 \cup C_2) \cap (C_1' \cup C_2') = \emptyset$.
- A number of times $2\frac{1}{3}$ for reducing the sizes of the clauses.

The situation is symmetric, hence setting $x \to False$ reduces the measure by the same quantities after replacing C_i by by C_i' and vice versa.

The table below gives all considered cases together with the minimum reductions in the measure obtained by each of the above reasons. In the first two columns, we give the number of C_i and C_i' with $|C_i| \geq 3$ and $|C_i'| \geq 3$, respectively. We assume that all other C_i and C_i' have size two. In the third and forth column, we give the reductions in the measure as a sum of four terms: the first one corresponds to the first bullet given above, the second corresponds to the second bullet, etc. In the last column, we give the branching number τ associated to this branching. By symmetry reasons, we can restrict ourselves to the given cases.

$\#C_i$: $\lvert C_i\rvert \geq 3$	$\#C'_i$: $\lvert C'_i\rvert \geq 3$	reduction of the measure k when we set x to $x \to True$	$x \to False$	τ
0	0	$5+20+10+0=35$	$5+20+10+0=35$	1.02001
1	0	$5+25+10+2\frac{1}{3}=42\frac{1}{3}$	$5+20+5+2\frac{1}{3}=32\frac{1}{3}$	1.01886
2	0	$5+25+10+4\frac{2}{3}=44\frac{2}{3}$	$5+20+0+4\frac{2}{3}=29\frac{2}{3}$	1.01910
1	1	$5+25+5+4\frac{2}{3}=39\frac{2}{3}$	$5+25+5+4\frac{3}{3}=39\frac{3}{3}$	1.01763
2	1	$5+25+5+7=42$	$5+25+0+7=37$	1.01773
2	2	$5+25+0+9\frac{1}{3}=39\frac{1}{3}$	$5+25+0+9\frac{1}{3}=39\frac{1}{3}$	1.01778

While there are variables x with $f_+(x) \geq 2$ and $f_-(x) \geq 2$, we branch on such a variable. So, by negating variables, assume without loss of generality that for each variable x we have have $f_-(x) \in \{0,1\}$ and $f_+(x) \geq 1$.

If $f_+(x) \geq 3$, we can make a similar table associated to the following situation:

$$(x, C_1) \wedge (x, C_2) \wedge (x, C_3) \wedge (\neg x, C) \wedge \Phi$$

Again, the first two columns give the size of $|C|$ and the $|C_i|$; the third and the fourth column contain the reductions in the measure in both branches as a sum of the reductions based on each of the four bullets given above; and the fifth column gives the associated branching number.

$\lvert C\rvert$	$\#C_i$: $\lvert C_i\rvert \geq 3$	reduction of the measure k when we set $x \to True$	$x \to False$	τ
2	0	$5+30+5+0=40$	$5+10+15+0=30$	1.02015
2	1	$5+35+5+2\frac{1}{3}=47\frac{1}{3}$	$5+10+10+2\frac{1}{3}=27\frac{1}{3}$	1.01924
2	2	$5+35+5+4\frac{2}{3}=49\frac{2}{3}$	$5+10+5+4\frac{2}{3}=24\frac{2}{3}$	1.01963
2	3	$5+35+5+7=52$	$5+10+0+7=22$	1.02013
≥ 3	0	$5+30+0+2\frac{1}{3}=37\frac{1}{3}$	$5+15+15+2\frac{1}{3}=37\frac{1}{3}$	1.01874
≥ 3	1	$5+35+0+4\frac{2}{3}=44\frac{2}{3}$	$5+15+10+4\frac{2}{3}=34\frac{2}{3}$	1.01773
≥ 3	2	$5+35+0+7=47$	$5+15+5+7=32$	1.01794
≥ 3	3	$5+35+0+9\frac{1}{3}=49\frac{1}{3}$	$5+15+0+9\frac{1}{3}=29\frac{1}{3}$	1.01820

Again, while there are variables x with $f_+(x) \geq 3$ and $f_-(x) \geq 1$, we branch on such a variable. Because variables x with $F(x) = (1,1)$ are removed by the reductions rules (Lemma 7(5)), the only remaining variables x for which we have to prove the lemma are those with $F(x) = (2,1)$. Let x be such a variable with $F(x) = (2,1)$.

If the negated literal of x occurs in a clause of size three, we apply the following transformation:

$$(x, C_1) \wedge (x, C_2) \wedge (\neg x, v_1, v_2) \wedge \Phi \quad \Longrightarrow \quad (v_1, v_2, C_1) \wedge (v_1, v_2, C_2) \wedge \Phi$$

This transformation is well known under the name resolution; see for example [3]. In this transformation, we remove one variable, but increase two clauses in size by one. Therefore, this transformation does not increase the measure: it is reduced by $5 - 2 \times 2\frac{1}{3} = \frac{1}{3}$.

If the negated literal of x occurs in a clause of size at least four, this corresponds to the following situation with $|C| \geq 3$.

$$(x, C_1) \wedge (x, C_2) \wedge (\neg x, C) \wedge \Phi$$

In this case, we branch on x. We again consider a number of subcases corresponding to the C_i having size two or at least three. The associated branching numbers are again computed in a table similar to the two tables given above.

$\lvert C_1\rvert$	$\lvert C_2\rvert$	reduction of the measure k when we set $x \to True$	$x \to False$	τ
2	2	$5+20+0+2\frac{1}{3}=27\frac{1}{3}$	$5+15+10+2\frac{1}{3}=32\frac{1}{3}$	1.02357
2	≥ 3	$5+25+0+4\frac{2}{3}=34\frac{2}{3}$	$5+15+5+4\frac{2}{3}=29\frac{2}{3}$	1.02183
≥ 3	≥ 3	$5+25+0+7=37$	$5+15+0+7=27$	1.02209

At this point, each case except one gives a branching number that is smaller than the claimed 1.02220. So, to obtain our result, we must analyse this case in more detail: this is the case where the variable x has $F(x) = (2, 1)$ and where $|C_1| = |C_2| = 2$ in the above situation. A refined analysis of the obtained reduction give the result.

Let us inspect this one case a little more thoroughly; it corresponds to the following situation:

$$(x, v_1, v_2) \wedge (x, v_3, v_4) \wedge (\neg x, w_1, w_2, w_3) \wedge \Phi$$

To obtain a branching number that improves upon the one given in the above table, we look at the effect of the branching on Φ. Consider setting x to $True$ and hence the v_i to $False$. At least two of the variables v_i must also occur somewhere in Φ by Lemma 7(6).

Let us first assume that a literal $\neg v_i$ occurs in Φ, and without loss of generality let this be $\neg v_1$. Consider the clause with $\neg v_1$. By Lemma 7(4), this clause cannot contain a literal of v_2, and it must contain at least two literals that are not literals of the variables v_3 and v_4. Hence, this clause must contain at least one variable that we have not considered this far. The literal of this variable will be set to $False$ reducing the measure by at least an additional 5.

If no literals of the form $\neg v_i$ occur in Φ, at least two positive literals of the v_i must occur in Φ; these literals will be set to $False$. We now consider several cases with a clause containing these literals depending on the number of literals in the clause that are not among the v_i. By Lemma 7(4), each clause in Φ can contain at most two v_i and thus must contain at least one literal of a different variable. If these literals fill a clause except for one spot, as in (v_1, v_3, y), then y is set to $True$ reducing the measure by at least an additional 5. And, if these literals fill a clause except for two spots, as in (v_1, v_3, y_1, y_2), then reduction rule 3 replaces y_2 by $\neg y_1$ also reducing the measure by an additional 5. Finally, if these literals fill a clause except for at least three spots, then each such literal will be removed reducing the measure by an additional $2\frac{1}{3}$ each.

Altogether, we conclude that with at least two v_i in Φ, this reduces the measure by at least an additional $4\frac{2}{3}$. Therefore, we obtain a branching number of $\tau(27\frac{1}{3} + 4\frac{2}{3}, 32\frac{1}{3}) = \tau(32, 32\frac{1}{3}) < 1.02179$. □

We notice that systematic subcase analyses as in the proof of the above lemma will be used throughout the rest of this section. We will more often first enumerate the different effects that reduce the measure in general by giving a series of bullets. For each bullet, we will also give the associated reduction in the measure associated with each of the specific properties of possible subcases. Thereafter, we will perform the subcase analysis by giving a table with a row for each subcase giving the relevant properties of this subcase, the total reduction of the measure in each branch as a sum of the effects of each bullet in the before given enumeration, and the associated branching number.

From now on, we assume, without loss of generality, that all variables occur as positive literals only. Based on this assumption, we can give a simple lower bound on the reduction of the measure when we set a number of literals in Φ to $False$. This is formalised in the following proposition. Its proof corresponds somewhat to the last few paragraphs of the proof of Lemma 8.

Proposition 4 *Let Φ be an XSAT formula containing positive literals only. Consider setting some variables with a total of l literals in Φ to $False$. Let Φ contain at least one variable that is not set to $False$. Then, setting the literals to $False$ reduces the measure of Φ by at least $2\frac{1}{3} \times l$ if $l \leq 2$ and 5 if $l \geq 3$ besides the reduction due to removing the corresponding variables.*

Proof: If Φ contains a clause in which all literals are set to $False$, then Φ is not satisfiable resulting in the removal of the whole formula. If Φ contains a clause in which all literals except for one are set to $False$, then the last literals is set to $True$ removing a variable and reducing the measure by at least 5. If Φ contains a clause in which all literals except for two are set to $False$, then the variables corresponding to the last two literals will be replaced by one by reduction rule 3 reducing the measure by at least 5. Finally, if Φ contains a clause in which a literal is set to $False$ and in which at least three literals are not assigned a value, then this reduces the size of the clause reducing the measure by at least $2\frac{1}{3}$ each.

We conclude that the minimum reduction in the measure is $\min(2\frac{1}{3} \times l, 5)$ corresponding to the statement of the lemma. □

We now continue with the next lemma related to the branching of our algorithm.

Lemma 9 *If an XSAT instance contains two clauses that have two or more variables in common, then we can either reduce the instance to an equivalent smaller instance, or we can branch on the instance such that the associated branching number is at most* 1.02220*.*

Proof: Notice, that all literals are positive literals at this point, otherwise, we apply Lemma 8.

Let C be the set of literals contained in both clauses, and let C_1 and C_2 be the literals in each clause not contained in the other. We have the following situation:

$$(C, C_1) \wedge (C, C_2) \wedge \Phi$$

with $|C| \geq 2$ as stated in the lemma, and $|C_1|, |C_2| \geq 2$ by Lemma 7(4).

In most cases, we will branch in the following way. In one subproblem, we assume that a literal in C will be true; consequently, we set all variables in C_1 and C_2 to $False$. In the other subproblem, we assume that none of the literals in C will be true; we set the corresponding variables to $False$. We will distinguish different cases with $|C|$, $|C_1|$ and $|C_2|$ equal to two or at least three.

In the first subproblem, where the literals in C_1 and C_2 are set to $False$, this leads to the following reductions in the measure k:

- 10 per C_i with $|C_i| = 2$ and 15 per C_i with $|C_i| \geq 3$ for removing the variables set to $False$. This is correct since by Lemma 7(2) all variables occur at most once per clause.
- 5 if $|C| = 2$ because then reduction rule 3 will fire and remove one more variable.
- a number of times $2\frac{1}{3}$ depending on the size of C, C_1 and C_2 for reducing the size of the two clauses.
- $4\frac{2}{3}$ if $|C_1| = |C_2| = 2$ and 5 otherwise for the extra reduction of the measure of Φ. By Lemma 7(6), at least two literals in Φ are set to $False$ if $|C_1| = |C_2| = 2$ and at least three literals otherwise. The given reductions correspond to the ones proven in Proposition 4.

In the second subproblem, where the literals in C are set to $False$, this leads to the following reductions in the measure k:

- 10 if $|C| = 2$ and 15 if $|C| \geq 3$ for removing the variables set to $False$.
- 5 for each C_i with $|C_i| = 2$ because, in these cases, reduction rule 3 will remove additional variables.
- a number of times $2\frac{1}{3}$ depending on the size of C, C_1 and C_2 for reducing the size of the two clauses.
- a quantity due to the reduction in the measure of Φ. If $|C| = 2$, this is $2\frac{1}{3}$ because by Lemma 7(6) one of the variables in C must occur in Φ; this leads to the given reduction due to Proposition 4. If $|C| \geq 3$, this is $4\frac{2}{3}$ by the same reasoning.

Next, we compute the branching numbers associated with the branching in the form of a table as in the proof of Lemma 8.

	$\#C_i$:	reduction of the measure k when we set		
$\|C\|$	$\|C_i\| \geq 3$	$C_1, C_2 \to False$	$C \to False$	τ
2	0	$20 + 5 + 4\frac{2}{3} + 4\frac{2}{3} = 34\frac{1}{3}$	$10 + 10 + 4\frac{2}{3} + 2\frac{1}{3} = 27$	1.02298
2	1	$25 + 5 + 7 + 5 = 42$	$10 + 5 + 7 + 2\frac{1}{3} = 24\frac{1}{3}$	1.02167
2	2	$30 + 5 + 9\frac{1}{3} + 5 = 49\frac{1}{3}$	$10 + 0 + 9\frac{1}{3} + 2\frac{1}{3} = 21\frac{2}{3}$	1.02088
≥ 3	0	$20 + 0 + 9\frac{1}{3} + 4\frac{2}{3} = 34$	$15 + 10 + 9\frac{1}{3} + 4\frac{2}{3} = 39$	1.01921
≥ 3	1	$25 + 0 + 11\frac{2}{3} + 5 = 41\frac{2}{3}$	$15 + 5 + 11\frac{2}{3} + 4\frac{2}{3} = 36\frac{1}{3}$	1.01797
≥ 3	2	$30 + 0 + 14 + 5 = 49$	$15 + 0 + 14 + 4\frac{2}{3} = 33\frac{2}{3}$	1.01712

Hence, we have proven the lemma for all cases except when $|C| = |C_1| = |C_2| = 2$. In this case, we have the following situation:

$$(x, y, v_1, v_2) \wedge (x, y, v_3, v_4) \wedge \Phi$$

If, in the branch in which we set $C_1, C_2 \to False$, the additional reduction of the measure of Φ is at least 7, then we obtain the required branching number since $\tau(20+5+4\frac{2}{3}+7, 27) = \tau(36\frac{2}{3}, 27) < 1.02220$.

By Lemma 7(6), at least two occurrences of the literals of v_1, v_2, v_3, and v_4 must occur in Φ. If these are exactly two occurrences, then both x and y must occur at least once in Φ also as reduction rule 12 would otherwise be applicable. In this case, the additional reduction in the measure of Φ in the branch where we set $C \to False$ can be lower bounded by $4\frac{2}{3}$ in stead of $2\frac{1}{3}$ by Proposition 4. In this case, we obtain a branching number of $\tau(34\frac{1}{3}, 10+10+4\frac{2}{3}+4\frac{2}{3}) = \tau(34\frac{1}{3}, 29\frac{1}{3}) < 1.02207$.

If there are at least three occurrences of the literals of v_1, v_2, v_3, and v_4 in Φ, yet setting them to $False$ reduces the measure by less than 7, then all of these v_i must occur in clauses of size three with exactly one other variable z; as for example in: $(v_1, v_3, z) \wedge (v_2, v_4, z)$. This is because all literals occur only as positive literals, and all other configurations that do not directly give a No-instance lead to an additional reduction in the measure of Φ of at least 7: at least three clauses of size at least four will be reduced in size ($3 \times 2\frac{1}{3} = 7$), or at least two variable will be removed ($2 \times 5 > 7$), or exactly one variable will be removed and at least one clause of size at least four is reduced in size ($5 + 2\frac{1}{3} > 7$).

In fact, the only remaining situation is the following:

$$(x, y, v_1, v_2) \wedge (x, y, v_3, v_4) \wedge (v_1, v_3, z) \wedge (v_2, v_4, z) \wedge \Phi$$

This is so since if z exist in a clause of size three with any of the v_i, then z will not occur in a clause of size three with the same v_i again due to Lemma 7(4). Hence, in order to put at least three of the v_i in size three clauses with no other variables than z, exactly one occurrence of each of the four v_is is necessary.

In this special case, we branch z in stead. If we set $z \to True$, this results in v_1, v_2, v_3 and v_4 being set to $False$, and in the replacement of y by $\neg z$ by reduction rule 3. Thus, this removes a total of 6 variables and 2 clauses of size four reducing the measure by at least $6 \times 5 + 2 \times 2\frac{1}{3} = 34\frac{2}{3}$. If we set $z \to False$, this directly results in the following replacements: $v_3 \to \neg v_1$ and $v_4 \to \neg v_2$. In the two clauses with x and y this leads to the following situation $(x, y, v_1, v_2) \wedge (x, y, \neg v_1, \neg v_2)$. This situation is reduced by reduction rule 7 by setting x and y to $False$. In this branch, a total of 5 variables and 2 clauses of size four are removed reducing the measure by at least $5 \times 5 + 2 \times 2\frac{1}{3} = 29\frac{2}{3}$. The associated branching number equals $\tau(34\frac{2}{3}, 29\frac{2}{3}) < 1.02183$ completing the proof of this lemma. □

The next step is to remove variables of relatively high frequency: variables x with $f(x) \geq 4$.

Lemma 10 *If an XSAT instance contains a variable x with $f(x) \geq 4$, then we can either reduce the instance to an equivalent smaller instance, or we can branch on the instance such that the associated branching number is at most* 1.02220.

Proof: We can assume that Lemmas 8 and 9 do not apply, otherwise we are done. Therefore, each variable only has positive literals and no two variables occur together in a clause more than once. This means that we have the following situation:

$$(x, C_1) \wedge (x, C_2) \wedge (x, C_3) \wedge (x, C_4) \wedge \Phi$$

We branch on x and again distinguish several cases based in the sizes of the C_i. If we set $x \to True$, we have the following reductions in the measure.

- 5 for removing x.
- $5 \times \sum_{i=1}^{4} |C_i|$ for removing the variables in the C_i; these are set to $False$.

- a number of times $2\frac{1}{3}$ for reducing the clauses.
- 5 extra since by Lemma 7(6) at least 4 variables must also occur in Φ and this leads to an additional reduction of at least 5 by Proposition 4.

If we set $y \to False$, we have the following reductions in the measure.

- 5 for removing x.
- 5 for each C_i with $|C_i| = 2$ because, in these cases, reduction rule 3 will remove a additional variables.
- a number of times $2\frac{1}{3}$ for reducing the clauses.

Identical to the proofs of the previous lemmas, we calculate the branching numbers for each of the considered cases in a table. In this table, we compute the reduction in the measure in each branch as a sum of the above bullets.

$\#C_i$:	reduction of the measure k when we set		
$\lvert C_i\rvert \geq 3$	$C_1, C_2 \to False$	$C \to False$	τ
0	$5+40+0+5=50$	$5+20+0=25$	1.01944
1	$5+45+2\frac{1}{3}+5=57\frac{1}{3}$	$5+15+2\frac{1}{3}=22\frac{1}{3}$	1.01891
2	$5+50+4\frac{2}{3}+5=64\frac{2}{3}$	$5+10+4\frac{2}{3}=19\frac{2}{3}$	1.01859
3	$5+55+7+5=72$	$5+5+7=17$	1.01849
3	$5+60+9\frac{1}{3}+5=79\frac{1}{3}$	$5+0+9\frac{1}{3}=14\frac{1}{3}$	1.01859

This completes the proof. □

What remains is to remove variables x with $f(x) = 3$. Hereafter, only variables x with $F(x) = (1,0)$ and $F(x) = (2,0)$ remain. At this point, the problem is solvable in polynomial time as noted in many earlier papers on XSAT, for example [3].

Before giving the last lemmas dealing with the branching of the algorithm, we first introduce a new proposition dealing with the additional reductions of the measure due to setting a number of literals in Φ to $False$ under some extra conditions: under these conditions, this will improve upon Proposition 4. Hereafter, we will introduce a new reduction rule that will make sure that these extra conditions apply when needed.

Proposition 5 *Let Φ be an XSAT formula containing positive literals only. Consider setting some variables with a total of l literals in Φ to $False$. Let Φ contain at least three variables that are not set to $False$. Then, setting the literals to $False$ reduces the measure of Φ by at least the following amounts besides the reduction due to removing the corresponding variables.*

1. $\min(2\frac{2}{3} \times l, 15)$ *if no variables exist in Φ that in at least two clauses only occur with literals that have been set to $False$.*
2. $\min(5\lfloor l/4 \rfloor + 2\frac{1}{3}(l \bmod 4), 15)$ *if no variables exist in Φ that in at least three clauses only occur with literals that have been set to $False$.*

Proof: We start with the first situation where no variables exist in Φ occurring only with literals that have been set to $False$ in at least two clauses. If any of the l literals that are set to $False$ occur in a clause of size at least four in Φ, then this removes one literal reducing the measure by $2\frac{1}{3}$. This shows that the minimum reduction of the measure is at most $2\frac{1}{3} \times l$. We will show that this minimum reduction can also be lower bounded by $\min(2\frac{1}{3} \times l, 15)$. Doing so, we consider clauses in Φ with the literals that have been set to $False$ and show that every other configuration reduces the measure by at least as much, or removes at least three variables.

We can assume that there are no clauses containing only literals that are set to $False$ as this results in a No-instance and will remove the whole formula Φ. First, consider clauses in Φ containing only one literal z that is not set to $False$. In these cases, the variable z will be set to

$True$. We note that in the current situation there can be at most one such clause with z. If the clause has size three, two occurrences of the v_i lead to the removal of one extra variable which has more measure than $2 \times 2\frac{2}{3}$. And, with larger clauses, we reduce the measure by an additional $2\frac{1}{3}$ per extra literal: this remains more than given by $2\frac{1}{3} \times l$.

Second, consider clauses in Φ containing two or more literals that have not been set to $False$ in advance. If more than one of these literals is set to $True$ because they also occur in clauses considered in the previous paragraph, then we have a NO-instance and Φ is removed completely. Hence, at most one literal can be set to $True$. If one literal has been set to $True$ in a clause of size three, the remaining one will be set to $False$ reducing the measure by an additional 5 while using only one occurrence of the l literals: this is more than given by $2\frac{1}{3} \times l$ and will remain more if we consider larger clauses also. And, if one literal has been set to $True$ and all other literals have been set to $False$ by the new assignments of the previous step, then, because no two literals may occur in a clause together more than once, at least three different variables that not among the variables initially set to $False$ are given a value: this gives the term 15 in the minimum $\min(2\frac{1}{3} \times l, 15)$.

What remains is to considering clauses in Φ containing two or more literals that have not been set to $False$ in advance and in which no literals are set to $True$ due to the effects in the previous paragraph. Here, we will distinguish between literals that are set to $False$ in advance, literals that are set to $False$ due to the effects in the previous paragraph, and literals of variables that have no given value yet. Again, if all literals have been set to $False$, then we again have a NO-instance. If all literals except for one have been set to $False$ due to the effects of the previous paragraph, then the last literal will be set to $True$; if the clause has size three, this removes one variable while using one occurrences of the l literals, and if the clause is larger, each extra literal increase the reduction of the measure according to the formula $2\frac{1}{3} \times l$. If all literals except for two have been set to $False$ due to the effects of the previous paragraph, then reduction rule 3 will fire also removing one additional variable leading to the same effect. Finally, if some literals have been set to $False$ due to the effects of the previous paragraph, but at least three others remain, then each of the l literals only reduces the size of the clause giving exactly the reductions in the measure of the formula $2\frac{1}{3} \times l$.

This proves the bound on the additional reduction of the measure of Φ under the first condition in the proposition.

For the reduction under the second condition, we can repeat the same proof. The only difference is that Φ can contain one structure that reduces the measure by less than given under the first condition. This is the case if a variable in Φ exists only with some of the l literals that are set to $False$ in two clauses: the situation excluded in the first point. If both clauses are of size three, four of the l literals set to $False$ are used while removing only one additional variable. This reduces the measure by 5 per four literals set to $False$. Using larger clauses, this again increases the reduction of the measure by $2\frac{1}{3}$ each. We conclude that the measure is reduced by at least $\min(5\lfloor l/4 \rfloor + 2\frac{1}{3}(l \bmod 4), 15)$. □

If reduction rules 1-12 and Lemmas 8, 9 and 10 do not apply, we try the following new reduction rule. This reduction rule considers a variable x of frequency three as in the following situation:

$$(x, v_1, v_2, \ldots) \wedge (x, v_3, v_4, \ldots) \wedge (x, v_5, v_6, \ldots) \wedge \Phi$$

13. If, in the above situation, there exists a variable z in Φ that occurs in one clause with only literals of the variables v_i, and all v_i from one of the clauses with x occur in some clause with z, then we apply the replacement: $z \to x$. (-5)

Lemma 11 *Reduction rule 13 is correct and reduces the measure by at least five.*

Proof: Setting $x \to True$ implies $z \to True$ because z occurs in a clause in which it only occurs with variables that are among the v_i. Also, setting $z \to True$ implies $x \to True$ because the v_i that are set to $False$ set all literals in a clause with x to $False$ except for x itself. We conclude that $x = z$ in any solution.

Since this removes a variable, it reduces the measure by at least five. □

Now, we continue by giving the remaining lemmas related to the branching of our algorithm.

Lemma 12 *If an XSAT instance contains a variable x with $f(x) \geq 3$ occurring only in clauses of size three such that the clauses containing x do not have the following form: $(x, v_1, w_1) \wedge (x, w_2, u_1) \wedge (x, w_2, u_2)$ with $f(v_1) = 3$, $f(w_i) = 2$, and $f(u_i) = 1$, for all i. Then we can either reduce the instance to an equivalent smaller instance, or we can branch on the instance such that the associated branching number is at most* 1.02220.

Proof: We can assume that Lemmas 8, 9 and 10 do not apply, otherwise we are done.

We consider the following situation:

$$(x, v_1, v_2) \wedge (x, v_3, v_4) \wedge (x, v_5, v_6) \wedge \Phi$$

Branching on x removes four variables if we set $x \to False$ by reduction rule 3. If we set $x \to True$ this results in the removal of seven variables and an additional reduction in the measure of the instance because of the effect that setting the $v_i \to False$ has on Φ. Let l be the number of occurrences of the literals of v_i in Φ, and let $k(l)$ be the minimum additional reduction in the measure of Φ as a result of setting these literals to $False$. In this way, we obtain a branching number of $\tau(20, 35 + k(l)) < \tau(20, 49) < 1.02171$ if $k(l) \geq 14$.

Notice that Φ cannot contain a variable z that occurs in two clauses with only variables that are among the v_i as this must be at least four different variables v_i and then reduction rule 13 applies. Hence, we can apply Proposition 5: $k(l) \geq \min(2\frac{2}{3} \times l, 15)$.

We will consider the branching numbers for three different values of l. By Lemma 7(6) at least 3 of the v_i must also occur in Φ. Actually, we can make this argument a little stronger by noticing that x may only occur in clauses with at most two unique variables as reduction rule 12 would otherwise apply. I.e., the v_i must occur at least 4 times in Φ: $l \geq 4$.

If $l \geq 6$, then $k(l) \geq 14$ giving a branching number of $\tau(35 + 14, 20) = \tau(49, 20) < 1.02171$.

If $l = 4$, then exactly four of the v_i occur exactly once in Φ and the other two do not occur in Φ. This means that at least one of the clauses with x must contain two literals also occurring in Φ; without loss of generality let these variables be v_1 and v_2. Since $F(v_1) = F(v_2) = (2, 0)$, these two variables are combined to one variable y with $F(y) = (1, 1)$ by reduction rule 3 in the branch where $x \to False$. This fires reduction rule 11 reducing the measure of the instance in this branch by an extra $2\frac{2}{3}$ by Lemma 5. Hence, we obtain a total reduction of the measure of at least $20 + 2\frac{2}{3} = 22\frac{2}{3}$ when $x \to False$. Since $k(l) \geq 9\frac{1}{3}$, the associated branching number is at most $\tau(35 + 9\frac{1}{3}, 22\frac{2}{3}) < 1.02173$.

Finally, let $l = 5$. If any of the three clauses with x contains two v_i with $F(v_i) = (2, 0)$ then we can repeat the argument of $l = 4$ as reduction rule 11 reduces the measure by an extra $2\frac{2}{3}$ in the branching where we set $x \to False$.

The case that remains is when $l = 5$ and no clause containing two v_i with $F(v_i) = (2, 0)$ exists. In this case, two v_i must be unique variables and one v_i must have $F(v_i) = (3, 0)$: this is the one special case excluded in the statement of the lemma. □

Lemma 13 *If an XSAT instance contains a variable x with $f(x) \geq 3$ occurring in two clauses of size three and one clause of size four, then we can either reduce the instance to an equivalent smaller instance, or we can branch on the instance such that the associated branching number is at most* 1.02220.

Proof: We can assume that Lemmas 8, 9, 10 and 12 do not apply, otherwise we are done.

We have the following situation:

$$(x, v_1, v_2) \wedge (x, v_3, v_4) \wedge (x, v_5, v_6, v_7) \wedge \Phi$$

If we set $x \to True$, the measure is reduced by 40 for removing eight variables, $2\frac{1}{3}$ for removing one clause of size four and an additional k_Φ for the additional effect on Φ. If we set $x \to False$,

the measure is reduced by 5 for removing x, 10 for the two replacements due to reduction rule 3, and $2\frac{1}{3}$ for removing one clause of size four. To obtain a bound on k_Φ, we first observe that at least five occurrences of the v_i exist in Φ because otherwise reduction rule 12 fires. If there exists no variable z that occurs in clauses in Φ in at least two clauses only with literals of the v_i, then we apply Proposition 5 to conclude that $k_\Phi \geq 11\frac{2}{3}$. In this case, we obtain a branching number of $\tau(54, 17\frac{1}{3}) < 1.02181$.

The only case that remains is if there exists a variable z that occurs in at least two clauses in Φ only with literals of the v_i. If these are at least three clauses or one of them has size at least four, then literals of at least five v_i are in these clauses as no literal may occur in a clause with z twice: in this case reduction rule 13 applies. Hence, exactly four literals of the v_i occur in clauses with z. I.e., the situation is isomorphic to the following:

$$(x, v_1, v_2) \wedge (x, v_3, v_4) \wedge (x, v_5, v_6, v_7) \wedge (v_1, v_5, z) \wedge (v_3, v_6, z) \wedge \Phi$$

In this specific case, we branch on z. Setting $z \to True$ directly results in the removal of z, v_1, v_3, v_5 and v_6 and indirectly removes three more variables as reduction rule 3 sets $v_2 \to \neg x$, $v_4 \to \neg x$ and $v_7 \to \neg x$, i.e., eight variables are removed reducing the measure by 40. Setting $z \to False$ results in the removal of z and the setting of $v_5 \to \neg v_1$ and $v_6 \to \neg v_3$. This results in the following clauses with x: $(x, v_1, v_2) \wedge (x, v_3, v_4) \wedge (x, \neg v_1, \neg v_3, v_7)$. To this instance, reduction rule 6 fires setting $x \to False$ resulting in $v_2 \to \neg v_1$ and $v_4 \to \neg v_3$. In total, six variables are removed and a clauses of size four is reduced: the measure is reduced by $32\frac{1}{3}$. This gives a branching number of $\tau(32\frac{1}{3}, 40) < 1.01943$. □

Lemma 14 *If an XSAT instance contains a variable x with $f(x) \geq 3$, then we can either reduce the instance to an equivalent smaller instance, or we can branch on the instance such that the associated branching number is at most* 1.02220.

Proof: We can assume that Lemmas 8, 9, 10, 12 and 13 do not apply, otherwise we are done. This means that we only have to consider the following remaining cases:

Two clauses of size three and one clause of size at least five. We have the following situation:

$$(x, v_1, v_2) \wedge (x, v_3, v_4) \wedge (x, v_5, v_6, v_7, v_8, \ldots) \wedge \Phi$$

Setting $x \to False$ removes three variable through reduction rule 3 and it reduces the larger clauses in size: this gives a reduction in the measure of $15 + 2\frac{1}{3} = 17\frac{1}{3}$. Setting $x \to True$ removes at least nine variables, removes a clause of size at least five, and sets at least six literals in Φ to $False$ as reduction rule 12 would otherwise fire. Reduction rule 12 ensures that at least six literals in Φ are set to $False$. Hence, this reduces the measure by at least $45 + 4\frac{2}{3} + 9\frac{2}{3} = 59\frac{1}{3}$ if the effect of the six $False$ literals in Φ is at least $9\frac{2}{3}$: this is the case if the second case of Proposition 5 applies. The resulting branching number equals $\tau(59\frac{1}{3}, 17\frac{1}{3}) < 1.02062$.

To show that the second case of Proposition 5 applies, we do the following. We can restrict ourselves to the case where the large clause with x is of size at most six, otherwise at least two extra variables are removed when $x \to True$ which is more than the $9\frac{2}{3}$ we are proving. Now, if Φ contains a variable that contains in three clauses with literals of the v_i, then, these must be with at most 5 v_i if the third clause has size five, and at most 6 v_i is the third clause has size six, as otherwise reduction rule 13 applies. However, no such variable with five v_i exist, and we have already branched on variables with six v_i using Lemma 12.

One clause of size three and two larger clauses. We have the following situation:

$$(x, v_1, v_2) \wedge (x, v_3, v_4, v_5, \ldots) \wedge (x, v_6, v_7, v_8, \ldots) \wedge \Phi$$

Setting $x \to False$ removes two variable as reduction rule 3 sets $v_2 \to \neg v_1$, and reduces the two larger clauses in size: this gives a reduction in the measure of $10 + 2 \times 2\frac{1}{3} = 14\frac{2}{3}$. Setting $x \to True$ removes at least nine variables, removes two clauses of size at least four, and sets at least six literals in Φ to $False$ as reduction rule 12 would otherwise fire. This reduces the measure

by at least $45 + 4\frac{2}{3} + 9\frac{2}{3} = 59\frac{1}{3}$ if the effect of the six $False$ literals in Φ is at least $9\frac{2}{3}$: this is the case if the second case of Proposition 5 applies. The resulting branching number equals $\tau(59\frac{1}{3}, 14\frac{2}{3}) < 1.02207$.

Now, the second case of Proposition 5 applies for the same reasons at with two clauses of size three and one clause of size at least five: either two additional variables are removed when $x \to True$, or no variable in three clauses with the v_i exists because either reduction rule 13 is applicable, or we have already branched on such variables.

Three clauses of size at least four. If all clauses have size at least four, then we have the following situation:

$$(x, v_1, v_2, v_3, \ldots) \wedge (x, v_4, v_5, v_6, \ldots) \wedge (x, v_7, v_8, v_9, \ldots) \wedge \Phi$$

Setting $x \to False$ removes one variable and reduces all three clauses in size: this gives a reduction in the measure of $5 + 3 \times 2\frac{1}{3} = 12$. Setting $x \to True$ removes at least ten variables, removes at least three clauses of size at least four, and sets at least seven literals in Φ to $False$ as reduction rule 12 would otherwise fire. This reduces the measure by at least $50 + 7 + 10 = 67$ since again, by the same reasoning as the above two cases, either two additional variables are removed, or the second case of Proposition 5 applies to the at least seven literals that are set to $False$ in Φ The resulting branching number equals $\tau(67, 12) < 1.02212$.

The special case of 3 clauses of size 3. At this point, the only variables x with $f(x) = 3$ that remain correspond to the following situation:

$$(x, v_1, v_2) \wedge (x, v_3, u_1) \wedge (x, v_4, u_2) \wedge \Phi$$

with $f(v_1) = 3$, $f(v_2) = f(v_3) = f(v_4) = 2$ and $f(u_1) = f(u_2) = 1$.

Since these are the only remaining variables of frequency three, v_1 must be a similar variable. Hence, a more specific view of the current case is:

$$(x, v_1, v_2) \wedge (x, v_3, u_1) \wedge (x, v_4, u_2) \wedge (v_1, v_5, u_3) \wedge (v_1, v_6, u_4) \wedge \Phi$$

with $f(v_i) = 2$ and $f(u_i) = 1$.

Branching on x results in the required branching number of $\tau(45 + 4\frac{2}{3}, 20) = \tau(49\frac{2}{3}, 20) < 1.02154$. Namely, setting $x \to True$ removes seven variables in the clauses with x, and two variables in the other two clauses due to reduction rule 3. Moreover, the measure of Φ is reduced by at least $4\frac{2}{3}$ by Proposition 4 since v_2 and v_3 also occur in Φ. Setting $x \to False$ removes x and three other variables due to reduction rule 3. □

We now take all our lemmas together to obtain our final result.

Theorem 5 *There is a $\mathcal{O}(1.02220^n)$ time and linear space algorithm for* Partition Into Triangles *on graphs of maximum degree four.*

Proof: We first use Theorem 2 and Lemma 6 to obtain an Exact Satisfiability instance of measure at most n that is equivalent to the Partition Into Triangles instance.

To this instance, we exhaustively apply reduction rules 1-12 and Lemmas 8-14. As a result, we generate a branching tree with at most 1.02220^n leaves, each containing an instance of Exact Satisfiability in which all variables x satisfy $F(x) = (1, 0)$ or $F(x) = (2, 0)$. These instances can be solved in polynomial time and linear space since the problem is equivalent to the question whether the following graph $H = (V', E')$ has a perfect matching.

Let X be the set of variables, and $\mathcal{C}$ be the set of clauses of a remaining XSAT instance. We construct H be letting $V' = \mathcal{C}$ and introducing an edge for each variable $x \in X$ of frequency two between the corresponding clauses. We also add self-loops to all clauses containing a variable x of frequency one. It is not hard to see that every solution of the XSAT instance corresponds to a perfect matching in H and vice versa. □

We notice that the polynomial part of the running time of this algorithm consists of only two components. One, the time required to test which reduction rules and which lemmas should

be applied to the current instance. Two, the the required to test whether there exists a perfect matching in the graphs we build in the leaves of the search tree. Both can be implemented quite efficiently, and thus no large polynomial factors are hidden in the running time of the algorithm.

7 Conclusion

We have shown that the Partition Into Triangles problem is linear time solvable on graphs of maximum degree three, that it is $\mathcal{NP}$-complete on graphs of maximum degree at least four, and that no subexponential time algorithm for this last problem exists unless the Exponential Time Hypothesis fails. For this seemingly hard problem on graphs of maximum degree four, we have given an efficient $\mathcal{O}(1.0222^n)$ time algorithm using only linear space. This raises the question: is this really a hard problem?

To obtain these results, we use an interesting relation between Partition Into Triangles on graphs of maximum degree four and Exact 3-Satisfiability. This relationship emerges by reducing Partition Into Triangles instances of maximum degree four until each vertex can have only two different local neighbourhoods. Connected series of vertices with one of these local neighbourhoods then form the variables of an Exact 3-Satisfiability instance and pairs vertices with the other local neighbourhood form the clauses of this Exact 3-Satisfiability instance.

References

[1] A. Björklund. Exact covers via determinants. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Sciences, STACS 2010*, 2010. To appear.

[2] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal of Computing*, 39(2):546–563, 2009.

[3] J. M. Byskov, B. A. Madsen, and B. Skjernaa. New algorithms for exact satisfiability. *Theoretical Computer Science*, 332(1-3):515–541, 2005.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[5] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, FOCS 1998*, pages 653–663. IEEE Computer Society, 1998.

[6] V. Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37(1):27–35, 1991.

[7] M. Koivisto. Partitioning into sets of bounded cardinality. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC 2009*, volume 5917 of *Lecture Notes in Computer Science*, pages 258–263. Springer Berlin/Heidelberg, 2009.

[8] O. Kuhllmann and H. Luckhardt. Deciding propositional tautologies: Algorithms and their complexity. Manuscript, 1997. http://cs-svr1.swan.ac.uk/∼csoliver/Artikel/tg.ps.

[9] R. J. Lipton. Gödel's lost letter and P=NP, fast exponential algorithms. Weblog, February 2009. http://rjlipton.wordpress.com/2009/02/13/polynomial-vs-exponential-time/.

[10] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Symposium on Theory of Computation, STOC 1978*, pages 216–226, 1978.

[11] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization: "Eureka, you shrink"*, pages 185–207, Berlin, 2003. Springer Lecture Notes in Computer Science, vol. 2570.