LIPIcs Leibniz International Proceedings in Informatics

A Fine-grained Analysis of a Simple Independent Set Algorithm*

Joachim Kneis, Alexander Langer, Peter Rossmanith
Dept. of Computer Science, RWTH Aachen University, Germany
{kneis,langer,rossmani}@cs.rwth-aachen.de

ABSTRACT. We present a simple exact algorithm for the INDEPENDENT SET problem with a runtime bounded by $O(1.2132^n poly(n))$. This bound is obtained by, firstly, applying a new branching rule and, secondly, by a distinct, computer-aided case analysis. The new branching rule uses the concept of satellites and has previously only been used in an algorithm for sparse graphs. The computer-aided case analysis allows us to capture the behavior of our algorithm in more detail than in a traditional analysis. The main purpose of this paper is to demonstrate how a very simple algorithm can outperform more complicated ones if the right analysis of its running time is performed.

1 Introduction

INDEPENDENT SET is one of the most important graph problems. Although it is one of the classical NP-complete problems, it allows for very fast exact algorithms. Even the very trivial *branching* algorithm that recursively tries whether a node of degree at least two belongs to an independent set or not yields a runtime of $O^*(1.47^n)$†. More sophisticated algorithms improve this bound by a large margin.

In this paper, we present a new algorithm for INDEPENDENT SET with a runtime of $O^*(1.2132^n)$ that improves over the runtime $O^*(1.2201^n)$ of the previously best published algorithm by Fomin, Grandoni, and Kratsch [5]. Our algorithm is based on their algorithm and is rather simple: We only use two simple branching rules and few simplification rules. The improvement is based on (1) the usage of the new *satellites* branching rule, and (2) on a new kind of a computer-generated proof. The latter enables us to estimate the effects of reduction rules beyond the neighborhood of a single vertex.

Of course, there is a long history of computer-aided proofs, e.g., for the four color theorem [1, 2]. Still, computer-aided proofs are often hard to verify and sometimes regarded as unaesthetic. We propose a framework that hopefully allows a better and easier verification of automated proofs. The INDEPENDENT SET problem is well-suited for our framework, since the efficiency of branching algorithms for INDEPENDENT SET depends mostly on the case distinctions in small induced subgraphs. Our approach is to use a computer to generate all of them and to evaluate the algorithm in every individual case. Only when time or space constraints render it impossible to generate all cases with a computer, we switch to a classical analysis.

We only briefly recall previous results for INDEPENDENT SET: The first algorithm that improves over the trivial bounds is due to Tarjan and Trojanowski [19]. This algorithm,

*Supported by the DFG under grant RO 927/7
†The O^* notation suppresses polynomial factors.

Foundations of Software Technology and Theoretical Computer Science (Kanpur) 2009.
Editors: Ravi Kannan and K. Narayan Kumar; pp 287–298
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.
Digital Object Identifier: 10.4230/LIPIcs.FSTTCS.2009.2326

which was introduced as early as in 1977, already has a runtime bound of $O^*(1.261^n)$. Further improvements were achieved by Jian [8] and Robson [17] to $O^*(1.235)$ and $O^*(1.228^n)$, respectively. In the same paper, Robson was also able to prove an upper bound of $O^*(1.211^n)$ using exponential space. It is noteworthy that this is based on the Memorization technique, which cannot be used in algorithms that employ so-called folding to remove nodes of degree two.

Fomin, Grandoni, and Kratsch [5] recently employed their Measure & Conquer technique [6] to a new algorithm for INDEPENDENT SET with a runtime bounded by $O^*(1.2201^n)$ that requires only polynomial space. The algorithm itself is extremely simple and the improved runtime is mainly due to an elegant analysis and a new branching rule using mirrors.

Furthermore, it is worthwhile to mention that there is work in progress that might to lead to an even faster, but very complicated algorithm that is partly computer-generated. A preliminary version was published by Robson as a technical report [16, 18].

2 Preliminaries

Let $G = (V, E)$ denote an undirected graph. The size of a maximum independent set in G is denoted by $\alpha(G)$. For any $v \in V$ and any $i \in \mathbf{N}$, the set of nodes of having distance exactly i to v is denoted by $N^i(v)$, i.e., the neighborhood of v is denoted by $N(v) = N^1(v)$. Similarly, $N^i[v]$ denotes the set of nodes having distance at most i to v, such that $N[v] = N^1[v] = N(v) \cup \{v\}$. The degree of a node $v \in V$, i.e., the number of its neighbors in G, is denoted by $\deg(v) = \deg_G(v)$. We assume the reader is familiar with the basic techniques and notation of branching algorithms, in particular with the concept of *measures* (or *potentials*), *branching vectors*, and their corresponding *branching number*.

The concept of *mirrors* was introduced by Fomin, Grandoni, and Kratsch [5]: For some $v \in V$, a node $u \in N^2(v)$ is called *mirror* of v, if $N(v) \setminus N(u)$ is a clique. We denote the set of of a node v mirrors by $M(v)$. Mirrors allow for efficient branching [5]:

LEMMA 1. *Let $G = (V, E)$ be a graph, and $v \in V$. Then $\alpha(G) = \max\{\alpha(G \setminus (M(v) \cup \{v\})), \alpha(G \setminus N[v]) + 1\}$.*

We also apply the concept of *satellites*, which has only been used in algorithms for sparse graphs [9] before. Figure 1 shows some examples of mirrors and satellites, the latter of which are defined as follows:

DEFINITION 2. *Let G be a graph $v \in V$. A node $u \in N^2(v)$ is called* satellite *of v, if there is some $u' \in N(v)$ such that $N[u'] \setminus N[v] = \{u\}$. The set of satellites of a node v is denoted by $S(v)$, and we also use the notation $S[v] := S(v) \cup \{v\}$.*

Note that simple branching algorithms such as the one by Fomin, Grandoni, and Kratsch or our own algorithm typically perform well when they branch on a node v such that $N^2(v)$ is large. If $N^2(v)$ is rather small, there usually is some mirror u of v and branching on v according to Lemma 1 is still efficient. However, there are also some situations where $N^2(v)$ is small, but v has no mirrors, which is the case in four of the five hardest cases in the analysis of Fomin, Grandoni, and Kratsch [5]. Fortunately, satellites allow us to improve these cases, since by the number of edges between $N(v)$ and $N^2(v)$, we can conclude that a satel-

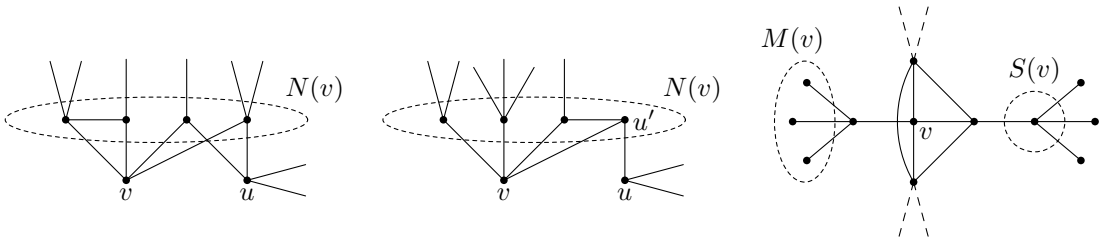

Figure 1: In the graph depicted on the left, u is a mirror of v. In the graph depicted in the middle, u is a satellite of v (through u'). An optimal independent set in the graph on the right contains all nodes in $M(v)$ but no node in $S(v)$. Thus, branching on $G \setminus (\{v\} \cup M(v))$ and $G \setminus N[\{v\} \cup S(v)]$ at the same time does not yield the correct solution.

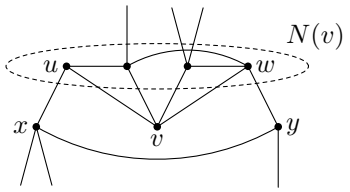

Figure 2: The node v has two adjacent satellites and thus $\alpha(G) = \alpha(G \setminus \{v\})$: If an optimal independent set contains x and v, we can replace v by w. If it contains y and v, we can pick u instead of v.

lite of v must exist, if there is no mirror. The following lemma ([9], Lemma 1) defines the corresponding branching rule.

LEMMA 3. *Let $G = (V, E)$ be a graph, and $v \in V$. Then $\alpha(G) = \max\{\alpha(G \setminus \{v\}), \alpha(G \setminus N[S[v]]) + |S(v)| + 1\}$.*

Note that satellites are particularly useful on graphs with large maximum degree that cannot be analyzed in a computer-aided proof. Unfortunately, we cannot simultaneously branch on mirrors and satellites, as depicted in Figure 1.

Furthermore, our algorithm uses the following well-known reduction rules for independent set that we shortly recall: Firstly, any nodes of degree zero or one can be added to the solution. Similarly, nodes that *dominate*[‡] some other node can be removed from G. Finally, a nodes of degree two not subject to domination can be *folded*, i.e., its neighbors can be merged and the node itself can be removed (see, e.g., [4]). Moreover, Fürer's reduction rule [7] guarantees that each small induced subgraph contains at least three nodes with edges to three distinct nodes in the remaining graph. A precise definition can be found in the appendix.

Finally, satellites can also be used in the following reduction rule (exemplified in Figure 2), which was proven in [9].

LEMMA 4. *Let $G = (V, E)$ be a graph, and $v, u, w \in V$, such that $u, w \in S(v)$ and $\{u, w\} \in E$. Then $\alpha(G) = \alpha(G \setminus \{v\})$.*

[‡]Let $u, v \in V$ be two adjacent nodes. We say u dominates v iff $N[u] \supseteq N[v]$.

We say G is *reduced*, if no further reduction rules can be applied. Moreover, we denote by $R(G)$ the reduced graph obtained from G by applying the reduction rules in some (arbitrarily) fixed order until no more rules can be applied.

Following the Measure & Conquer paradigm [6], we define the following measure on G. This will allow us to prove a better runtime bound than an analysis in $|V|$.

DEFINITION 5. *Let $\varphi_i = 0$ for $i \leq 2$, $\varphi_i = 1$ for $i \geq 7$, $\varphi_3 = 0.474506$, $\varphi_4 = 0.786716$, $\varphi_5 = 0.920901$, and $\varphi_6 = 0.979383$: For a graph $G = (V, E)$ and $v \in V$, we let $\varphi_G(v) := \varphi_{\deg_G(v)}$ and*

$$\varphi(G) = \sum_{v \in V} \varphi_G(v).$$

Obviously, $\varphi(G) \leq |V|$. Any runtime bound in φ therefore immediately implies a runtime bound in $|V|$. Note that the values for φ_i used in this definition are chosen in a way that optimizes the obtained runtime bound. However, these values depend on several thousand recurrences introduced later on, hence they can not be derived easily. We thus used a complex optimization heuristic to compute these values. For $i \geq 5$, φ_i is determined by the runtime on regular graphs of degree i, whereas the values for φ_3 and φ_4 are determined by more complex cases (see [10]).

We write $\varphi(v)$ instead of $\varphi_G(v)$ whenever G can easily be deducted from the context, and let $\Delta_d := \min\{\, \varphi_i - \varphi_{i-1} \mid 4 \leq i \leq d \,\}$ be the minimal measure difference between two nodes in reduced graphs with maximum degree d. Applying the reductions rules does not increase the measure:

LEMMA 6. *Let $G = (V, E)$ be a graph. Then, $\varphi(R(G)) \leq \varphi(G)$.*

PROOF. Removing nodes from the graph respects $\varphi = \varphi(G)$, as some nodes are removed completely and the degree of some adjacent nodes decreases. This does not increase the degree of any node and since $\varphi_i \leq \varphi_{i+1}$ for all $i \in \mathbf{N}$, φ decreases whenever a node is removed.

Whenever a node is folded, its two neighbors u, v are merged. The new node v' can be of higher degree than u and v, but will be at most $\deg(v') \leq \deg(u) + \deg(v) - 2$. Thus, the measure changes by at most $a := \varphi_{\deg(u)+\deg(v)-2} - \varphi_{\deg(u)} - \varphi_{\deg(v)}$. A short computation of all possible combinations shows $a \leq 0$.

Finally, Fürer's reduction rule either removes some nodes of the separator $\{u_1, u_2\}$, adds at most on edge between $\{u_1, u_2\}$ or merges $\{u_1, u_2\}$ into a new node u. Similar to the cases above, removing nodes and merging nodes cannot increase φ. Adding an edge between u_1 and u_2 does not increase φ, because at the same time other edges incident to u_1 and u_2 are removed. ∎

3 A Simple Algorithm for the Independent Set Problem

Combining all the results above, we easily obtain a simple algorithm for the INDEPENDENT SET problem (see Algorithm 1). Its correctness is easy to see, since the reduction rules are valid and the two possible respective branching rules are correct by Lemmas 1 and 3.

Algorithm 1 A fast algorithm for INDEPENDENT SET.

Input: a graph $G = (V, E)$
Output: $\alpha(G)$

```
01: apply reduction rules to G;
02: if G is not connected then compute α for each component independently;
03: if G is cubic then apply algorithm for cubic graphs;
04: select v ∈ V of maximum degree that yields the best branching number;
05: if the mirror branch on v is more efficient than the satellite branch then
06:         return max(α(G \ M[v]), 1 + α(G \ N[v]));
07: else return max(α(G \ {v}), 1 + |S(v)| + α(G \ N[S[v]]));
```

THEOREM 7. *Let $G = (V, E)$. Then, Algorithm 1 correctly returns $\alpha(G)$.*

The remaining part of this paper is devoted to the runtime analysis of Algorithm 1. Basically, this is done by a large case distinction on the effects of the branching and reduction rules when branching on a node v, until we obtain a cubic graph where a faster algorithm exists [9]. If v is of rather high degree, even the trivial algorithm is fast enough. However, with decreasing degree of v, the effects of branching and the subsequent application of reduction rules become more and more important. Down to a degree of five (in general) and for some special cases of degree four, we are still able to give a classical theoretical analysis. However, for the majority of cases having maximum degree of four, even very similar graphs can result in completely different branching vectors. These effects are extremely hard to tackle by an analysis that combines multiple cases. The sharpest runtime bound can be obtained by looking at each possible case individually.

4 A Computer-Aided Case Distinction

Since it is impossible to enumerate each of the millions of possible cases by hand, we use a computer-aided case distinction. Computer-aided proofs are nothing new in the analysis of algorithms, although they still play only a minor role in this field. One example for a computer-aided proof is the algorithm for MAX-2SAT by Kojevnikov and Kulikov [11].

The main problem of computer-generated proofs, and maybe the cause why they are only seldom used, is the complicated verification. While traditional mathematical proofs can be checked rather easily — or at least, we are used to it — this does not hold for computer programs. We therefore propose a framework for computer-aided proofs that allows for an easier verification.

4.1 A General Framework for Computer-Aided Proofs

The first step in any computer-aided proof is to decide which parts of the proof should use the aid of a computer and which parts should be be proven by hand. This step naturally must contain a (traditional) proof of how the computer-aided parts can be incorporated into the traditional proof. The second step is to develop a program that outputs the proof itself and additionally a well-defined *certificate* that lets a reader validate the proof (on a related

note, we refer the reader to the concept of robust and certifying algorithms, see, e.g., [3, 12]). Finally, the proof must be independently validated using the certificate.

We use the framework for the INDEPENDENT SET problem as follows: As outlined above, we want to use a computer-aided proof for certain graphs of maximum degree four, all remaining cases are to be proven by a traditional analysis. Since there are infinitely many graphs of maximum degree four, we only evaluate the branching on a finite number of subgraphs (called *graphlets*, for a formal definition see below). By Theorem 13, this is sufficient.

We developed a computer program that generates all of these graphlets and simulates the two possible branchings (mirrors and satellites) and the subsequent application of the reduction rules. This yields a list of corresponding branching vectors. A complete documentation of this program can be found in [13]. The certificate is given as the complete list of graphlets generated together with their corresponding branching vectors. The certificate and its documentation is publicly available at [10].

In order to verify our proof, one can use the certificate to check (1) whether the certificate is complete, i.e., contains each graphlet or an isomorphic one, (2) whether the corresponding branching vector matches the graphlet, and finally (3) whether the branching vector yields a branching number at most 1.2132.

Finally, an independent team developed programs that validated our certificate, and verified that each of the aforementioned claims actually holds. In the verification team, there was a strong emphasize on clean and simple code so that the verification process can easily be understood by third parties. A full documentation of the verification programs can be found in [15].

We are not aware of any similarly exhaustive approaches to computer-aided proofs that include a formal definition of goals, the proof including a certificate, and particularly an independent verification, with a full documentation of the programs available. An example of an automated proof coming close to our framework are those for MAX-2-SAT by Kojevnikov and Kulikov [11]. Their certificate however does not seem to have been verified independently before publication.

4.2 Generating all Graphlets of Maximum Degree Four

In this section, we give the theoretical foundations for the computer-aided proof. Firstly, we define a notion for reduction rules applied to only a well-defined subgraph of a graph G and show that it this suffices to obtain lower bounds for the real effects of the reduction rules.

DEFINITION 8. *Let $G = (V, E)$ be a graph and let $I \subseteq V$. We define $R_I(G)$ as the graph obtained from G by applying the reduction rules applied to nodes in I only, i.e., (1) remove $u \in I$ if $\deg(u) \leq 1$; (2) remove $u \in I$ if u dominates some $u' \in I$; (3) remove $u \in I$ if u has adjacent satellites $u_1, u_2 \in S(u) \cap I$; (4) if I contains a separator for G of size at most two, apply Fürer's reduction rule; and (5) apply folding to $u \in I$ if $N(u) \subseteq I$.*

From now on, we wlog assume that the reduction rules R on G are always applied in the same order as in R_I.

LEMMA 9. *Let $G = (V, E)$ be a reduced graph of maximum degree d, $I \subseteq V$, $U \subseteq I$. Let $G' = (V', E') = R_I(G \setminus U)$ and let $\Delta_e := \big|\{\, \{u, v\} \in E(G) \mid u \in I, v \notin I \,\}\big| - \big|\{\, \{u, v\} \in E(G') \mid u \in I, v \notin I \,\}\big|$ denote by how much the number of edges between I and $V \setminus I$ changes when applying the reduction rules to I. Then*

$$\begin{aligned} \varphi(R(G \setminus U)) &\leq \varphi(R_I(G \setminus U)) = \varphi(G')) \\ &\leq \varphi(G) - \sum_{v \in I \setminus V'} \varphi_G(v) - \sum_{v \in I \cap V'} \big(\varphi_G(v) - \varphi_{G'}(v)\big) - \Delta_e \min\{\varphi_3/3, \Delta_d\}. \end{aligned}$$

This lemma allows us to evaluate our branching on subgraphs $G[I]$ quite easily: After removing nodes by branching and applying the reduction rules R_I, we can simply count how the degree of all nodes in I changed and add $\min\{\varphi_3/3, \Delta_d\}$ for each removed edge from I to the remaining graph. Note that this is the minimum value each edge contributes to the measure.

DEFINITION 10. *Let $H = (I \cup O, E)$ be graph, such that $I \cap O = \emptyset$, and let $v \in I$ such that $I = N^i[v]$, $O = N^{i+1}(v)$ and $\deg(u) = 1$ for $u \in O$. Moreover, let $\deg(v) \geq deg(u)$ for all $u \in I \cup O$. We call (H, v)* graphlet *of radius i. We call I the* inner nodes *of (H, v) and the set of edges between I and O the* anonymous edges.

Note that the notation of the radius i is motivated by the fact that we are only interested in the number of edges from $N^i(v)$ to $N^{i+1}(v)$ and to which nodes in $N^i(v)$ they are incident. Similarly to Lemma 9, we will restrict our branching and the application of the reduction rules to $I = N^i[v]$.

DEFINITION 11. *Let $G = (V, E)$ be a graph, $v \in V$, and $(H = (I \cup O, E'), v)$ be a graphlet of radius i. We say G contains (H, v) iff (1) $I \subseteq V$, (2) $H[I]$ is an induced subgraph of G, (3) $N_G^{i-1}[v] = I$, and (4) $\deg_G(u) = \deg_H(u)$ for all $u \in I$.*

Note that by these conditions, $|O| = |\{\, \{u, w\} \in E \mid u \in I, w \notin I \,\}|$. While this definition is somewhat technical, the intuition behind it is rather simple: The nodes in I form not only an induced subgraph of G, but I is only connected to $G \setminus I$ via nodes in $N_G^i(v)$. Moreover, the degree of all nodes in I is the same in both graphs and thus the number of edges between $G[I]$ and $G \setminus I$ as well as between $H[I] = G[I]$ and $H \setminus I$ is identical. See Figure 3 for an example.

LEMMA 12. *Let $G = (V, E)$ be a reduced graph that contains a graphlet $(H = (I \cup O, E'), v)$ and $U \subseteq I$. Let $G' = R_I(G \setminus U)$ and $H' = R_I(H \setminus U)$. Then, (1) $G[V \setminus I] = G'[V' \setminus I]$, (2) $\deg_{G'}(v) \leq \deg_{H'}(v)$ for all $v \in I$, and (3) $\big|\{\, \{u, w\} \in E(G') \mid u \in I, w \notin I \,\}\big| \leq \big|\{\, \{u, w\} \in E(H') \mid u \in I, w \notin I \,\}\big|$.*

PROOF. Since we restrict the reduction rules to I, any edge that is not incident to I cannot be affected by the reduction rules. Moreover, only nodes in I can be removed by the restricted reduction rules. Thus, $G[V \setminus I] = G'[V' \setminus I]$.

By induction over the number of applied reduction rules, we easily obtain $G_i[I] = H_i[I]$ and $\deg_{G_i}(u) \leq \deg_{H_i}(u)$ for all $u \in I$, where G_i (and H_i, resp.) denotes the graph G (and H, resp.) after i reduction steps:

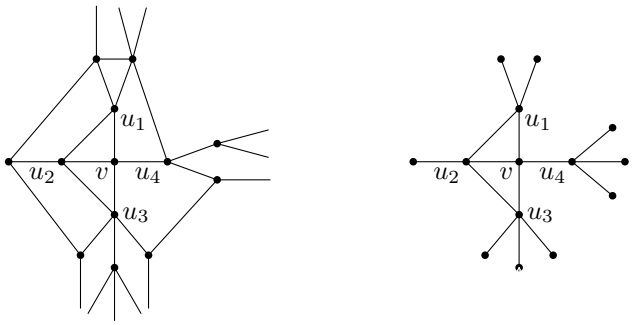

Figure 3: The graph G on the left contains the graphlet (H, v) of orbit 1 on the right. Note that u_1 and u_4 have a common neighbor in $N^2(v)$ in G, but not in (H, v).

1. If I contains a separator of size two in H_i, this is also a separator in G_i and vice versa. Moreover, $G_i[I] = H_i[I]$ implies that Fürer's reduction rule is applied in exactly the same way in both graphs, as the optimal independent sets of these graphs are the same. Hence, $G_{i+1}[I] = H_{i+1}[I]$ and $\deg_{G_{i+1}}(u) \leq \deg_{H_{i+1}}(u)$ for all $u \in I$.

2. Let $u \in I$ be a node that is removed by one of the reduction rules. Removing u in G_i and removing u in H_i removes exactly the same edges within $G_i[I] = H_i[I]$ before the node is removed. Thus, after the removal $\deg_{G_{i+1}}(w) \leq \deg_{H_{i+1}}(w)$ for all $w \in I$ and again $G_{i+1}[I] = H_{i+1}[I]$.

3. Let $u \in I$ be a node that is subject to folding in R_I. By definition of R_I, both neighbors u_1, u_2 of u must belong to I. In G_i as well as in H_i, any edge $\{u_2, w\}$ becomes the new edge $\{u_1, w\}$, Moreover, u and u_2 are removed in both graphs. Therefore, $G_{i+1}[I] = H_{i+1}[I]$ holds after folding u. Since only edges incident to u_2 are changed, we have $\deg_{G_{i+1}}(w) \leq \deg_{H_{i+1}}(w)$ for all $w \in I \setminus \{u_1\}$.

Let $S_H = (N_{H_i}(u_1) \cap N_{H_i}(u_2)) \setminus \{u\}$ and $S_G = (N_{G_i}(u_1) \cap N_{G_i}(u_2)) \setminus \{u\}$. Then $S_H \subseteq S_G$, as the only common neighbors of u_1 and u_2 in H_i must be in I and $G_i[I] = H_i[I]$. But then, $\deg_{G_{i+1}}(u_1) = \deg_{G_i}(u_1) + \deg_{G_i}(u_2) - 2 - |S_G|$ and $\deg_{H_{i+1}}(u_1) = \deg_{H_i}(u_1) + \deg_{H_i}(u_2) - 2 - |S_H|$ imply $\deg_{G_{i+1}}(u_1) \leq \deg_{H_{i+1}}(u_1)$ by induction. We obtain

$$\left|\left\{\, \{u, w\} \in E(G') \mid u \in I, w \notin I \,\right\}\right| \leq \left|\left\{\, \{u, w\} \in E(H') \mid u \in I, w \notin I \,\right\}\right|$$

as a direct consequence of this. ∎

Combining the results above, we can now conclude that is sufficient to evaluate our branching algorithm on graphlets of some fixed radius. After branching and applying the reduction rules to the inner nodes of the graphlet, we only need to analyze how the inner nodes changed and how many anonymous edges are removed to obtain a branching number and (together with the remaining cases) an upper bound for the runtime of Algorithm 1.

THEOREM 13. *Let $G = (V, E)$ be a reduced graph of maximum degree d that contains the graphlet $(H = (I \cup O, E'), v)$. Let $U \subseteq I$ and $H' = R_I(H \setminus U)$. Then*

$$\begin{aligned}\varphi(G) - \varphi(R(G \setminus U)) \;\geq\; & \sum_{u \in I \cap V(H')} \varphi_H(u) - \varphi_{H'}(u) + \sum_{u \in I \setminus V(H')} \varphi_H(u) \\ & + \Delta_{E(H)} \min\{\varphi_3/3, \Delta_d\},\end{aligned}$$

where

$$\Delta_{E(H)} := \left|\{\, \{u,w\} \in E(H) \mid u \in I, w \in O \,\}\right| - \left|\{\, \{u,w\} \in E(H') \mid u \in I, w \in O \,\}\right|$$

denotes the number of anonymous edges that are removed by the reduction rules.

PROOF. Let $G' = R_I(G \setminus U)$. By Lemma 9, we have

$$\begin{aligned}\varphi(G) - \varphi(R(G \setminus U)) \;\geq\; & \sum_{u \in I \cap V(G')} \varphi_G(u) - \varphi_{R_I(G \setminus U)}(u) + \sum_{u \in I \setminus V(G')} \varphi_G(u) \\ & + \Delta_{E(G)} \min\{\varphi_3/3, \Delta_d\},\end{aligned}$$

where

$$\Delta_{E(G)} = \left|\{\, \{u,w\} \in E(G) \mid u \in I, w \notin I \,\}\right| - \left|\{\, \{u,w\} \in E(G') \mid u \in I, w \notin I \,\}\right|.$$

Since G contains the graphlet (H, v), we have

$$\left|\{\, \{u,w\} \in E(G) \mid u \in I, w \notin I \,\}\right| = \left|\{\, \{u,w\} \in E(H) \mid u \in I, w \notin I \,\}\right|.$$

Thus, statement (3) from Lemma 12 yields $\Delta_{E(G)} \geq \Delta_{E(H)}$. By the definition of graphlets, $\varphi_G(u) = \varphi_H(u)$ for all $u \in I$. Moreover, Lemma 12 implies $\deg_{G'}(u) \leq \deg_{H'}(u)$ for all $u \in I$. Hence, $\varphi_{G'}(u) \leq \varphi_{H'}(u)$ for all $u \in I$ and we obtain the claimed estimation. ∎

We can now use use a computer-aided proof for the following theorem:

THEOREM 14. *Let $G = (V, E)$ be a reduced graph of maximum degree four and let $v \in V$ such that $\deg(v) = 4$ and $|N^2(v)| \leq 7$. Then branching on v as described in Algorithm 1 yields a branching with a branching number of at most* 1.2132.

PROOF. Let $\mathcal{H}$ denote the set of all graphlets (H, v) of radius 2 such that $\deg(v) = 4$ and $|N^2(v)| \leq 7$. Then G contains some graphlet $(H', v) \in \mathcal{H}$. By Theorem 13, it is sufficient to simulate the branching on (H', v) and count how the node of the inner nodes changes and how many anonymous edges are removed.

We now have a formal specification of what the computer shall compute as required by the framework outlined in the previous section. Generating all graphlets and computing the branching vectors yields a branching number of at most 1.2132. The certificate is publicly available at [10] and a complete description of the generation and verification programs can be found in [13] and [15], respectively. ∎

5 A Traditional Analysis of the Remaining Cases

Finally, we give an traditional analysis for the remaining cases. Due to the combinatorial explosion, it is impossible to use a computer-aided case distinction for these cases using the methods described in the previous section.

Once the graph is cubic, we apply the algorithm for sparse graphs by Razgon [14], which solves INDEPENDENT SET on cubic graphs in time $O^*(1.0892^n)$. However, we need to be careful because in general, $n \geq \varphi(G)$, but we measure the running time in the latter. Rewriting the statement in terms of our measure, we obtain the following bound.

COROLLARY 15. *Let $G = (V, E)$ be a reduced cubic graph, i.e., $\varphi(G) = n\varphi_3$. Then, Algorithm 1 solves* INDEPENDENT SET *on G in time $O^*(1.0892^n) = O^*(1.0892^{\varphi(G)/\varphi_3}) \leq O^*(1.198^{\varphi(G)})$.*

Please note that we could use a slower but simpler algorithm on cubic graphs, as long as its runtime is at most $O^*(1.096^n)$. For increased readability, we will denote the measure difference between G and $R(G \setminus U)$, for $U \subseteq V$, by $\Delta_\varphi(U) := \varphi(G) - \varphi(R(G \setminus U))$.

For graphs of maximum degree four, we only need to handle the case where $|N^2(v)| \geq 8$. Since a lot of nodes are affected in this case, we easily obtain a good runtime bound.

LEMMA 16. *Let $G = (V, E)$ be a reduced graph of maximum degree four. Let $v \in V$ such that $\deg(v) = 4$ and $|N^2(v)| \geq 8$. Then branching on v as described in Algorithm 1 yields a branching with a branching number of at most 1.201.*

PROOF. In $G \setminus N[v]$, the degree of all nodes in $N^2(v)$ is reduced by at least one. Thus, the measure changes by at least $8 \min\{\varphi_3, \varphi_4 - \varphi 3\}$. Let $d_3 = |\{\, u \in N(v) \mid \deg(u) = 3 \,\}|$. Then $\Delta_\varphi(\{v\}) = \varphi_4 + d_3\varphi_3 + (4 - d_3)(\varphi_4 - \varphi_3)$ and $\Delta_\varphi(N[v]) = \varphi_4 + d_3\varphi_3 + (4 - d_3)\varphi_4$. Computing all five possible branching vectors $\big(\varphi_4 + d_3\varphi_3 + (4 - d_3)(\varphi_4 - \varphi_3), \varphi_4 + d_3\varphi_3 + (4 - d_3)\varphi_4\big)$ yields the desired bound. ∎

Now, only the analysis for graphs of higher degree remains. A complete list of the respective branching vectors and their corresponding branching numbers obtained in the following lemmas can be found at [10].

LEMMA 17. *Let $G = (V, E)$ be a reduced graph of maximum degree $d \geq 5$ and let $v \in V$ such that $\deg(v) = d$. Moreover, let $M(v) = \emptyset$ and $S(v) = \emptyset$. Then branching on v as described in Algorithm 1 yields a branching with a branching number of at most 1.2132.*

LEMMA 18. *Let $G = (V, E)$ be a reduced graph of maximum degree $d \geq 5$ and let $v \in V$ such that $\deg(v) = d$. Moreover, let $u \in M(v)$. Then branching on v as described in Algorithm 1 yields a branching with a branching number of at most 1.2132.*

PROOF. Let $l = \deg(u)$, $S := N(v) \cap N(u)$ and $s := |N(v) \cap N(u)|$. Moreover, let $T := (N(v) \cup N(u)) \setminus S$. Note that the degree of all nodes in S decreases by at least two in $R(G \setminus \{v, u\})$. Therefore, we have

$$\Delta_\varphi(\{v, u\}) \geq \varphi_d + \varphi_l + \sum_{w \in T} (\varphi_{\deg(w)} - \varphi_{\deg(w)-1}) + \sum_{w \in S} (\varphi_{\deg(w)} - \varphi_{\deg(w)-2}) \text{ and}$$

$$\Delta_\varphi(N[v]) \geq \varphi_d + \varphi_l - \varphi_{l-s} + \sum_{w \in N(v)} \varphi_{\deg(w)} + (d - s) \min\{\varphi_3, \Delta_d\},$$

which yields a good enough branching vector $(\Delta_\varphi(\{v, u\}), \Delta_\varphi(N[v]))$ for all cases. ∎

LEMMA 19. *Let $G = (V, E)$ be a reduced graph of maximum degree $d \geq 5$ and let $v \in V$ such that $\deg(v) = d$. Moreover, let $S(v) \neq \emptyset$ and let $M(v) = \emptyset$. Then branching on v as described in Algorithm 1 yields a branching with a branching number of at most 1.2132.*

PROOF. Note that $M(v) = \emptyset$ implies that each node in $N^2(v)$ has at most $d-2$ neighbors in $N(v)$. Assume $|S(v)| \geq 2$ or $S(v) = \{u\}$ and $N(u) \setminus N(v) \neq \emptyset$. Since wlog $V \setminus (N[v] \cup N[S(v)]) \neq \emptyset$ (otherwise the graph is of constant size), we obtain the branching vectors

$$\Big(\varphi_d + \sum_{w \in N(v)} (\varphi_{\deg(w)} - \varphi_{\deg(w)-1}), \varphi_d + \sum_{w \in N(v)} \varphi_{\deg(w)} + 2\varphi_3 + 3\min\{\varphi_3/3, \Delta_d\}\Big),$$

because at least two nodes in $N^2(v)$ of degree at least three are removed and at least three edges connect the corresponding graph to the remaining graph. Otherwise, G contains a separator of size two.

Finally, let $S(v) = \{u\}$ and $N(u) \subseteq N(v)$. Let $\deg(u) = d'$. Since at least $d - d'$ nodes in $N(v)$ have at least two neighbors in $N^2(v)$ (otherwise, $|S(v) > 1|$), we obtain the branching vector

$$\Big(\varphi_d + \sum_{w \in N(v)} (\varphi_{\deg(w)} - \varphi_{\deg(w)-1}), \varphi_d + \varphi_{d'} + \sum_{w \in N(v)} \varphi_{\deg(w)} + 2(d-d')\min\{\varphi_3/3, \Delta_d\}\Big).$$

Again, these branching vectors are good enough except if $\deg(v) = 5$ and all neighbors of v are of degree of five as well. But then we can branch on a neighbor v' of v, such that $N(v')$ contains a node of degree four or less, because the satellite is no mirror and hence of degree 3. ∎

We now easily obtain our main result:

THEOREM 20. *Let $G = (V, E)$ be a graph. Algorithm 1 solves* INDEPENDENT SET *on G in time bounded by $O^*(1.2132^n)$.*

PROOF. First note that for graphs of maximum degree $d > 7$, even the simple branching vector

$$\Big(\varphi_d + \sum_{w \in N(v)} (\varphi_{\deg(w)} - \varphi_{\deg(w)-1}), \varphi_d + \sum_{w \in N(v)} \varphi_{\deg(w)}\Big)$$

is good enough. Also note that $\varphi_i = \varphi_7$ for all $i \geq 8$, and thus increasing the maximum degree to values larger than 8 can never yield a worse branching vector than for a smaller maximum degree, as $N(v)$ contains only more neighbors and it makes no difference whether $N(v)$ contains node of degree 8 or a node of higher degree. A complete list of the respective branching vectors for graphs of maximum degree 8 can be found at [10].

For graphs of maximum degree at most seven, the runtime bound follows from Lemmas 17, 18, and 19, Lemma 16 and Theorem 14, as well as Corollary 15. ∎

6 Conclusion

Although it took some considerable effort to analyze the running time of our algorithm for INDEPENDENT SET, in particular the computer-aided part and its independent verification, we believe the results legitimate this effort. Hopefully, the proposed framework is able to resolve some doubt regarding computer-aided proofs, especially since the certificate can be (and already has been) used to independently validate the proof. We hope that this approach can be used for other problems as well.

References

[1] K. Appel and W. Haken. Solution of the four color map problem. *Scientific American*, 237:108–121, 1977.

[2] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. *Journal of Mathematics*, 21:439–567, 1977.

[3] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.

[4] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. In *Proc. of 25th WG*, number 1665 in LNCS, pages 313–324. Springer, 1999.

[5] F. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm. In *Proc. of 17th SODA*, pages 18–25, 2006.

[6] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: Domination – A case study. In *Proc. of 32nd ICALP*, LNCS, pages 191–203. Springer, 2005.

[7] M. Fürer. A faster algorithm for finding maximum independent sets in sparse graphs. In *Proc. of 7th LATIN*, number 3887 in LNCS, pages 491–501. Springer, 2006.

[8] T. Jian. An $O(2^{0.304n})$ algorithm for solving Maximum Independent Set problem. *IEEE Transactions on Computers*, 35(9):847–851, 1986.

[9] J. Kneis and A. Langer. Satellites and mirrors for solving independent set on sparse graphs. Technical Report AIB-2009-08, RWTH Aachen University, April 2009.

[10] J. Kneis, A. Langer, and P. Rossmanith. Independent set proof homepage, 2009. `http://www.tcs.rwth-aachen.de/independentset/`.

[11] A. Kojevnikov and A. S. Kulikov. A new approach to proving upper bounds for MAX-2-SAT. In *Proc. of 17th SODA*, pages 11–17, 2006.

[12] D. Kratsch, R. M. McConnell, K. Mehlhorn, and J. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM J. Comput.*, 36(2):326–353, 2006.

[13] M. Nett. Generating all relevant cases for the maximum independent set problem. Technical Report AIB-2009-09, RWTH Aachen University, 2009.

[14] I. Razgon. Faster computation of maximum independent set and parameterized vertex cover for graphs with maximum degree 3. *J. Discrete Algorithms*, 7(2):191–212, 2009.

[15] F. Reidl and F. Sánchez Villaamil. Automatic verification of the correctness of the upper bound of a maximum independent set algorithm. Technical Report AIB-2009-10, RWTH Aachen University, 2009.

[16] J. M. Robson. Personal communication.

[17] J. M. Robson. Algorithms for maximum independent sets. *J. Algorithms*, 7:425–440, 1986.

[18] J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical Report 1251-01, Université Bordeaux I, LaBRI, 2001.

[19] R. E. Tarjan and A. E. Trojanowski. Finding a Maximum Independent Set. *SIAM Journal on Computing*, 6(3):537–550, 1977.