Volume 7, issue 2
June 2009
ISSN 1570-8667
ELSEVIER
JOURNAL OF DISCRETE ALGORITHMS
Selected papers from the 2nd Algorithms and Complexity in Durham Workshop ACiD 2006
Editors
Hajo Broersma
Stefan Dantchev
Matthew Johnson
Stefan Szeider
Available online at
ScienceDirect
www.sciencedirect.com

Journal of Discrete Algorithms 7 (2009) 191–212

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Faster computation of maximum independent set and parameterized vertex cover for graphs with maximum degree 3

Igor Razgon

Computer Science Department, University College Cork, Ireland

A R T I C L E   I N F O

Article history:
Available online 20 September 2008

Keywords:
Maximum independent set
Exact exponential algorithms
Vertex cover
Parameterized complexity

A B S T R A C T

In this paper we propose an $O(1.0892^n)$ algorithm solving the Maximum Independent Set problem for graphs with maximum degree 3 improving the previously best upper bound of $O(1.0977^n)$. A useful secondary effect of the proposed algorithm is that being applied to $2k$ kernel, it improves the upper bound on the parameterized complexity of the Vertex Cover problem for graphs with maximum degree 3 (VC-3). In particular, the new upper bound for the VC-3 problem is $O(1.1864^k + n)$, improving the previously best upper bound of $O(k^2 * 1.194^k + n)$. The presented results have a methodological interest because, to the best of our knowledge, this is the first time when a new parameterized upper bound is obtained through design and analysis of an exact exponential algorithm.

1. Introduction

In this paper we study the complexity of a problem of computing a maximum independent set (MIS) of a graph with maximal degree 3. We give this problem a short name MIS-3. We propose an algorithm solving the problem in $O(1.0892^n)$ time. This improves upon a recent method reported in [2] which solves the problem in $O(1.0977^n)$.[1]

A good side effect of the algorithm proposed in the present paper is that it allows to improve the upper bound on the complexity of the parameterized Vertex Cover problem for graphs with maximal degree 3 (we call this problem VC-3). The approach is to solve the MIS-3 problem for the $2k$ kernel existing for the VC-3 problem according to [4] and to take the complement. The resulting parameterized complexity is $O(1.0892^{2k} + n) < O(1.1864^k + n)$, which improves the currently best upper bound $O(k^2 1.194^k + n)$ achieved by Chen et al. [4] for this problem. To the best of our knowledge, this is the first time where a new parameterized upper bound has been obtained through design and analysis of an exact exponential algorithm.

The rest of the introduction consists of 4 subsections. In the first one we overview the existing methods of solving the MIS-3. In the second subsection we introduce the terminology and notations which are necessary for the description of the proposed algorithm. In the third subsection we outline the main ingredients of the proposed algorithm with particular emphasis on the features that allow to get a runtime better than the runtime of other algorithms solving the MIS-3 problem. The structure of the rest of the paper is outlined in the fourth subsection.

E-mail address: i.razgon@cs.ucc.ie.

[1] At the time when the paper was first submitted (February 2006), the best algorithm for the MIS-3 problem was the one proposed in [10] which solved the problem in $O(1.1034^n)$.

1570-8667/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2008.09.004

1.1. Overview of the algorithms solving the MIS-3 problem

The existing exact algorithms for solving the MIS-3 problem can be classified into a number of groups according to their underlying methodology.

The first groups includes *branch-and-prune* based methods, which solve the general MIS problem (not just MIS-3) but whose complexity is measured in terms of the number of edges rather than the number of vertices. The result for the graphs of max-degree 3 is obtained as by-product by taking into account that the number of edges in such graphs are at most 1.5 times the number of vertices. The first algorithm of this group is due to Beigel [1]. This algorithm runs in $O(1.1259^n)$. This upper bound has been improved by Fürer [8] who proposed an $O(1.1120^n)$ algorithm for the MIS-3 problem. Very recently this upper bound has been improved to $O(1.0977^n)$ by Bourgeois et al. [2].

The second group includes algorithms proposed by Chen et al. [3–5], which respectively introduce $O(1.161^n)$, $O(1.1504^n)$, and $O(1.1255^n)$ upper bounds on the complexity of the MIS-3 problem. These algorithms are branch-and-prune methods solving the parameterized VC-3 problem which is complementary to the MIS-3 problem. The transformation of the complexity expression for the VC-3 to the complexity expression for the MIS-3 is based on the fact that the size of the smallest vertex cover of a connected n-vertex graph with max-degree 3 does not exceed $(2n+1)/3$. An interesting feature of the algorithm proposed in [4] is the separate treatment of so-called *alternating paths* i.e. paths where the first and the last vertices have degree 3 and vertices of degree 3 alternate with vertices of degree 2. In particular, the authors proved that the vertices of degree 3 can be either simultaneously selected to a MIS of the given graph or simultaneously removed. If the alternating path is long enough then the branching decision based on this statement very efficiently reduces the problem size. This statement plays a crucial role in isolating the case where *all* vertices of the given graph have degree 3. We use this approach in the present paper for the same purpose.

Fomin and Høie proposed an algorithm [7] which stays away from other algorithms in that it is *not* based on the branch-and-prune methodology. In particular, the authors proved that for a sufficiently large n the path-width on an n-vertex graph of max-degree 3 can be bounded by a number very close to $n/6$. The authors show that this fact allows to solve the MIS-3 problem in $O(1.2225^n)$ by a dynamic programming algorithm. The same upper bound as of [7] was obtained independently by Kojevnikov and Kulikov [9] through solving the MAX-2-SAT problem.

1.2. Notations

In this paper the notion *graph* refers to a simple undirected graph, all vertices of which have degree at most 3. This property is implicitly assumed for all graphs considered in the paper. For example, proving some claim we can say something like 'let v_1, v_2, v_3 be the neighbors of vertex u' without explicitly recalling that by definition the degree of u is at most 3, hence it cannot have more neighbors.

Let $G=(V,E)$ be a graph. The sets of vertices and edges of G are denoted by $V(G)$ and $E(G)$, respectively. Let $S \subseteq V(G)$. We denote by $G[S]$ the graph induced by S and by $G \setminus S$ the graph induced by $V(G) \setminus S$. If S consists of a single vertex v, we write $G \setminus v$ rather than $G \setminus \{v\}$.

For $u \in V(G)$, we denote $N_G(u)$ the set of neighbors of u in G. Let us introduce a number of related notations. $N^+_G(u)$ denotes the set $N_G(u) \cup \{u\}$. For a set $S \subseteq V(G)$, $N_G(S) = (\bigcup_{u \in S} N_G(u)) \setminus S$, $N^+_G(S) = N_G(S) \cup S$. If the considered graph is clear from the context, the subscripts may be omitted for the notations presented in the paragraph.

A set $S \subseteq V(G)$ is *independent* if no two vertices of it are adjacent in G. S is a maximum independent set (MIS) if it is largest subject to this property.

We call a connected component of G a *small* component if it contains at most 50 vertices of degree 3.[2] We denote by $SmallVert(G)$ the set of all vertices that belong to the small components of G and by $SmallVert3(G)$ the set of all vertices of degree 3 that belong to the small components of G. We say that S is a *good* cut if one of the following conditions is satisfied:

- $|S| \leqslant 2$ and $|SmallVert3(G \setminus S)| \geqslant 1$;
- $|S| = 3$ and $|SmallVert3(G \setminus S)| \geqslant 5$;
- $|S| = 4$ and $|SmallVert3(G \setminus S)| \geqslant 10$.

If a good cut S consists of a single vertex u, we sometimes call u a *good cut vertex*.

1.3. Overview of the algorithm and its analysis

We present the proposed algorithm in the form of a function $FindIndep(G)$ whose output is a MIS of the given graph G. Function $FindIndep(G)$ makes a *branching decision* depending on the conditions satisfied by G. An example of a branching decision is the selection of a vertex $v \in V(G)$ and returning the larger set of $\{v\} \cup FindIndep(G \setminus N^+(v))$ and $FindIndep(G \setminus v)$. This branching decision has two *branches* on the first of which v is selected to the returned set, on the other one v is

[2] Number 50 is selected arbitrarily as a sufficiently large constant.

removed. On each branch *FindIndep*(G) is applied recursively to the respective *residual* graph. In the algorithm description we present the conditions checked by *FindIndep*(G) and the branching decisions specified by each condition.

The upper bound on the complexity of *FindIndep*(G) is derived by a standard methodology of analysis of exact algorithms presented, for example, in [11]. The approach is to fix the notion of problem size and to analyze the problem size reductions made by the branching decisions. The simplest measure of problem size is $|V(G)|$, for the input graph G. In this paper we employ a more sophisticated measure expressing problem size as the number of vertices of G having degree 3. The reason is that such vertices determine the exponential complexity of the algorithm: if there are no vertices of degree 3 then a MIS of the given graph can be found efficiently. Moreover, measuring the problem size as the number of vertices of degree 3 allows better reduction of the problem size as compared to the simplest measure. According to the measure, a vertex is considered “removed” not only if it is *actually* removed from the graph but also when its degree is reduced, which essentially increases the number of “removed” vertices at each branch.

In the main part of the algorithm we assume that the graph is *nontrivial*, that is all its vertices are of degree at least 2, there are no adjacent vertices of degree 2, there are no vertices of degree 2 whose neighbors are adjacent, there are no two vertices of degree 2 having the same pair of neighbors. We show that if the input graph is trivial then there is always a vertex guaranteed to belong to a MIS of G. Iteratively selecting such vertices to the returned set, the algorithm eventually “reduces” the input graph to a nontrivial one. A nontrivial graph G can be represented by graph $C(G)$, where vertices of degree 2 are replaced by edges connecting their neighbors. Thus the graph $C(G)$ has two types of edges: *normal* ones, i.e. existing in G and *odd* ones replacing the vertices of degree 2. We show that two ends of an odd edge either both belong to a MIS of G or none of them belongs to a MIS of G. This fact allows to branch on sets of vertices rather than on single vertices, which essentially improves the complexity of the algorithm.

The following conditions are checked by *FindIndep*(G) regarding $C(G)$: presence of a good cut vertex, presence of a good cut of size 2, presence of a good cut of size 3, presence of a cycle of length 4, presence of a cycle of length 3, presence of an odd edge, presence of a good cut of length 4, and the case where none of the previous conditions is satisfied. The branching decision made by *FindIndep*(G) is specified by the *first* satisfied condition in the above list.

The design and analysis of branching decisions for all the above conditions except the last one are based on a relatively easy analysis of the subgraphs induced by the vertices lying close to the vertex being selected by the considered branching decision. For the case where the underlying graph has no small cuts, triangles, rectangles, or odd edges, such approach does not seem to work and more sophisticated means are needed. To achieve the required upper bound in this case, we apply two techniques.

First we employ a branching decision, the last branch of which is based on the assumption that the independent sets constructed on the previous branches are not largest ones. A simple decision based on this paradigm first selects a vertex v then one of the neighbors of v and then the remaining two neighbors of v. This branching decision is based on an observation that if v does not belong to any MIS of G then at least 2 neighbors of v do. We use a more sophisticated branching decision which on the last branch selects four additional vertices *besides* the remaining two neighbors of v. This branching decision is not valid in general, hence we apply it only if graph G has certain properties which make this branching decision valid.

The second applied technique is a method of selection a candidate vertex having particular properties which allow to prune vertices of the branches so that the desired runtime is achieved. The property of the selected vertices is sophisticated in the sense that from the description of the algorithm it is not trivial to see that at least one required vertex exists and we explicitly prove the existence in the analysis part.

1.4. Structure of the paper

The rest of the paper is organized as follows. Section 2 presents the branching decisions that reduce a trivial graph to a nontrivial one. Section 3 consider the branching decisions made in case $C(G)$ has a good cut vertex or a good vertex cut of size 2. Section 4 introduces additional notations and lemmas which are necessary for the further description. Sections 5–10 contain the rest of the description of the algorithm. Section 11 presents the correctness proof, complexity analysis and the upper bound on the parameterized complexity of the VC-3 problem.

In this paper we present the algorithm in a non-standard form. Instead of providing a pseudocode followed by its correctness proof and complexity analysis, we partition the description into a number of sections. Each section presents the branching decisions corresponding to a particular condition. The correctness proof of these branching decisions and the reduction of problem size caused by them are presented in *the same section*. The final section only summarizes the results obtained throughout the paper. We believe that such a form of description has the advantage that the reader can concentrate on the analysis of a particular branching decision of the algorithm without having to remember a dozen of other cases.

2. Initial simplification

We say that a graph G is *nontrivial* if the following conditions are satisfied regarding G.

(1) Each vertex of G has degree at least 2.

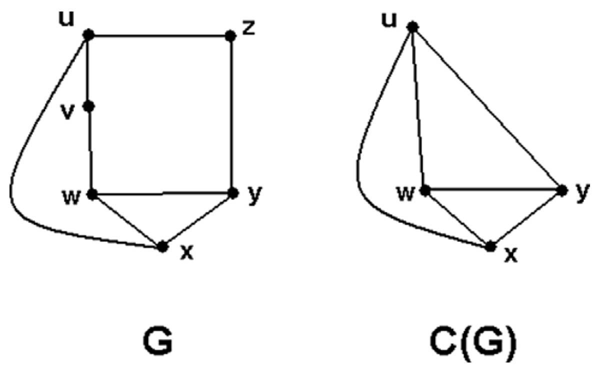

Fig. 1. Transformation from G to $C(G)$.

(2) There are no two adjacent vertices of degree 2.
(3) The neighbors of each vertex of degree 2 are nonadjacent.
(4) There are no two vertices of degree 2 adjacent to the same pair of neighbors.

The first operation performed by *FindIndep*(G) is *simplifying* G in case it is trivial. In particular, if G has a vertex u of degree at most 1 or of degree 2 with both its neighbors adjacent then the set $\{u\} \cup \mathit{FindIndep}(G \setminus N^+(u))$ is returned. If G has two vertices u_1 and u_2 of degree 2 adjacent to the same pair of neighbors then the set $\{u_1, u_2\} \cup \mathit{FindIndep}(G \setminus N^+(\{u_1, u_2\}))$ is returned. Finally, if there are two adjacent vertices u_1 and u_2 of degree 2, the algorithm works as follows. If these vertices are adjacent to the same vertex u then return $\{u_1\} \cup \mathit{FindIndep}(G \setminus N^+(u_1))$. If the remaining neighbors v_1 and v_2 of u_1 and u_2 are distinct then let G' be the graph obtained by the removal of $\{u_1, u_2\}$ from G and introducing the edge between $\{v_1, v_2\}$ (if there is no such an edge in G). Let $S = \mathit{FindIndep}(G')$. Select one vertex $u' \in \{u_1, u_2\}$ which is nonadjacent to S (since v_1 and v_2 are adjacent in G' there is necessarily such a vertex). Return $S' \cup \{u'\}$.

Eventually, as a result of the above operations, $\mathit{FindIndep}$ is applied to a nontrivial graph (note that an empty graph is also a nontrivial one). Let us prove correctness of the simplification operations.

Lemma 1.

(1) *If graph G has a vertex u having degree at most* 1, *or degree* 2 *and adjacent to* 2 *adjacent vertices, this vertex belongs to a MIS of G.*
(2) *If graph G has two nonadjacent vertices u_1 and u_2 of degree* 2 *having the same pair of neighbors then there is a MIS of G containing both u_1 and u_2.*
(3) *If graph G has two adjacent vertices u_1, u_2 of degree* 2 *then at least one of them is contained in a MIS of G.*

Proof.

(1) Let S be a MIS of G, $u \notin S$. Then S contains exactly one neighbor v of u. Observe that $(S \setminus \{v\}) \cup \{u\}$ is a MIS of G.
(2) Let v_1 and v_2 be the 2 neighbors of u_1 and u_2. Let S be a MIS of G such that at least one of u_1 and u_2 is not contained in S. Hence at least one of $\{v_1, v_2\}$ is contained in S. Consequently both u_1 and u_2 are not contained in S. Observe that $(S \setminus \{v_1, v_2\}) \cup \{u_1, u_2\}$ is another MIS of G.
(3) Let v_1 and v_2 be the remaining neighbors of u_1 and u_2. Let S be a MIS of G such that both u_1 and u_2 are not contained in S. Consequently, both v_1 and v_2 are contained in S. Replacing v_1 by u_1 produces another MIS containing u_1. □

From this point and to the end of the description of the algorithm, we consider the behavior of $\mathit{FindIndep}$ applied to a nontrivial graph.

Given a nontrivial graph G, we introduce graph $C(G)$ obtained from G as follows. Each vertex of degree 2 (which is adjacent to two vertices of degree 3 by definition) is replaced by an edge between its neighbors. If vertex v is replaced by edge e, we say that v *corresponds* to e. Observe that $C(G)$ does not have multiple edges because otherwise condition 3 or 4 of the definition of a nontrivial graph is violated. Thus $C(G)$ is a simple cubic graph. An example of transformation from G to $C(G)$ is shown in Fig. 1. We call the edges of $C(G)$ existing in G *normal* edges, the new edges are *odd*. A cycle of $C(G)$ is a normal cycle if all its edges are normal. Otherwise, it is an odd cycle. In Fig. 1, edges $\{u, x\}$, $\{w, y\}$, $\{w, x\}$, $\{x, y\}$ are normal ones, the other edges are odd ones. The cycle w, y, x is normal, the other cycles are odd.

3. Further simplification

In this section we consider the cases where the underlying graph G is empty or it has a small component or it has a good vertex cut of size at most 2.

If G is empty then *FindIndep*(G) returns the empty set. This is the only case where *FindIdep* does not apply itself recursively, so it serves as the stopping condition for the function.

If G has a small component G' then find a MIS S' of G' in a constant time and return $S' \cup$ *FindIndep*$(G \setminus V(G'))$.

The correctness of behavior of *FindIndep*(G) in the last two cases is obvious.

Assume that G has a good cut vertex u. Let $V_1 = SmallVert(G \setminus u)$, $V_2 = (V(G) \setminus \{u\}) \setminus V_1$. Let S_1 be a MIS of $G[V_1]$ and let S_2 be a MIS of $G[V_1] \setminus N(u)$ (both are computed in a constant time without recursive application of *FindIndep*). If $|S_1| > |S_2|$ then return $S_1 \cup$ *FindIndep*$(G[V_2])$. Otherwise, return $S_2 \cup$ *FindIndep*$(G \setminus V_1)$.

Lemma 2. *If G has a good cut vertex u then FindIndep(G) returns a MIS of G provided that the recursive call applied by FindIndep(G) returns a correct answer.*

Proof. Let S be the set returned by *FindIndep*(G) in the considered case. Let S_1 and S_2 be as in the description of the algorithm.

Assume first that $|S_1| > |S_2|$. Clearly S is a MIS of $G \setminus u$. Therefore, the only possible reason why S is not a MIS of G is that any MIS of G contains u. Let S^* be a MIS of G, $|S^*| > |S|$ by our assumption. Observe that $|(S^* \setminus \{u\}) \cap V_2| \leqslant$ *FindIndep*$(G[V_2])$ and that $|(S^* \setminus \{u\}) \cap V_1| < |S_1|$. Consequently, $|S^* \setminus \{u\}| \leqslant |S| - 1$, that is $|S^*| \leqslant |S|$, a contradiction.

If $|S_1|$ is not greater that $|S_2|$ then S is the union of a MIS of $G[V_1]$ and a MIS of $G \setminus V_1$. Since $S \cap (V_1 \cap N(u)) = \emptyset$, S is an independent set. Thus, since S is the union of MISes of two disjoint graphs whose set of vertices partition the graph G, S is a MIS of G. □

Assume that G has a good vertex cut $\{u_1, u_2\}$. Let $V_1 = SmallVert(G, \{u_1, u_2\})$, $V_2 = (V(G) \setminus \{u_1, u_2\}) \setminus V_1$. Let the sets S', $S'(u_1)$, $S'(u_2)$, $S'(u_1, u_2)$ be a MIS of $G[V_1]$, a MIS of $G[V_1] \setminus N(u_1)$, $G[V_1] \setminus N(u_2)$, a MIS of $G[V_1] \setminus N(\{u_1, u_2\})$, respectively. The operations performed by *FindIndep*(G) are described the list below. Each item starts with a condition. We assume that *FindIndep*(G) performs operations corresponding to the *first* satisfied condition in the list below.

(1) $|S'| = |S'(u_1, u_2)|$. Return $S'(u_1, u_2) \cup$ *FindIndep*$(G \setminus V_1)$.
(2) $|S'(u_1)| < |S'(u_2)| = |S'|$. Return $S'(u_2) \cup$ *FindIndep*$(G[V_2 \cup \{u_2\}])$.
(3) $|S'(u_2)| < |S'(u_1)| = |S'|$. Return $S'(u_1) \cup$ *FindIndep*$(G[V_2 \cup \{u_1\}])$.
(4) $|S'(u_1)| = |S'(u_2)| = |S'|$. Let G^* be the graph obtained from $G \setminus V_1$ by introducing an edge between u_1 and u_2 if they are not adjacent. Let $S^* =$ *FindIndep*(G^*). Let S'' be one of $S'(u_1)$, $S'(u_2)$ nonadjacent with S^* (such a set necessarily exist because u_1 and u_2 cannot belong both to S^*). Return $S^* \cup S''$.
(5) $|S'(u_1, u_2)| \leqslant |S'| - 2$. Return $S' \cup$ *FindIndep*$(G[V_2])$.
(6) None of the previous conditions is satisfied. Let G' be a graph obtained from $G \setminus V_1$ by introducing a new vertex w adjacent to u_1 and u_2. Let $S^* =$ *FindIndep*(G'). If exactly one of $\{u_1, u_2\}$ belongs to S^*, remove this vertex from S^*; remove w as well in case $w \in S^*$. Let S'' be the resulting set. If $\{u_1, u_2\} \subseteq S''$, return $S'(u_1, u_2) \cup S''$. Otherwise return $S' \cup S''$.

Lemma 3. *If G has a good vertex cut $\{u_1, u_2\}$ then FindIndep(G) returns a MIS of G provided that the recursive call applied by FindIndep(G) returns a correct answer.*

Proof. Let S be the set returned by *FindIndep*(G). Assume first that $|S'| = |S'(u_1, u_2)|$. In this case S is the union of a MIS of $G[V_1]$ and a MIS of $G \setminus V_1$. The set S is independent because no vertex of $N(\{u_1, u_2\}) \cap V_1$ is contained in S. Thus, since S is the union of MISes of two disjoint graphs whose set of vertices partition the graph G, S is a MIS of G.

Assume now that $|S'(u_1)| < |S'(u_2)| = |S'|$. Arguing as for the previous case, one can see that *FindIndep*(G) returns a MIS of $G \setminus u_1$. Assume by contradiction that any MIS of G contains u_1 and let S^* be a MIS of G. Observe that $S^* \setminus \{u_1\}$ can be partitioned into $S'_1 = (S^* \setminus \{u_1\}) \cap (V_2 \cup \{u_2\})$ and $S'_2 = (S^* \setminus \{u_1\}) \cap V_1$. Clearly, $|S'_1| \leqslant$ *FindIndep*$(G[V_2 \cup \{u_2\}])$ and $|S'_2| \leqslant |S'(u_2)| - 1$. It follows that $|S^* \setminus \{u_1\}| \leqslant |S| - 1$, hence $|S^*| \leqslant |S|$. The case where $|S'(u_2)| < |S'(u_1)| = |S'|$ can be proven symmetrically.

Assume now that $|S'(u_1)| = |S'(u_2)| = |S'|$. Arguing as for the case where $|S'| = |S'(u_1, u_2)|$, we see that S is a MIS of a graph obtained from G by making u_1 and u_2 adjacent. Assume that S is not a MIS of G. Then any MIS S^* of G contains both u_1 and u_2. Consequently, $|S^* \cap V_1| \leqslant |S'| - 1$ and $|S^* \cap (V(G) \setminus V_1)| \leqslant |FindIndep(G')| + 1$. Thus $|S^*| = |S^* \cap V_1| + |S^* \cap (V(G) \setminus V_1)| \leqslant |S'| + |FindIndep(G')| = |S|$, a contradiction.

Assume now that $|S'(u_1, u_2)| \leqslant |S'| - 2$. Observe that S is the largest independent set of G subject to non-including u_1 and u_2. Assume that a MIS S^* of G includes exactly one of u_1, u_2. Then $|S^* \cap (V(G) \setminus V_1)| \leqslant |S \cap (V(G) \setminus V_1)| + 1$ and $|S^* \cap V_1| \leqslant |S \cap V_1| - 1$, that is, we get "compensation" in total. Similarly, if we assume that both u_1 and u_2 belong to S then $|S^* \cap (V(G) \setminus V_1)| \leqslant |S \cap (V(G) \setminus V_1)| + 2$ and $|S^* \cap V_1| \leqslant |S \cap V_1| - 2$, the latter is by our assumption in the considered case.

Finally, consider the last case. Clearly in this case $|S'(u_1, u_2)| = |S'(u_1)| = |S'(u_2)| = |S'| - 1$. Denote by $\alpha(G)$ the size of a MIS of G. Let G' be as in the last item of the above list of cases.

Assume that $\alpha(G) = \alpha(G[V_1]) + \alpha(G \setminus V_1)$. This only possible if $\alpha(G[V_2]) = \alpha(G \setminus V_1)$. Observe that in this case $\alpha(G') = \alpha(G \setminus V_1) + 1$ (G' is as defined in the description of the case) and any MIS of G' contains the new vertex w. Clearly, $|S''| = \alpha(G \setminus V_1)$. Taking into account that $|S'| = \alpha(G[V_1])$ and that S is a disjoint union of S' and S'', $|S| = \alpha(G)$.

Now assume that $\alpha(G) < \alpha(G[V_1]) + \alpha(G \setminus V_1)$. We will show that in this case $|S| \geqslant \alpha(G[V_1]) + \alpha(G \setminus V_1) - 1$. If the set S^* returned by *FindIndep*(G') contains w or exactly one of u_1 or u_2 then, after the transformation, the size of the resulting set S'' is at least $\alpha(G \setminus V_1) - 1$ because $|S''| \geqslant \alpha(G') - 1 \geqslant \alpha(G \setminus V_1) - 1$. In this case $|S| = |S''| + |S'| \geqslant \alpha(G \setminus V_1) - 1 + \alpha(G[V_1])$.

Finally, if the set S^* returned by *FindIndep*(G') contains both u_1 and u_2 then $|S''| = \alpha(G \setminus V_1)$. Consequently, $|S| = |S''| + |S'(u_1, u_2)| = \alpha(G \setminus V_1) + \alpha(G[V_1]) - 1$. □

4. Odd components, FNSes, and related claims

This section presents additional terminology and related claims. They are necessary for further description of the proposed algorithm.

For $u \in V(C(G))$, we denote by *OddComp*$_{C(G)}(u)$ the set of vertices of $C(G)$ consisting of u and the vertices connected to u by paths consisting of odd edges only (the subscript may be omitted, if there is no risk of confusion).

The following lemmas will be very useful for the correctness proof of the proposed algorithm.

Lemma 4. *Assume that there is a MIS of G that does not contain* some *vertices of OddComp(u). Then there is a MIS of G that does not contain* any *vertex of OddComp(u).*

Proof. Let D be the largest subset of *OddComp*(u) such that there is a MIS of G disjoint with D. If $D = $ *OddComp*(u), there is nothing to prove. Otherwise, observe that any MIS of $G \setminus D$ is a MIS of G. Since in $C(G)$ the vertices of D are connected to the vertices of *OddComp*$(u) \setminus D$ by paths consisting of odd edges only, $G \setminus D$ contains a vertex v of degree 1 adjacent to a vertex $w \in$ *OddComp*$(u) \setminus D$. Clearly, v belongs to a MIS of $G \setminus D$ (Lemma 1). Hence there is a MIS of $G \setminus D$, which is, in turn, a MIS of G disjoint with $D \cup \{w\}$ in contradiction to our assumption. □

The following corollary immediately follows from Lemma 4.

Corollary 1. *There is a MIS of G including OddComp(u) or there is a MIS of G non-intersecting with the vertices of OddComp(u).*

Now we define the notion of a *First Nontrivial Successor* (FNS) of the given graph G. A graph G' is a successor of G if *FindIndep*(G') is applied during processing of *FindIndep*(G) (for example, an empty graph is always a successor of G). Graph G' is a nontrivial successor (NS) of G, if G' is a nontrivial graph and does not have a good vertex cut of size at most 2. Graph G'' is *intermediate* between G and G', if G'' is a successor of G and G' is a successor of G''. Graph G' is a first nontrivial successor (FNS) of G, if G' is an NS of G and there is no other NS G'' of G, which is intermediate between G and G'.

If G is nonempty then each branch applied by *FindIndep*(G) leads to exactly one FNS G' of G: for any intermediate graph G'' between G and G', *FindIndep*(G'') applies only one branch (as described in the previous two sections), this sequence of successors can eventually leads to exactly one FNS. As described in the next 4 sections, the branching decision made by *FindIndep*(G) involves at most 3 branches, hence G has at most 3 FNSes. For the convenience, we introduce special notations for these FNSes. In order to do this, we order the branches of *FindIndep*(G) according to the appearance of their description in the text of the paper. Then the *first* branch leads to an FNS of G denoted by G_L (the letter 'L' associated with the left branch in the search tree), the *last* branch leads to an FNS denoted by G_R. If *FindIndep*(G) applies three branches then the *second* (the middle) branch leads to the FNS of G denoted by G_M.

We conclude the section by a number of lemmas which will be useful for obtaining upper bounds on the sizes of FNSes of G.

Lemma 5. *Let G' be a FNS of a nontrivial graph G obtained as a result of making some branching decision on G (selecting vertices to the independent set, removing them from the graph, etc.). Assume that as a result of application of this branching decision to G some vertex u remains of degree* 1*. Then both u and its neighbor v are removed from G'.*

Proof. Let G'' be the residual graph obtained as a result of applying of the branching decision. Let S be the set of all one-degree and isolated vertices of G'' selected by *FindIndep*(G'') to be included to the returned set. If $u \in S$ then clearly both u and v are removed from G'.

Assume that $u \notin S$. In order to understand why it can happen, consider the process of transformation of G'' into a nontrivial graph. It starts from iterative selection into the returned set of all the vertices having degree at most 1 and removal of their neighbors. If at the end of the process, u has not been selected then it has been removed as a neighbor of one of the selected vertices. The only neighbor of u is v. Consequently, v is selected to the returned set and u is removed, as a result, they are both removed from G'. □

Lemma 6. *Let G' and G be as in the previous lemma. Assume that as a result of the branching decision transforming G into G', two neighbors of some vertex u in $C(G)$ are removed from G'. Then u itself is removed from G'.*

Proof. Let G'' be as in the proof of Lemma 5. If vertex u has degree at most 1 in G'' then see Lemma 5. Otherwise, let v_1 and v_2 be 2 neighbors of u in $C(G)$ which do not belong to $V(G'')$. It follows that either $\{u, v_1\}$ or $\{u, v_2\}$ is an odd edge in $C(G)$ and the vertex w of degree 2 corresponding to this odd edge belongs to $V(G'')$. Moreover, this vertex w has degree 1 in $V(G'')$ because one end of the corresponding odd edge is removed. The statement now follows from Lemma 5. □

Lemma 7. *Let G and G' be as in the previous lemmas. Assume that a vertex $u \in V(C(G))$ does not belong to $V(G')$.Then none of the neighbors of u in $C(G)$ belongs to $V(C(G'))$.*

Proof. Let v be a neighbor of u in $C(G)$ which is not removed from $V(G')$. If the edge $\{u, v\}$ is a normal one then the degree of v is at most 2 in G', hence v does not belong to $V(C(G'))$. If the edge $\{u, v\}$ is an odd one then the vertex corresponding to this edge is removed from G' as having degree 1 or by selection of u. Consequently, the degree of v is again at most 2 in G' and the vertex cannot belong to $V(C(G'))$. □

Lemma 8. *Let G' and G be as in the previous lemmas. Assume that as a result of the branching decision transforming G into G', all vertices of $OddComp(u)$ for some vertex $u \in V(C(G))$ are selected into the returned set. Then all the neighbors of u in $C(G)$ are removed from G' and the vertices lying at distance 2 from u in $C(G)$ are removed from $C(G')$.*

Proof. Let v be a neighbor of u in $C(G)$. If the edge $\{u, v\}$ is an odd one then $v \in OddComp(u)$ and removed from G by the assumption of the lemma. If $\{u, v\}$ is a normal edge then v is removed from G' as being a neighbor of a vertex selected to the returned independent set. That the vertices lying at distance 2 from u in $C(G)$ are removed from $C(G')$ follows from Lemma 7. □

5. A good cut of size 3

Let $\{z_1, z_2, z_3\}$ be a good cut of $C(G)$. Let $S = SmallVert(C(G) \setminus \{z_1, z_2, z_3\})$. We may assume that each z_i is adjacent to 2 vertices outside $S \cup \{z_1, z_2, z_3\}$ or that $|S| \geqslant 10$ for otherwise, if, for example, z_1 is adjacent to only one vertex v_1 outside $S \cup \{z_1, z_2, z_3\}$ (if it is adjacent to none, z_1 can be excluded from the cut at all leaving the cut to be of size at most 2, what is processed in the previous section), the algorithm can replace z_1 by v_1 getting a 3-cut separating a superset of S. Proceeding in such a way, the algorithm eventually gets a 3-cut satisfying one of the above conditions. In this case we observe that $|S| \geqslant 5$. Indeed if this is not so then $|S| = 4$ and each z_i is adjacent to 2 vertices outside $S \cup \{z_1, z_2, z_3\}$. Consequently, each z_i is adjacent to *exactly* one vertex w_i *within* S. All of w_1, w_2, w_3 are pairwise distinct because otherwise the existence of a good cut of size 2 follows. Let w_4 be the remaining vertex of S. Since $C(G)$ is a cubic graph, w_4 is adjacent to w_1, w_2, w_3. Now we get a contradiction: w_1, w_2, w_3 cannot simultaneously have degree 3. In particular, to avoid one of them to be of degree 4, $C(G)[\{w_1, w_2, w_3\}]$ can contain at most one edge. Consequently, one of $\{w_1, w_2, w_3\}$ remains of degree 2. This contradiction shows the correctness of our observation that $|S| \geqslant 5$.

In the considered case, $FindIndep(G)$ returns the larger set of $OddComp(z_1) \cup FindIndep(G \setminus N^+(OddComp(z_1)))$ and $FindIndep(G \setminus OddComp(z_1))$. The correctness of this behavior follows from Corollary 1. Let us compute the sizes of FNSes of G.

Lemma 9. *In the considered case, $|V(C(G_L))| \leqslant |V(C(G))| - 10$ and $|V(C(G_R))| \leqslant |V(C(G))| - 8$.*

Proof. Observe that the vertices of S belong neither to G_L not to G_R, otherwise these graphs are trivial or satisfy one of the cases considered in Section 3 (the mere removal of z_1 from G leaves the vertices of S separated from the rest of the graph by a vertex cut of size 2).

It follows that both $|V(C(G_L))|$ and $|V(C(G_R))|$ are at most $|V(C(G))| - |S|$. If $|S| = 10$, there is nothing to prove further. Otherwise, z_1 is adjacent to 2 vertices y_1, y_2 outside $S \cup \{z_1, z_2, z_3\}$. Taking into account that z_1, y_1, y_2 are removed from both $V(C(G_L))$ and $V(C(G_R))$ (Lemma 7) and that $|S| \geqslant 5$ by definition, $|V(C(G_L))| \leqslant |V(C(G))| - 8$ and $|V(C(G_R))| \leqslant |V(C(G))| - 8$. It remains to find 2 additional vertices removed from $C(G_L)$. Note that there are at least two vertices w_1, w_2 different from y_1, y_2 and adjacent to y_1, y_2 (otherwise y_1, y_2 can be separated from the rest of the graph by a cut of size at most 2). Vertices w_1, w_2 cannot belong to S since y_1, y_2 are not adjacent to any vertex of S by definition. Vertices w_1, w_2 are removed from $C(G_L)$ by Lemma 8. Hence w_1, w_2 are the desired two additional vertices removed from $C(G_L)$. □

6. Processing a rectangle

In this section we describe behavior of $FindIndep(G)$, if $C(G)$ has a rectangle (a cycle of length 4).

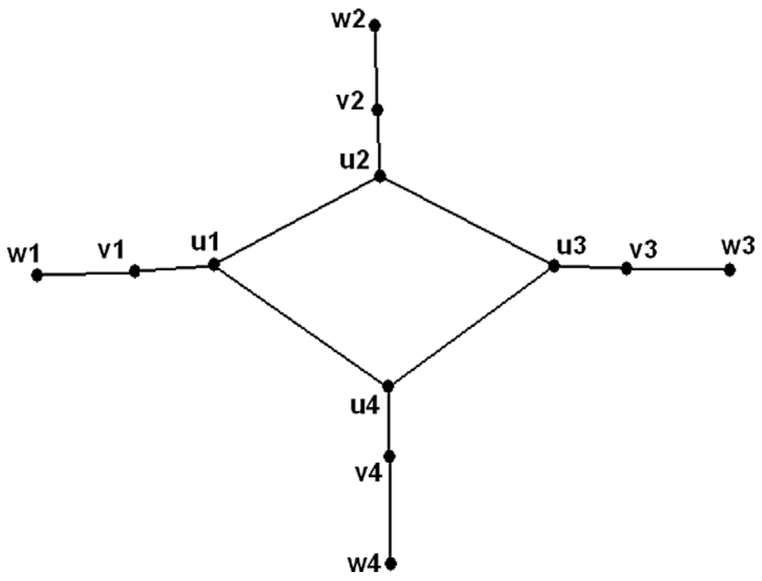

Fig. 2. A rectangle of $C(G)$ and surrounding vertices.

Pick a rectangle of $C(G)$ and denote its vertices by $u_1, \dots, u_4$. The next lemma creates the necessary basis for justifying the behavior of *FindIndep* for the considered case.

Lemma 10. *There are vertices $v_1, \dots, v_4, w_1, \dots, w_4$ connected to $u_1, \dots, u_4$ as shown in Fig. 2. (Notice, we claim that the graph shown in Fig. 2 is a subgraph of $C(G)$ but* not an induced subgraph *of $C(G)$. There may be edges between vertices $v_1, \dots, v_4, w_1, \dots, w_4$ not shown in Fig. 2, but they are not relevant for our discussion.)*

Proof. Note first that u_1 is not adjacent to u_3 as well as u_2 is not adjacent to u_4 because otherwise either $u_1, \dots, u_4$ is disconnected from the rest of the graph or there is a good cut of size 2. Let $S = N_{C(G)}(\{u_1, \dots, u_4\})$. To avoid a good cut of size at most 3, $|S| = 4$ and each vertex of S is adjacent to exactly one vertex of $\{u_1, \dots, u_4\}$ analogously to vertices $\{v_1, \dots, v_4\}$ shown in Fig. 2.

Now, let $T = N_{C(G)}(S) \setminus \{u_1, \dots, u_4\}$. Observe that for each $K \subseteq S$, $|N_{C(G)}(K) \cap T| \geqslant |K|$. Indeed, if otherwise, then $(S \setminus K) \cup (N_{C(G)}(K) \cap T)$ is good cut of size at most 3. Consider now a bipartite graph (S, T, E'), where E' are the edges of $C(G)$ having one end in S, the other in T. By Hall's Theorem (see, for example, [6, Theorem 2.1.2]), the considered bipartite graph has a matching of size 4. Let $w_1, \dots, w_4$ be the vertices matched to $v_1, \dots, v_4$, respectively, in such a matching. The edges of the matching together with the edges connecting $v_1, \dots, v_4$ to $u_1, \dots, u_4$ and together with the rectangle formed by $u_1, \dots, u_4$ form the desired subgraph of $C(G)$. □

Now we are ready to present the behavior of *FindIndep*(G) in case $C(G)$ contains a rectangle. The presentation is divided into a number of subsection depending on the status of the edges of the considered rectangle denoted by R.

6.1. Rectangle R is normal

In this case *FindIndep*(G) returns the larger set of $\{u_1, u_3\} \cup \mathit{FindIndep}(G \setminus N^+(\{u_1, u_3\}))$ and $\{u_2, u_4\} \cup \mathit{FindIndep}(G \setminus N^+(\{u_2, u_4\}))$. Let us justify correctness of this behavior.

Lemma 11. *Assuming that the recursive calls to FindIndep are correct, one of the above sets is a MIS of G.*

Proof. Assume that the former set is not a MIS of G. In other words, there is no MIS of G containing both u_1 and u_3. Assume that there is a MIS of G containing u_1 only. Such a MIS can be represented as $\{u_1\} \cup S'$, where S' is a MIS of $G \setminus N^+(u_1)$. Since u_3 has degree 1 in $G \setminus N^+(u_1)$, we may assume $u_3 \in S'$ (Lemma 1), i.e. both u_1 and u_3 are included into a MIS of G in contradiction to our assumption. We get a similar contradiction assuming that there is a MIS of G containing only u_3.

It follows that neither u_1 nor u_3 is contained in a MIS of G or, in other words that a MIS of $G \setminus \{u_1, u_3\}$ is a MIS of G. Since both u_2 and u_4 have degree 1 in $G \setminus \{u_1, u_3\}$ and they are nonadjacent, they are both contained in a MIS of $G \setminus \{u_1, u_3\}$ (Lemma 1) and, consequently, in a MIS of G. Thus, the latter set considered by *FindIndep* is a MIS of G. □

Now we compute upper bounds on the size of FNSes of G.

Lemma 12. *In the considered case, at least one of the following statements is true.*

(1) $|V(C(G_L))| \leqslant |V(C(G))| - 10$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 8$;
(2) $|V(C(G_L))| \leqslant |V(C(G))| - 8$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 10$;
(3) $|V(C(G_L))| \leqslant |V(C(G))| - 9$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 9$.

Proof. Since all of $\{u_1, \dots, u_4\}$ are removed from both G_L and G_R, neither of $v_1, \dots, v_4$ belong to $V(C(G_L))$ or to $V(C(G_R))$ (Lemma 7). Thus we obtain that $|V(C(G_L))| \leqslant |V(C(G))| - 8$ and $|V(C(G_R))| \leqslant |V(C(G))| - 8$. Further tightening of the upper bounds can be obtained by considering the status of edges $\{u_1, v_1\}$ and $\{u_3, v_3\}$. If both they are normal edges then v_1 and v_3 are removed from G_L and, as a result, w_1 and w_3 do not belong to $V(C(G_L))$ (Lemma 7) satisfying the first statement. Assume that both the edges are odd. Excluding of u_1 and u_3 on the second branch, leaves the vertices corresponding to edges $\{u_1, v_1\}$ and $\{u_3, v_3\}$ to be of degree 1. This causes removal of v_1 and v_3 from G_R (Lemma 5) and, as a result, removing of w_1 and w_3 from $V(C(G_R))$ (Lemma 7), which satisfies the second statement. Finally, if exactly one of $\{u_1, v_1\}$ and $\{u_3, v_3\}$ is odd, say the former w.l.o.g., then, arguing as in the previous case one gets that w_3 is removed from $V(C(G_L))$, while w_1 is removed from $V(C(G_R))$, satisfying the third statement. □

6.2. *R has exactly one odd edge*

We assume w.l.o.g. that this edge is $\{u_1, u_2\}$. Let t_3 and t_4 be vertices of G defined as follows. If the edge $\{u_3, v_3\}$ is a normal one then $t_3 = v_3$, otherwise t_3 is the vertex corresponding to the edge $\{u_3, v_3\}$. Vertex t_4 is defined analogously with respect to u_4 and v_4. If G has at least one edge with both ends in $OddComp(u_1) \cup \{t_3, t_4\}$ then return $FindIndep(G \setminus OddComp(u_1))$. Otherwise return the larger set of $OddComp(u_1) \cup \{t_3, t_4\} \cup FindIndep(G \setminus N^+(OddComp(u_1) \cup \{t_3, t_4\}))$ and $FindIndep(G \setminus OddComp(u_1))$. Let us justify the behavior of *FindIndep* in the considered case.

Lemma 13. *Assuming that the recursive calls to FindIndep return correct outputs, FindIndep returns a MIS of G in the considered case.*

Proof. Assume that $FindIndep(G \setminus OddComp(u_1))$ does not return a MIS of G. By Lemma 4, $OddComp(u_1)$ is a subset of a MIS S of G. Clearly, neither u_3 nor u_4 are contained in S. Assume that t_3 is *not* contained in S. It follows that exactly one neighbor of u_3 is contained in S, namely u_2. Consequently, $S \setminus \{u_2\} \cup \{u_3\}$ is another MIS of G. In other words, there is a MIS of G that does not contain *some* vertices of $OddComp(u_1)$. By Lemma 4, there is a MIS of G containing none of $OddComp(u_1)$, that is $FindIndep(G \setminus OddComp(u_1))$ returns a MIS of G in contradiction to our assumption. It follows that $t_3 \in S$. It can be shown analogously that $t_4 \in S$.

Assume now that G has at least one edge with both ends in $OddComp(u_1) \cup \{t_3, t_4\}$. Clearly, $OddComp(u_1) \cup \{t_3, t_4\}$ cannot be a subset of a MIS of G. It follows by the proven above that $FindIndep(G \setminus OddComp(u_1))$ returns a MIS of G. Also, according to the previous paragraph, if no two vertices of $OddComp(u_1) \cup \{t_3, t_4\}$ are neighbors, one of the two sets considered by $FindIndep(G)$ is a MIS of G. Thus, $FindIndep(G)$ returns a correct answer in the considered case. □

Now we compute upper bounds on the sizes of FNSes of G.

Lemma 14. *If in the considered case FindIndep(G) explores two branches then* $|V(C(G_L))| \leqslant |V(C(G))| - 12$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

Proof. Selection of u_1 and u_2 into the constructed MIS causes removal of u_3, u_4, v_1, v_2 from G_L and removal of w_1 and w_2 from $C(G_L)$ (Lemma 8). Also, v_3 and v_4 are removed from G_L (being themselves t_3 and t_4 or their neighbors), which implies that w_3 and w_4 are removed from $C(G_L)$ (Lemma 7). To summarize, none of the 12 vertices shown in Fig. 2 belongs to $V(C(G_L))$, from which follows that $|V(C(G_L))| \leqslant |V(C(G))| - 12$.

On the second branch, u_1 and u_2 are removed from G_R causing removal of vertices u_3, u_4, v_1, v_2 from $C(G_R)$ (Lemma 7). In total, $|V(C(G_R))| \leqslant |V(C(G))| - 6$. □

6.3. *Rectangle R has at least two odd edges having a common end*

Let $\{u_1, u_2\}$ and $\{u_2, u_3\}$ be these edges. In the considered case, the behavior of *FindIndep* is straightforward: it selects the larger set of $OddComp(u_1) \cup FindIndep(G \setminus N^+(OddComp(u_1)))$ and $FindIndep(G \setminus OddComp(u_1))$. The correctness follows from Corollary 1. The upper bounds on the sizes of FNSes of G are derived in the next lemma.

Lemma 15. *In the considered case* $|V(C(G_L))| \leqslant |V(C(G))| - 11$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 8$.

Proof. The vertices u_1, u_2, u_3 are selected on the first branch and thus cause removal of u_4, v_1, v_2, v_3 from $V(G_L)$ and removal of v_4, w_1, w_2, w_3 from $V(C(G_L))$ (Lemma 8). That is, all vertices in Fig. 2, except possibly w_4, are removed from $V(C(G_L))$ which means that $|V(C(G_L))| \leqslant |V(C(G))| - 11$.

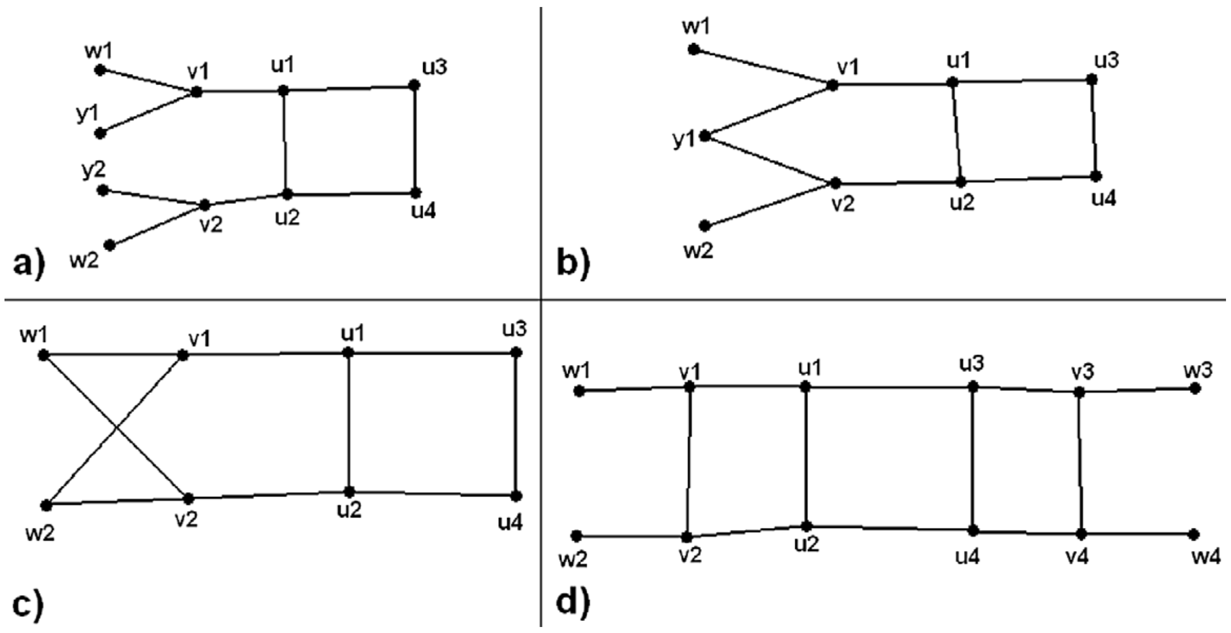

Fig. 3. A rectangle of $C(G)$ and surrounding vertices.

On the second branch, vertices u_1, u_2, u_3 are removed from G_R causing removal of v_1, v_2, v_3 from $C(G_R)$ (Lemma 7). Also, by Lemma 6, u_4 is removed from G_R and v_4 is removed from $C(G_R)$ (Lemma 7). In total,

$$\left|V\left(C(G_R)\right)\right| \leqslant \left|V\left(C(G)\right)\right| - 8. \qquad \square$$

6.4. *None of the considered cases regarding R is satisfied*

Clearly, this means that R has *exactly* two odd edges that do not have a common end. Let $\{u_1, u_2\}$ and $\{u_3, u_4\}$ be such edges.

First, *FindIndep*(G) checks whether there is another rectangle R' of $C(G)$ that falls to one of the previous cases. If such a rectangle R' is found, *FindIndep*(G) behaves with respect to R' as described in the previous subsection depending on the particular case suitable to R'.

If there is no rectangle R' mentioned in the previous paragraph, *FindIndep*(G) considers the rectangle R shown in Fig. 2 and checks whether v_1 or v_2 is adjacent to v_3 or v_4. In this case *FindIndep*(G) returns the larger set of *OddComp*$(u_1) \cup$ *FindIndep*$(G \setminus N^+(\mathit{OddComp}(u_1)))$ and *FindIndep*$(G \setminus \mathit{OddComp}(u_1))$.

If none of the previous cases is satisfied, *FindIndep*(G) checks whether there is a subgraph of G isomorphic to one of the graphs shown in Fig. 3(a), (b), (c). If such a subgraph is found, *FindIndep*(G) returns the larger set of *OddComp*$(u_1) \cup$ *FindIndep*$(G \setminus N^+(\mathit{OddComp}(u_1)))$ and *FindIndep*$(G \setminus \mathit{OddComp}(u_1))$, vertices are named according to the appropriate case shown in Fig. 3.

If none of the above cases is satisfied, *FindIndep*(G) picks a rectangle R with vertices named as shown in Fig. 2 and returns the larger set of *OddComp*$(v_1) \cup$ *FindIndep*$(G \setminus N^+(\mathit{OddComp}(v_1)))$ and *FindIndep*$(G \setminus \mathit{OddComp}(v_1))$.

The correctness of the behavior of *FindIndep*(G) described in this section follows from Corollary 1. Let us analyze upper bounds on the sizes of FNSes of G.

Lemma 16. *In the considered case, at least one of the following statements is true.*

(1) $|V(C(G_L))| \leqslant |V(C(G))| - 10$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 8$;
(2) $|V(C(G_L))| \leqslant |V(C(G))| - 8$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 10$;
(3) $|V(C(G_L))| \leqslant |V(C(G))| - 9$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 9$;
(4) $|V(C(G_L))| \leqslant |V(C(G))| - 12$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

Proof. If *FindIndep*(G) finds a rectangle satisfying the condition of one of the previous subsections, the lemma follows from Lemmas 12, 14, 15.

If v_1 or v_2 is adjacent to v_3 or v_4, we may assume w.l.o.g. that v_1 is adjacent to v_3. (Although the case where v_1 is adjacent to v_4 is not isomorphic to the considered one, the upper bounds on the sizes of FNSes of G are derived analogously.) Consider the first branch. Selection of u_1 and u_2 causes removal of v_1, v_2, u_3, u_4 from G_L and of w_1, w_2, v_3, v_4 from $C(G_L)$ (Lemma 8). Since u_3 and v_1 are removed from G_L, v_3 is also removed from G_L (Lemma 6) as well as w_3 is removed from $C(G_L)$ (Lemma 7). In total, we have shown that 11 vertices are removed from $C(G_L)$. In order to satisfy the last item in the statement of the lemma, we show that there is an additional twelfth vertex removed from $C(G_L)$. To this point consider vertex v_2. In Fig. 2, it is adjacent to u_2 and w_2. Since $C(G)$ is a cubic graph, there must be the third neighbor w of v_2 removed from $C(G_L)$ by Lemma 8. If w is not among the vertices shown in Fig. 2 or $w = w_4$, we are done. If $w = v_4$ then v_4 is removed from G_L by Lemma 6 causing w_4 to be removed from $C(G_L)$ by Lemma 7. Finally, it may be that $w = w_1$ or $w = w_3$, w.l.o.g. assume the former. Then w_1 is removed from G_L by Lemma 6. Observe that w_1 is adjacent to some vertex y which is not among the 11 vertices considered above because otherwise w_2, w_3, v_4 constitute a good cut separating vertices $u_1, \dots, u_4, v_1, \dots, v_3, w_1$. Vertex y removed from $C(G_L)$ by Lemma 7 is the desired twelfth vertex. Thus in any case $|V(C(G_L))| \leqslant |V(C(G))| - 12$. The statement for G_R is easily verified because removal of u_1 and u_2 from G_R causes removal of v_1, v_2, u_3, u_4 from $C(G_R)$ (Lemma 7).

Now assume that *FindIndep*(G) processes the case shown in Fig. 3(a). Note that in this case, the remaining neighbors v_3 and v_4 of u_3 and u_4 (not shown in the picture) do not coincide with w_1, w_2, y_1, y_2 because as a result we get the case considered in the previous paragraph. By Lemma 8, all vertices shown in Fig. 3(a) together with v_3 and v_4 are removed from $C(G_L)$, thus implying $|V(C(G_L))| \leqslant |V(C(G))| - 12$. The same argumentation as in the previous paragraph shows $u_1, \dots, u_4, v_1, v_2$ are removed from $C(G_R)$ satisfying the last item of the statement of the present lemma.

Assume now that the case shown in Fig. 3(b) is processed. Arguing as in the previous paragraph, we get that $|V(C(G_L))| \leqslant |V(C(G))| - 11$ and $|V(C(G_R))| \leqslant |V(C(G))| - 6$. In order to satisfy the last item in the statement of the lemma, we must show that there is additional vertex removed from $V(C(G_L))$. Note that by Lemma 6, y_1 is removed from G_L. If the third neighbor of y_1 is anyone besides w_1, w_2, v_3, v_4, we are done. Otherwise, there are vertices z_1, z_2, z_3, z_4, not shown in Fig. 3(b), which are neighbors of w_1, w_2, v_3, v_4, respectively: this can be verified by the argument using Hall's Theorem which was used to prove Lemma 10. Now, if, for example, y_1 is adjacent to w_1 then applying again Lemma 6, we see that w_1 is removed from G_L and z_1 is the desired twelfth vertex removed from $V(C(G_L))$ (Lemma 7).

Consider now the case shown in Fig. 3(c). We observe that the remaining neighbors v_3 and v_4 of u_3 and u_4 do not coincide with and do not adjacent to w_1 and w_2 in order to avoid appearance of a good cut of size at most 3 or a small connected component. Let y_1 and y_2 be the remaining neighbors of w_1 and w_2 respectively. These neighbors should be different on order to avoid a good cut of size 3. Observe that all the vertices in Fig. 3(c) together with y_1, y_2, v_3, v_4 are removed from $V(C(G_L))$: v_1, v_2 are removed from G_L by Lemma 8, they cause removal of w_1, w_2 from G_L by Lemma 6 which, in turn, causes removal of y_1 and y_2 from $C(G_L)$ (Lemma 7), u_3, u_4, v_3, v_4 are removed from $C(G_L)$ by Lemma 8. As in the previous paragraphs, $u_1, \dots, u_4, v_1, v_2$ are removed from $V(C(G_R))$. It follows that $|V(C(G_L))| \leqslant |V(C(G))| - 12$ and $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

Consider the remaining case. In this case v_1 is adjacent to v_2 in $C(G)$ because, up to isomorphism, all possible cases where v_1 is not adjacent to v_2 were considered in the previous paragraph. By the same reason v_3 is adjacent to v_4. Furthermore, the edge $\{v_1, v_2\}$ is odd because otherwise we would get a rectangle of a type considered in the previous subsections. The resulting configuration is shown in Fig. 3(d). Observe that removal of v_1 and v_2 produces a trivial graph which initiates the simplification process described in Section 2. As a result of this simplification, $u_1, \dots, u_4, v_3, v_4$ are removed from both G_L and G_R by the operations described in Section 2. Taking into account that w_1, w_2 are removed from both G_L and G_R by Lemma 7, the first statement of the lemma is satisfied. □

7. Processing of triangles

7.1. *There is a triangle of $C(G)$ with at least 2 odd edges*

Let u_1, u_2, u_3 be such a triangle of $C(G)$. Assume that there are two odd edges, say $\{u_1, u_2\}$ and $\{u_2, u_3\}$. Let t_1 and t_2 be the vertices of G corresponding to these edges. The vertices u_1, t_1, u_2, t_2, u_3 create a pentagon in G. At most 2 vertices of this pentagon can be included to a MIS of G and t_1, t_2 are the only two vertices which are not adjacent to any vertex outside the pentagon. It follows that there is a MIS of G including t_1 and t_2: in any other MIS, the vertices of the pentagon can be replaced by t_1 and t_2 without reducing the size of the set and without violating the nonadjacency property. It follows that the only branch needed on the triangle u_1, u_2, u_3 is removal of u_1, u_2, u_3. Similar reasoning applies to the case where all edges of the triangle are odd with the only difference that 3 vertices corresponding to the odd edges are taken to the MIS.

7.2. *Each vertex of $C(G)$ belongs to a triangle*

In this case *FindIndep*(G) picks an arbitrary vertex u_1 and returns the larger set of *OddComp*$(u_1) \cup$ *FindIndep*$(G \setminus N^+(\mathit{OddComp}(u_1)))$ and *FindIndep*$(G \setminus \mathit{OddComp}(u_1))$. The correctness of this behavior follows from Corollary 1. Let us compute the upper bounds on the sizes of FNSes of G.

Lemma 17. *In the considered case $|V(C(G_L))| \leqslant |V(C(G))| - 12$ and $|V(C(G_R))| \leqslant |V(C(G))| - 6$.*

Proof. We start from observing that u_1 participates in a subgraph of G isomorphic to the one shown in Fig. 4. Let u_1, u_2, u_3 be a triangle containing u_1. The remaining neighbors v_1, w_1, t_1 of u_1, u_2, u_3 are pairwise different because otherwise a

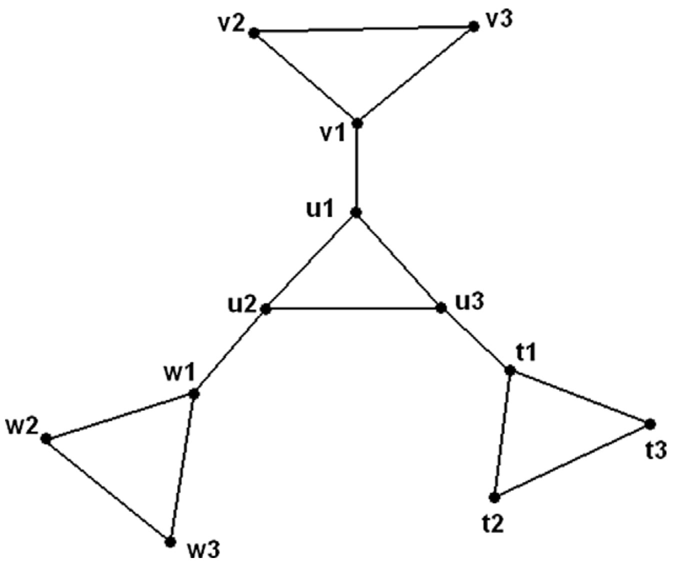

Fig. 4. A possible configuration of triangles in $C(G)$.

rectangle is occurs. Each of v_1, w_1, t_1 participates in a triangle. All this triangles are pairwise disjoint and disjoint from u_1, u_2, u_3 because otherwise either a rectangle or a vertex of degree 4 occurs. Thus we get the configuration shown in Fig. 4.

When u_1 is selected, u_2, u_3, v_1 are removed, the residual graph becomes trivial and the rest of vertices shown in Fig. 4 are removed from $C(G_L)$ by the process described in Section 2. On the second branch, to transform a trivial graph obtained as a result of removal of u_1 into a nontrivial one, vertices u_2, u_3, v_1, v_2, v_3 are removed from $C(G_R)$. $\square$

Each vertex being included into a triangle is a sufficient condition for the branching decisions shown in this section but not a necessary one. The same branching decision may be applied, if there is a triangle u_1, u_2, u_3 surrounded by other triangles as shown in Fig. 4. Hence in the rest of the subsection, we assume that such a subgraph does not occur in G, i.e. for each triangle u_1, u_2, u_3 there is an outside neighbor of one of the vertices which does not belong to a triangle.

7.3. The above cases do not hold

Let u_1, u_2, u_3 be an arbitrary triangle of $C(G)$. Let v_1 be a neighbor of u_1 which does not belong to a triangle. Let v_2, v_3 be the remaining neighbors of v_1 and let $w_1, \ldots, w_4$ be the remaining neighbors of v_2, v_3 as shown in Fig. 5(a)–(c). Since rectangles are forbidden in $C(G)$, all the vertices of $w_1, \ldots, w_4$ are pairwise different as well as no one of them coincides with u_2, u_3. On the same reason, v_2, v_3 do not coincide with u_2, u_3. Regarding the neighbors of u_2, u_3, there may be three different situations.

First, u_2, u_3 may be nonadjacent to any other vertex considered above. This situation is shown in Fig. 5(a) (here, vertices y_1 and y_2 must be different because otherwise a rectangle is created).

Second (the most involved subcase), exactly one of u_2, u_3 may be adjacent to some of $w_1, \ldots, w_4$. We may assume w.l.o.g. that u_3 is adjacent to w_4. This situation is shown in Fig. 5(b). Two subcases related to the remaining neighbor of w_4 are possible here. Vertex w_4 may be adjacent to vertex y_2 different from all the other vertices in the figure or w_4 may adjacent to one of w_1, w_2, say to w_2 but in this case w_2 must be adjacent to a vertex different from the vertices shown in the figure. All other possibilities of neighborhood of w_4 contradict to assumption of the section. In particular, w_4 cannot be adjacent to y_1 because a rectangle is created. Also, w_4 cannot be adjacent to w_3 because in this case y_1, v_2, and the remaining neighbor of w_3 create a good cut of size 3 of $C(G)$. Finally, if w_4 is adjacent to w_2, the latter has to be adjacent to an “outside” vertex because otherwise $\{w_1, w_3, y_1\}$ is a good cut of $C(G)$.

Third, both u_2 and u_3 may be adjacent to $w_1, \ldots, w_4$. It is not particularly important for the further discussion, *which* exactly vertices of $w_1, \ldots, w_4$ are adjacent to u_2, u_3. What important is that to avoid a good cut of size 3, those adjacent vertices should themselves be adjacent to vertices y_1 and y_2 (see Fig. 5(c)) that differ from all other vertices shown in the picture.

To describe the behavior of *FindIndep*(G), assume first the edge $\{u_1, v_1\}$ is odd. In this case, *FindIndep*(G) returns the larger set of *OddComp*$(v_1) \cup$ *FindIndep*$(G \setminus N^+($*OddComp*$(v_1)))$ and *FindIndep*$(G \setminus$ *OddComp*$(v_1))$. The correctness is justified by Corollary 1. Let us compute upper bounds on the sizes of FNSes of G. On the first branch, selection of u_1 and v_1 causes removal of u_2, u_3, v_2, v_3 from G_L, the rest of the vertices in Fig. 5(a) are removed from $C(G_L)$ (Lemma 8). In the situations shown in Figs. 5(b) and (c), additional explanation is needed regarding the vertices on the top part of the pictures. If u_3 is adjacent to w_4 then removal of u_3 and v_3 from G_L causes removal of w_4 from G_L (Lemma 6) and its outside neighbor y_2 (if such one occurs) is removed from $C(G_L)$ (Lemma 7). If w_4 is adjacent to w_2 then, by Lemma 6, removal of w_4 and v_2 from G_L causes removal of w_2 from G_L, its outside neighbor is removed from $C(G_L)$ (Lemma 7). Finally, if u_2 is adjacent

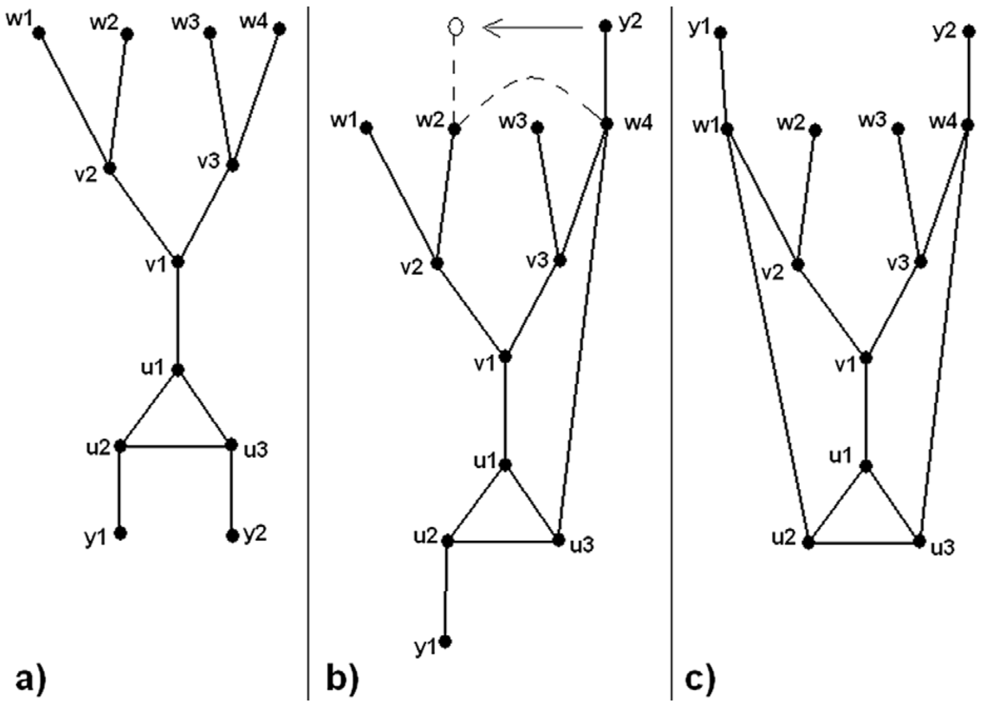

Fig. 5. A triangle of $C(G)$ and surrounding vertices.

to w_1 (relevant for Fig. 5(c)) then removal of v_2 and u_2 from G_L causes removal of w_1 from G_L and the removal of its neighbor y_1 from $C(G_L)$ (Lemmas 6 and 7). Thus we get that $|V(C(G_L))| \leqslant |V(C(G))| - 12$. On the second branch, removal of u_1 and v_1 from G_R causes removal of u_2, u_3, v_2, v_3 from $C(G_R)$ (Lemma 7), in total $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

If the edge $\{u_1, v_1\}$ is normal, our next assumption is that either $\{u_1, u_2\}$ or $\{u_1, u_3\}$ is an odd edge, it does not matter which one of them exactly, we assume that it is $\{u_1, u_2\}$. The behavior of *FindIndep*(G) in the considered case is analogous to the previous case, i.e. *FindIndep*(G) returns the larger set of *OddComp*$(v_1) \cup$ *FindIndep*$(G \setminus N^+($*OddComp*$(v_1)))$ and *FindIndep*$(G \setminus$ *OddComp*$(v_1))$. On the first branch, selection of v_1 removes v_2, v_3, u_1 from G_L. The vertex t of G corresponding to the edge $\{u_1, u_2\}$ remains of degree 1 as a result of removal of u_1 hence its neighbor u_2 is removed from G_L by Lemma 5. Removal of u_1 and u_2 causes removal of u_3 by Lemma 6. Further reasoning is analogous to the previous case. On the second branch we observe that after removal of v_1, the resulting graph becomes trivial because u_1 and the vertex t corresponding to the edge $\{u_1, u_2\}$ become two adjacent vertices of degree 2. The simplification process necessarily removes u_1, u_2, and u_3 from $C(G_R)$. Also, v_2 and v_3 are removed from $C(G_R)$ by Lemma 7. It follows that $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

It remains to consider the cases where the edge $\{u_1, v_1\}$ is normal and the triangle u_1, u_2, u_3 is normal or edge $\{u_2, u_3\}$ is odd. Here *FindIndep*(G) returns the larger set of *OddComp*$(v_1) \cup$ *FindIndep*$(G \setminus N^+($*OddComp*$(v_1)))$ and *FindIndep*$(G \setminus ($*OddComp*$(v_1) \cup \{u_2, u_3\}))$. Observe, u_1 participates in a MIS of $G \setminus$ *OddComp*(v_1), hence u_2 and u_3 can be safely removed on the second branch. Let us calculate the upper bounds on the sizes of MISes of G. On the first branch, $v_1, \dots, v_3, u_1, \dots, u_3, w_1, \dots, w_4$ are removed from $C(G_L)$ by Lemma 8. In total, $|V(C(G_L))| \leqslant |V(C(G))| - 10$. On the second branch, in addition to $u_1, \dots, u_3, v_1, \dots, v_3$, two remaining neighbors of u_2 and u_3 are removed from $C(G_R)$ (Lemma 7). For any case in Fig. 5, 8 vertices are removed in total.

Thus, we have proved the following lemma.

Lemma 18. *FindIndep*(G) *behaves correctly and at least one of the following two statements holds*:

(1) $|V(C(G_L))| \leqslant |V(C(G))| - 10$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 8$;
(2) $|V(C(G_L))| \leqslant |V(C(G))| - 12$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

8. $C(G)$ contains odd edges

In the rest of the paper we consider situations where $C(G)$ has no good cuts of size at most 3,[3] no rectangles and no triangles. In this situation any vertex u together with its neighbors and neighbors of their neighbors in $C(G)$ create a subgraph of $C(G)$ shown in Fig. 6: coinciding of any of w_i with any of other vertices causes appearance of a rectangle or a triangle. In this situation, Lemma 8 can be reformulated as follows.

[3] Formally, we considered good cuts of sizes 1 and 2 for G, not for $C(G)$, but it is not hard to observe that G has a good cut if and only if $C(G)$ has.

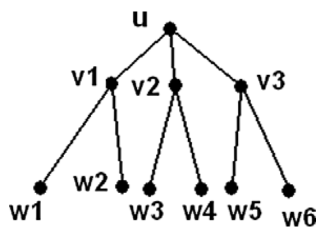

Fig. 6. A vertex u and surrounding vertices of $C(G)$ is case where $C(G)$ has no rectangles and no triangles.

Corollary 2. *Let G, G', and u be as in Lemma 8 with the assumption that G satisfies conditions of the present section. Then* $|V(C(G'))| \leqslant |V(C(G))| - 10$.

Proof. Immediately follows from Lemma 8 by taking into account Fig. 6. □

In this section we consider the case where $C(G)$ has odd edges. Let u be a vertex incident to an odd edge, assume w.l.o.g. that this edge is $\{u, v_1\}$ (in the rest of the algorithm's description, each time when we refer to vertex u, we assume that the vertices around u are named as in Fig. 6). $FindIndep(G)$ returns the larger set of $OddComp(u) \cup FindIndep(G \setminus N^+(OddComp(u)))$ and $FindIndep(G \setminus OddComp(u))$. Correctness of this behavior follows from Corollary 1. Let us compute the sizes of FNSes of G.

Lemma 19. *In the considered case* $|V(C(G_L))| \leqslant |V(C(G))| - 12$ *and* $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

Proof. By Corollary 2, all vertices shown in Fig. 6 are removed from $C(G_L)$. We have to find two additional vertices removed from that graph. Observe that $OddComp(u) = OddComp(v_1)$, hence, by Lemma 8, w_1, w_2 are removed from G_L and their neighbors are removed from $C(G_L)$. Consequently, if w_1 and w_2 are adjacent to at least two vertices not shown in Fig. 6, we are done. Consider what happens otherwise.

To avoid a good cut of size at most 3, $w_1, \dots, w_6$ are adjacent to at least 4 vertices $y_1, \dots, y_4$ not shown in Fig. 6. If w_1, w_2 are adjacent to none of them, 4 remaining edges incident to w_1, w_2 connect them to all of $w_3, \dots, w_6$ (otherwise a rectangle is created). By Lemma 6, all of $w_3, \dots, w_6$ are removed from G_L, hence their neighbors including all of $y_1, \dots, y_4$ are removed from $C(G_L)$ by Lemma 7.

Assume that w_1, w_2 are adjacent to exactly one of $y_1, \dots, y_4$, w.l.o.g. we may assume the existence of an edge $\{w_1, y_1\}$. The remaining three edges incident to w_1, w_2 connect them to 3 vertices of $w_3, \dots, w_6$, say w.l.o.g. to w_3, w_4, w_5, which in turn are incident to at least one of y_2, y_3, y_4 because otherwise y_2, y_3, y_4 are incident to w_6 increasing its degree to 4. All of w_3, w_4, w_5 are removed from G_L by Lemma 6 hence y_1 and a vertex of $y_2, \dots, y_4$ incident to w_3, w_4, w_5 are removed from $C(G_L)$ by Lemma 7.

To see that $|V(C(G_R))| \leqslant |V(C(G))| - 6$ observe that u, v_1 are removed from G_R hence v_2, v_3, w_1, w_2 are removed from $C(G_R)$ by Lemma 7. □

In the rest of the description of $FindIndep(G)$, graph $C(G)$ has no odd edges, hence $C(G) = G$ and there is no need to refer to $C(G)$ anymore.

9. Processing of good cut of size 4

In the present section we consider the behavior of $FindIndep(G)$ if there is a good cut of size 4. Let $y_1, \dots, y_4$ be such a good cut. Let $S = SmallVert(G \setminus \{y_1, y_2, y_3, y_4\})$. We may assume that each y_i is adjacent to 2 vertices outside of $S \cup \{y_1, \dots, y_4\}$ or that $|S| \geqslant 20$. Otherwise, applying the iterative replacement shown in the first paragraph of Section 5, the algorithm constructs a cut satisfying the desired condition. $FindIndep(G)$ returns the largest set among the following three: $\{y_1\} \cup FindIndep(G \setminus N^+(y_1))$, $\{y_2\} \cup FindIndep((G \setminus y_1) \setminus N^+(y_2))$, $FindIndep((G \setminus y_1) \setminus y_2)$. The correctness of such behavior is obvious. Let us compute the sizes of FNSes of G.

Lemma 20. *In the considered case* $|V(C(G_L))| \leqslant |V(C_G)| - 10$, $|V(C(G_M))| \leqslant |V(C_G)| - 17$, $|V(C(G_R))| \leqslant |V(C_G)| - 15$.

Proof. The inequality for $V(C(G_L))$ follows from Corollary 2.

When y_1 and y_2 are removed, S is separated from the rest of the graph by a cut of size 2. Hence, in order to avoid a good cut of size 2, no vertex of S belongs to G_M and to G_R. Consequently, there is nothing to prove if $|S| \geqslant 20$. We assume that the other condition is satisfied, i.e. each of y_i has two neighbors outside of the set $S \cup \{y_1, \dots, y_4\}$. Let z_1, z_2 be these 2 neighbors of y_2 and let $t_1, \dots, t_4$ be the remaining neighbors of z_1, z_2. To avoid occurrence of previously considered cases of the proposed algorithm, none of $z_1, z_2, t_1, \dots, t_4$ coincide. Also, none of $t_1, \dots, t_4$ belongs to S because S is separated from the rest of the vertices by $y_1, \dots, y_4$ and z_1, z_2 do not belong to $S \cup \{y_1, \dots, y_4\}$. Vertices $y_2, z_1, z_2, t_1, \dots, t_4$ are removed from $C(G_M)$ by Lemma 8. Since $y_2, z_1, z_2, t_1, \dots, t_4$ is disjoint from S, $|\{y_2, z_1, z_2, t_1, \dots, t_4\} \cup S| = |\{y_2, z_1, z_2, t_1, \dots, t_4\}| + |S| \geqslant 17$, hence $|V(C(G_M))| \leqslant |V(C_G)| - 17$.

On the last branch vertices z_1 and z_2 are removed from $C(G_R)$ by Lemma 7. Observe that $|S \cup \{y_1, y_2, z_1, z_2\}| \geqslant 14$, hence at least 14 vertices are removed from $C(G_R)$ while we need 15 ones. Consider two neighbors w_1, w_2 of y_1 lying outside of $S \cup \{y_1, \dots, y_4\}$ and removed from $C(G_R)$ by Lemma 7. To avoid a rectangle at least one of w_1, w_2 (say w_1) does not coincide with z_1 nor with z_2. Vertex w_1 is the desired 15th vertex removed from $C(G_R)$. □

10. No rectangles, no triangles, no good cuts, no odd edges

In order to proceed, we extend our notation. For a vertex $u \in V(G)$, we denote by $L(G, u, i)$ the set of vertices lying at distance i from u. For example, in Fig. 6, $L(G, u, 0) = \{u\}$, $L(G, u, 1) = \{v_1, \dots, v_3\}$, $L(G, u, 2) = \{w_1, \dots, w_6\}$. The vertices of $L(G, u, 3)$ can be classified into 3 categories. Those that are adjacent to exactly one vertex of $L(G, u, 2)$ are called *single* vertices with respect to u. Accordingly, there are also *double* and *triple* vertices with respect to u.

The first subcase considered by *FindIndep*(G) occurs, if there exists a vertex u such that there is a triple vertex w with respect to u.

In this case *FindIndep*(G) returns the larger set of $\{u\} \cup$ *FindIndep*$(G \setminus N^+(u))$ and $\{w\} \cup$ *FindIndep*$(G \setminus (N^+(w)))$. The correctness of this behavior is justified by the following lemma.

Lemma 21. *Let $u, w \in V(G)$ and assume that w is a triple vertex with respect to u. Then there is a MIS of G that contains u or w.*

Proof. Assume that no MIS of G contains u. Then any MIS of G contains at least two neighbors of u. Let v_1 and v_2 be two neighbors of u contained in some MIS of G. In other words, a MIS of $G' = G \setminus N^+(\{v_1, v_2\})$ united with $\{v_1, v_2\}$ is a MIS of G. The lemma will follow if we show that w belongs to a MIS of G'.

To this end observe that w does not have two common neighbors with any neighbor of u because otherwise a cycle of length 4 is induced. Taking into account that w is a triple vertex, it follows that w has exactly one common neighbor with each neighbor of u. Hence w has degree 1 in G' and clearly belongs to at least one MIS of G'. □

Lemma 22. *In the considered case $|V(C(G_L))| \leqslant |V(C(G))| - 10$ and $|V(C(G_R))| \leqslant |V(C(G))| - 10$.*

Proof. Immediately follows from Corollary 2. □

The following lemma is necessary for correctness proof of *FindIndep*(G) for the rest of the cases.

Lemma 23. *There is a MIS of G containing u or there is a MIS of G containing v_1 or there is a MIS of G containing w_1, w_2, v_2, v_3, and t_1, t_2 as in Fig. 7, the latter two vertices are included only if v_1 belongs to a pentagon as shown in Fig. 7.*

Proof. Assume that no MIS of G contains u or v_1 and let S be a MIS of G. S necessarily contains two neighbors of u. Since $v_1 \notin S$, both v_2 and v_3 belong to S. Observe further that both w_1 and w_2 belong to S. Otherwise, v_1 can be added to S in contradiction to its maximality or v_1 can replace a single vertex of $\{w_1, w_2\}$ that belongs to S creating a MIS of G containing v_1.

If v_1 belongs to the pentagon as shown in Fig. 7 then both t_1 and t_2 belong to S. Indeed, assume that, for example, t_1 does not belong to S. Taking into account that both w_1 and w_2 belong to S, w_1 is the only neighbor of z_1 contained in S. Hence w_1 may be replaced by z_1 in contradiction to our conclusion done the previous paragraph that if no MIS of G contains u or v_1 then any MIS of G contains both w_1 and w_2. □

The rest of the section is divided into three subsections describing the behavior of *FindIndep*(G) when certain conditions are satisfied. As usually, it is assumed for the second and the third subsections that the conditions of the earlier subsections are not satisfied.

10.1. *There is a vertex u such that $G[L(G, u, 2)]$ contains isolated vertices and ($|L(G, u, 3)| \geqslant 9$ or $|L(G, u, 3)| + |L(G, u, 4) \cap L(G, v, 3)| \geqslant 11$ for some neighbor v of u)*

Let v_1 be a neighbor of u such that $|L(G, v_1, 3) \cap L(G, u, 4)|$ is the largest possible. *FindIndep*(G) returns the largest set among S_1, S_2, and S_3 computed as follows. $S_1 \leftarrow \{u\} \cup$ *FindIndep*$(G \setminus N^+(u))$, $S_2 \leftarrow \{v_1\} \cup$ *FindIndep*$(G \setminus N^+(v_1))$, $S_3 \leftarrow \{w_1, w_2, v_2, v_3\} \cup$ *FindIndep*$(G \setminus (\{v_1\} \cup N^+(\{w_1, w_2, v_2, v_3\})))$. The correctness of the above behavior follows from Lemma 23.

Lemma 24. *In the considered case $|V(C(G_L))| \leqslant |V(C(G))| - 10$, $|V(C(G_M))| \leqslant |V(C(G))| - 10$, and $|V(C(G_R))| \leqslant |V(C(G))| - 21$.*

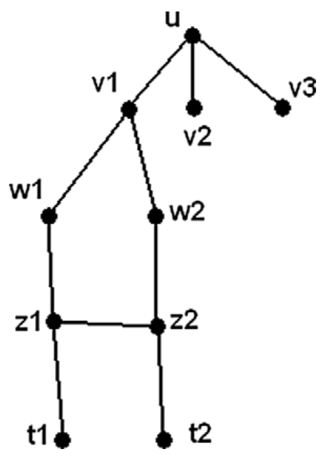

Fig. 7. A possible subgraph of $C(G)$.

Proof. The statement regarding $V(C(G_L))$ and $V(C(G_M))$ immediately follow from Corollary 2. The set of vertices removed from $C(G_R)$ is $L(G,u,0) \cup L(G,u,1) \cup L(G,u,2) \cup L(G,u,3) \cup (L(G,u,4) \cap L(G,v_1,3))$ (Lemma 8). All the sets participating in the union are disjoint, the first 3 sets are those shown in Fig. 6 and contain totally 10 elements. The statement regarding $|V(C(G_R))|$ is obviously true, if $|L(G,u,3)|+|L(G,u,4) \cap L(G,v,3)| \geqslant 11$ for *at least one* neighbor v of u, because $|L(G,u,3) \cup (L(G,u,4) \cap L(G,v_1,3))|$ is the largest possible among *all* the neighbors of u. If $|L(G,u,3)| \geqslant 9$ is satisfied, we note that $|L(G,u,4)| \geqslant 5$ because otherwise a good cut is produced. On the other hand, $L(G,u,4)$ is partitioned into $L(G,u,4) \cap L(G,v_i,3)$ for $i=1,2,3$, $L(G,u,4) \cap L(G,v_1,3)$ being the largest. Hence $|L(G,u,4) \cap L(G,v_1,3)| \geqslant 2$ and we get again that $|L(G,u,3) \cup (L(G,u,4) \cap L(G,v_1,3))| \geqslant 11$ which implies validity of the statement regarding $V(C(G_R))$. □

Note that in this subsection we do not explore the condition that $G[L(G,u,2)]$ has isolated vertices. This condition will be explored in the final part of the complexity analysis.

10.2. *There is a vertex u such that the graph induced by $L(G,u,2)$ has no edges*

An alternative formulation of the considered case is that there are 12 edges between $L(G,u,2)$ and $L(G,u,3)$ for some $u \in V(G)$. In this case *FindIndep*(G) returns the largest set among S_1, S_2, and S_3 computed as follows. $S_1 \leftarrow \{u\} \cup$ *FindIndep*$(G \setminus N^+(u))$, $S_2 \leftarrow \{v_1\} \cup$ *FindIndep*$(G \setminus N^+(v_1))$, $S_3 \leftarrow \{w_1,w_2,v_2,v_3\} \cup$ *FindIndep*$(G \setminus (\{v_1\} \cup N^+(\{w_1,w_2,v_2,v_3\})))$ where v_1 is the neighbor of u such that the number $p(v_1)$ of vertices of $L(G,u,4)$ removed from $V(C(G_R))$ is the largest possible. The correctness follows from Lemma 23.

Lemma 25. *In the considered case $|V(C(G_L))| \leqslant |V(C(G))|-10$, $|V(C(G_M))| \leqslant |V(C(G))|-10$, and $|V(C(G_R))| \leqslant |V(C(G))|-21$.*

Proof. The statement regarding $V(C(G_L))$ and $V(C(G_M))$ immediately follows from Corollary 2. The following sets of vertices are removed from $V(C(G_R))$ by Lemma 8: $L(G,u,0),\dots,L(G,u,3)$, $L(G,v_1,3) \cap L(G,u,4)$ (the latter set of vertices as being neighbors of neighbors of w_1 and w_2). Observe also, the double vertices of $L(G,u,3)$ are removed from $V(G_R)$ independently on the choice of v_1 (Lemma 6), hence their neighbors in $L(G,u,4)$ are removed from $V(C(G_R))$ (Lemma 7). Taking into account these observations, we show that $L(G,u,3)$ together with the vertices of $L(G,u,4)$ removed from $V(C(G_R))$ are at least 11 vertices which, together with 10 vertices of $L(G,u,0) \cup L(G,u,1) \cup L(G,u,2)$ constitute the desired 21 vertices removed from $V(C(G_R))$.

Taking into account that there are 12 edges between $L(G,u,2)$ and $L(G,u,3)$ and that there are no triple vertices in $L(G,u,3)$, it follows that $|L(G,u,3)| \geqslant 6$. If $|L(G,u,3)|=6$, all vertices of $L(G,u,3)$ are double, hence *all* vertices of $L(G,u,4)$ are removed from $C(G_R)$ independently on the choice of v_1. Note that $|L(G,u,4)| \geqslant 5$, otherwise a good cut occurs. Hence $L(G,u,3) \cup L(G,u,4)$ constitute the desired 11 vertices in the considered case.

Consider now the case of $|L(G,u,3)|=7$. In this case 5 vertices of $L(G,u,3)$ are double ones (taking into account 12 edges connecting $L(G,u,2)$ and $L(G,u,3)$). Let C be the subset of $L(G,u,4)$ adjacent to these double vertices. Observe that $|C| \geqslant 3$ because otherwise C together with the single vertices of $L(G,u,3)$ constitute a good cut. If $|C| \geqslant 4$ then C together with the 7 vertices of $L(G,u,3)$ constitute the desired 11 vertices removed from $V(C(G_R))$ independently on the choice of v_1. If $|C|=3$, we show that there is a neighbor v_1 of u such that $p(v_1) \geqslant 4$. To this end, take a vertex $w \in L(G,u,4) \setminus C$ (there must be such a vertex since $|L(G,u,4)| \geqslant 5$, see the previous paragraph). This vertex w is adjacent to a single vertex $y \in L(G,u,3)$, which is, in turn, adjacent to a vertex $w_1 \in L(G,u,2)$. Let v_1 be the neighbor of w_1 in $L(G,u,1)$. Observe that if v_1 is selected by *FindIndep*(G) as the neighbor of u then $|C| \cup \{w\}$ are removed from $V(C(G_R))$. In other words, $p(v_1) \geqslant 4$.

Assume now that $|L(G,u,3)|=8$. In this case we have to show that there is a neighbor v_1 of u with $p(v_1) \geqslant 3$. If there is a neighbor v_1 of u with $|L(G,v_1,3) \cap L(G,u,4)| \geqslant 3$, the statement follows immediately. Otherwise, taking into account that $|L(G,u,4)| \geqslant 5$ and each element of $L(G,u,4)$ belongs to $L(G,v,3)$ for some neighbor v of u, there are two neighbors v_1 and v_2 such that $L(G,v_1,3) \cap L(G,u,4)$ is disjoint with $L(G,v_2,3) \cap L(G,u,4)$ and the size of each of these sets is 2. Besides that, $L(G,u,3)$ contains 4 double vertices and least one of them has a neighbor y in $L(G,u,4)$ (otherwise, single vertices of $L(G,u,3)$ constitute a good cut). This vertex y is removed from $C(G_R)$ independently on the choice of v_1 (see the first paragraph of the proof) and it *does not* belong to at least one of $L(G,v_1,3) \cap L(G,u,4)$ or $L(G,v_2,3) \cap L(G,u,4)$ (remember these sets are disjoint!), let us say to the former one. It follows that $p(v_1) \geqslant 3$. Finally, note that the case where $|L(G,u,3)| = 9$ was analyzed in the previous subsection and hence cannot happen in the considered case according to our assumption. □

10.3. No one of the above cases happens

In this case *FindIndep*(G) branches on a vertex satisfying a particular condition. We define this condition and then prove that a vertex satisfying this condition exists.

We say that a vertex u of G is *adjacent to a pentagon* if there is a neighbor v of u participating in a cycle of size 5, which does not include u and the two vertices of this cycle, which are not neighbors of v belong to $L(G,u,3)$; in this case we also say that this cycle *certifies* u.

We say that vertex $u \in V(G)$ is *good* if it is adjacent to a pentagon and $G[L(G,u,2)]$ contains isolated vertices.

Theorem 1. *In the considered case there is at least one good vertex in G.*

Proof. Assume first that for any $u \in V(G)$, $G[L(G,u,2)]$ contains isolated vertices. In this case it remains to find a vertex adjacent to a pentagon. Pick an arbitrary vertex u. If it *is* adjacent to a pentagon, we are done. Otherwise consider the graph $G' = G[L(G,u,2)]$. Since the condition of the previous subsection is not satisfied, G' contains at least one edge.

Assume that G' has an isolated edge, i.e. an edge whose ends do not incident to any other edge. This situation is shown in Fig. 8(a), where the isolated edge is $\{w_2,w_3\}$. We emphasize that there may be additional edges between vertices of $L(G,u,2)$ but not incident to w_2 nor to w_3. We show that vertex v_3 is adjacent to a pentagon. Indeed, consider the pentagon $\{u,v_1,v_2,w_2,w_3\}$. The only condition that needs to be verified is that both w_2 and w_3 belong to $L(G,v_3,3)$. But assuming otherwise implies an edge between $\{w_2,w_3\}$ and $\{w_5,w_6\}$ in contradiction to our assumption that $\{w_2,w_3\}$ is an isolated edge. Note that we considered the only possible case of occurrence of an isolated edge up to isomorphism (for example, w_1 and w_2 are not adjacent because otherwise a triangle is created).

Assume now that G' does not contain an isolated edge but contains exactly two edges. In this case the configuration shown in Fig. 8(b) is the only possible one up to isomorphism (others induce short cycles). Note that this time the edges shown in the picture are the only ones that occur in G'. Observe that in the considered case, vertex w_1 is adjacent to a pentagon, which is certified by pentagon $\{v_1,u,v_2,w_3,w_2\}$. Indeed, to force v_2 or w_3 to be in $L(G,w_1,2)$, either edge $\{w_1,w_4\}$ or edge $\{w_1,w_5\}$ must occur, in contradiction to our assumption that there are only two edges in G'.

Assume now that G' does not contain an isolated edge and contains exactly three edges. Then the only possible configuration up to isomorphism is shown in Fig. 8(c). Other non-isomorphic configurations are unsuitable because they induce short cycles. Using argumentation analogous to the previous case, we can show that w_1 is adjacent to a pentagon.

Observe that we have considered all possible configurations of G': if G' has at least four edges then $L(G,u,3)$ has at most four vertices constituting a good cut.

Assume now that there is a vertex $u \in V(G)$ such that $G[L(G,u,2)]$ does not have isolated vertices. As we pointed out in the previous paragraph, $G[L(G,u,2)]$ cannot have 4 or more edges, hence it has exactly 3 edges none of which share a common end. Moreover, two vertices of $L(G,u,2)$ adjacent to the same neighbor of u cannot be themselves adjacent: it causes existence of a triangle. It follows that the placement of the edges shown in Fig. 9 is only possible up to isomorphism. Note that any neighbor of u is adjacent to a pentagon. For example, v_1 is certified by a pentagon u, v_2, w_4, w_5, v_3. Consequently, if for at least one neighbor v of u, $G[L(G,v,2)]$ has isolated vertices, we are done. Consider what happens otherwise.

Consider vertex v_1. Among the vertices shown in Fig. 9(a), vertices v_2, w_3, v_3, w_6 belong to $L(G,v_1,2)$. Given the edges $\{v_2,w_3\}$ and $\{v_3,w_6\}$, the remaining two vertices y_1 and y_2 of $L(G,v_1,2)$ (one is the neighbor of w_1, the other is the neighbor of w_2) must be adjacent in order to satisfy the pattern described in the previous paragraph. Arguing analogously, we get that y_3 and y_4, the remaining neighbors of w_3, w_4, must be adjacent in order to ensure that $G[L(G,v_2,2)]$ does not have isolated vertices as well as y_5 and y_6, the remaining neighbors of w_5 and w_6, must be adjacent in order to ensure that $G[L(G,v_3,2)]$ does not have isolated vertices. Let us show that vertices $y_1,\dots,y_6$ are pairwise different. There are a number of ways to show this. For example, assume that y_2 coincides with y_3. Then to avoid y_2 to have degree 4, y_4 has to coincide with y_1. However, in this case, y_1, y_2, y_5, y_6 constitute a good cut. Arguing analogously we get that y_1 and y_2 differ from y_5, y_6 as well as that y_3, y_4 differ from y_5, y_6. The vertices shown in Fig. 9(a) together with $y_1,\dots,y_6$ create a subgraph of G shown in Fig. 9(b).

Observe further that each of $w_1,\dots,w_6$ is adjacent to a pentagon. For example, w_1 is certified by v_1, u, v_2, w_3, w_2. It follows that if for at least one w_i, $G[L(G,w_i,2)]$ contains isolated vertices, we are done. We will show that otherwise we get a good cut, which will finish the proof of the present theorem.

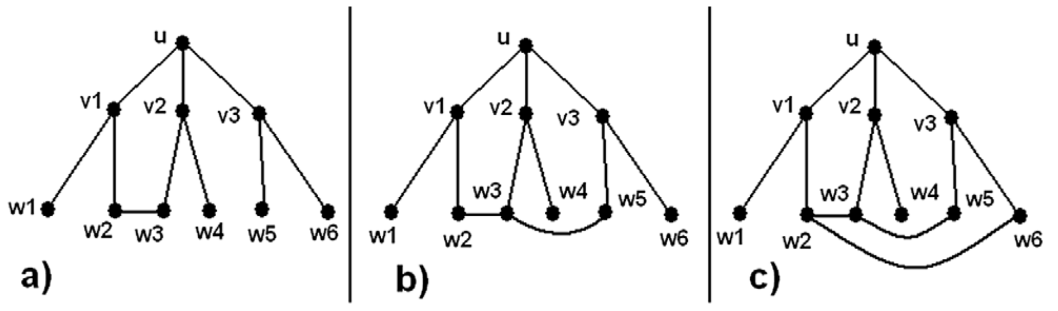

Fig. 8. Illustration of proof of Theorem 1.

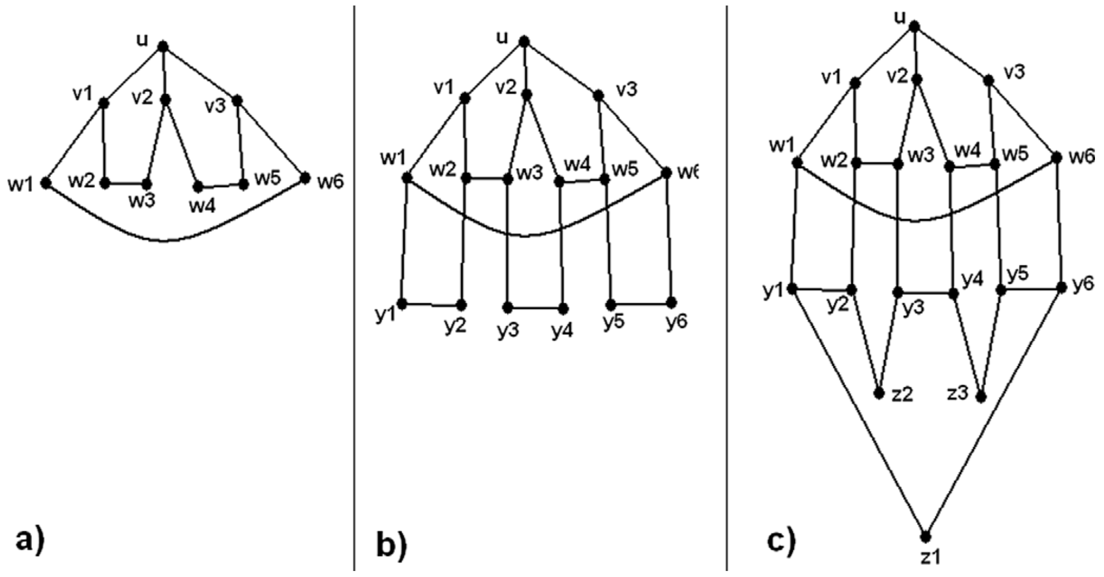

Fig. 9. Illustration of proof of Theorem 1.

Consider vertex w_1. The vertices of $L(G, w_1, 2)$ shown in Fig. 9(b) are u, v_3, w_2, y_2, y_6. Vertex y_6 is the only one which is not engaged into adjacency with other vertices shown in the picture. To ensure that $G[L(G, w_1, 2)]$ does not have isolated vertices the remaining neighbor z_1 of y_1 must be adjacent to y_6. Arguing similarly regarding w_2 and w_5 we get that y_2 and y_3 must have a common third neighbor z_2 as well as y_4 and y_5 have a common third neighbor z_3. Vertices z_1, z_2, z_3 constitute a good cut an example of which is illustrated in Fig. 9(c). □

Now we are ready to present the behavior of *FindIndep*(G) in the considered case. *FindIndep*(G) picks a good vertex u whose existence is guaranteed by Theorem 1. Let v_1, w_2, z_2, z_1, w_1 be the pentagon certifying u (consider Fig. 7). If $\{v_2, v_3, w_1, w_2, t_1, t_2\}$ is an independent set then *FindIndep*(G) returns the largest set among $\{u\} \cup$ *FindIndep*$(G \setminus N^+(u))$, $\{v_1\} \cup$ *FindIndep*$(G \setminus \{v_1\})$, and $\{v_2, v_3, w_1, w_2, t_1, t_2\} \cup$ *FindIndep*$(G \setminus (\{v_1\} \cup N^+(\{v_2, v_3, w_1, w_2, t_1, t_2\})))$. Otherwise the only difference is that the last branch is not considered. The correctness of such behavior follows from Lemma 23.

Let us prove upper bound on the sizes of FNSes of G.

Lemma 26. *In the considered case, if G has 3 FNSes then $|V(C(G_L))| \leqslant |V(C(G))| - 10$, $|V(C(G_M))| \leqslant |V(C(G))| - 10$, and $|V(C(G_R))| \leqslant |V(C(G))| - 21$. Otherwise $|V(C(G_L))| \leqslant |V(C(G))| - 10$ and $|V(C(G_R))| \leqslant |V(C(G))| - 10$.*

Proof. If G has 2 FNSes then the statement of the lemma immediately follows from Corollary 1. In the rest of the proof we assume that G has 3 FNSes.

The statement regarding $V(C(G_L))$ and $V(C(G_M))$ immediately follows from Corollary 1. To prove the statement regarding G_R we consider the sizes of $L(G, u, 3)$ from 5 to 8. By Lemma 8, $V(C(G_R))$ does not contain vertices of $L(G, u, i)$ for i from 0 to 3 as well as $N^+(N^+(\{t_1, t_2\}))$. Denote $L(G, u, 0) \cup L(G, u, 1) \cup L(G, u, 2) \cup L(G, u, 3)$ by MB (abbreviation of Main Block). There are 10 vertices in $L(G, u, 0) \cup L(G, u, 1) \cup L(G, u, 2)$. Hence, to prove the statement regarding $C(G_R)$, it is enough to show that the number of vertices removed from $C(G_R)$ outside of MB is at least $11 - |L(G, u, 3)|$. In most cases we will show that these additional vertices are contained in $N^+(N^+(\{t_1, t_2\}))$ and only once in $N^+(N^+(\{w_1, w_2\}))$.

In the rest of the proof we denote by $SN(t_1, t_2)$ the set $N(N(\{t_1, t_2\})) \setminus (N^+(\{t_1, t_2\}))$. In other words, $SN(t_1, t_2)$ consists of vertices lying at distance 2 from t_1 or from t_2 and does not include t_1, t_2, and their neighbors. Note that $\{t_1, t_2\}$, $N(\{t_1, t_2\})$, and $SN(t_1, t_2)$ are 3 disjoint sets, their union equals $N^+(N^+(\{t_1, t_2\}))$.

Assume that $|L(G,u,3) = 5|$. Then both t_1 and t_2 do not belong to MB. Also $|N(\{t_1,t_2\}) \setminus MB| \geqslant 2$ and $|SN(t_1,t_2) \setminus MB| \geqslant 2$. Indeed, violation of anyone of these condition implies existence of a good cut. For example, if $|SN(t_1,t_2) \setminus MB| < 2$ then $SN(t_1,t_2) \setminus MB$ together with $L(G,u,3) \setminus \{z_1,z_2\}$ constitute the above good cut. In total, $|N^+(N^+(\{t_1,t_2\})) \setminus MB| \geqslant 6$, as required.

Assume that $|L(G,u,3)| = 6$. Here we consider the cases where both t_1 and t_2 do not belong to MB and where one of them, say t_2, belongs to MB. The case where both t_1 and t_2 belong to $L(G,u,3)$ cannot occur because otherwise existence of a good cut would follow.

Assume that both t_1 and t_2 do not belong to $L(G,u,3)$. Then, to avoid a good cut, we have to assume that $|N(\{t_1,t_2\}) \setminus MB| > 0$ and that $|SN(t_1,t_2) \setminus MB| > 0$. Thus we are guaranteed to have at least 4 vertices apart from MB. In the problematic case where $|N^+(N^+(\{t_1,t_2\})) \setminus MB| = 4$ (it happens if $|N(\{t_1,t_2\}) \setminus MB| = |SN(t_1,t_2) \setminus MB| = 1$), there are at least two edges incident to $\{t_1,t_2\}$ such that the other ends of these edges can belong only to $L(G,u,3) \setminus \{z_1,z_2\}$. Let $z \in L(G,u,3) \setminus \{z_1,z_2\}$ be a vertex incident to one of these edges. Note that z is incident to a vertex y outside of MB and distinct from the vertices of $N^+(N^+(\{t_1,t_2\})) \setminus MB$ considered before: otherwise $L(G,u,3) \setminus \{z_1,z_2,z\} \cup SN(t_1,t_2)$ is a good cut. Since vertex y belongs to $SN(t_1,t_2) \setminus MB$, $|SN(t_1,t_2) \setminus MB| \geqslant 2$, a contradiction.

Assume that $t_2 \in MB$. Then $|N(\{t_1,t_2\}) \setminus MB| \geqslant 2$ as well as $|SN(t_1,t_2) \setminus MB| \geqslant 2$. Otherwise, a good cut is created. For example, if $|SN(t_1,t_2) \setminus MB| = 1$ then vertex $w \in SN(t_1,t_2) \setminus MB$ together with the 3 vertices of $L(G,u,3) \setminus \{z_1,z_2,t_2\}$ constitute a good cut of size 4. Vertex t_1 together with at least 2 vertices of $N(\{t_1,t_2\}) \setminus MB$ and at least 2 vertices of $SN(t_1,t_2) \setminus MB$ constitute the desired 5 vertices removed from $V(C(G_R))$ besides MB.

Assume that $|L(G,u,3)| = 7$. Assume first that both t_1 and t_2 do not belong to MB. Taking into account that only 4 vertices of $N^+(N^+(\{t_1,t_2\}))$ apart from MB are needed in the considered case, there is nothing to prove if both $N(\{t_1,t_2\}) \setminus MB$ and $SN(t_1,t_2) \setminus MB$ are nonempty or some of these sets includes at least 2 elements.

Assume that $N(\{t_1,t_2\}) \setminus MB = \emptyset$. Then at least 3 edges connect t_1, t_2 to at least 2 vertices z_3, z_4 of $L(G,u,3) \setminus \{z_1,z_2\}$. Observe that there are vertices y_3, y_4 outside of MB and different from t_1, t_2 such that y_3 is adjacent to z_3 and y_4 is adjacent to z_4. Otherwise, $L(G,u,3) \setminus \{z_1,\dots,z_4\}$ together with at most one existing vertex of y_3, y_4 constitute a good cut. Vertices y_3, y_4 belong to $SN(t_1,t_2)$, thus we get the desired 4 vertices of $N^+(N^+(\{t_1,t_2\}))$ outside of MB.

If we assume that $|N(\{t_1,t_2\}) \setminus MB| = 1$ but $SN(t_1,t_2) = \emptyset$, the analogous argument helps us to get a contradiction. To complete the degree of t_1 and t_2, they have at least one neighbor $z_3 \in L(G,u,3) \setminus \{z_1,z_2\}$. This vertex must be in incident to a vertex y_3 outside MB which is different from the 3 vertices considered before, otherwise $L(G,u,3) \setminus \{z_1,z_2,z_3\}$ constitute a good cut (since we assume $SN(t_1,t_2) \setminus MB = \emptyset$, the only vertex of $N(\{t_1,t_2\}) \setminus MB$ is not connected "outwards"). This vertex y_3 belongs to $SN(t_1,t_2) \setminus MB$ in contradiction to our assumption.

Consider the case where exactly one of t_1, t_2, say t_2, belongs to $L(G,u,3)$. In this case, to avoid $L(G,u,3) \setminus \{z_1,z_2,t_2\}$ to be a good cut, we get that both $N(\{t_1,t_2\}) \setminus MB$ and $SN(t_1,t_2) \setminus MB$ are nonempty. If at least one of these sets is of size 2, we are done. Otherwise, we derive a contradiction. In particular, to complete the degree 3 of t_1, it is adjacent to a vertex z_4 of $L(G,u,3) \setminus \{z_1,z_2,t_2\}$. This vertex z_4 must be incident to a vertex y_4 outside of MB, of $N(\{t_1,t_2\}) \setminus MB$, and of $SN(t_1,t_2) \setminus MB$ because otherwise the only vertex of $SN(t_1,t_2) \setminus MB$ together with $L(G,u,3) \setminus \{z_1,z_2,t_2,z_4\}$ constitute a good cut. Vertex y_4 is the second vertex of $SN(t_1,t_2) \setminus MB$, in contradiction to our assumption.

Finally, consider the case where $\{t_1,t_2\} \subset L(G,u,3)$. In this case both $N(\{t_1,t_2\}) \setminus MB$ and $SN(t_1,t_2) \setminus MB$ contain at least 2 vertices each one. Otherwise if any of these sets consists of at most one vertex w then w (if exists) together with $L(G,u,3) \setminus \{z_1,z_2,t_1,t_2\}$ constitute a good cut.

Assume that $|L(G,u,3) = 8|$. Assume first that neither t_1 nor t_2 belong to MB. If at least one neighbor of t_1 and t_2 does not belong to $L(G,u,3)$, we are done. Otherwise, 3 edges connect t_1 and t_2 to at least 2 vertices z_3, z_4 of $L(G,u,3) \setminus \{z_1,z_2\}$. To avoid $L(G,u,3) \setminus \{z_1,z_2,z_3,z_4\}$ to be a good cut, z_3, z_4 must be adjacent to at least one vertex y outside of MB which is different from t_1, t_2. Since $y \in SN(t_1,t_2)$, it is the desired third vertex removed from $V(C(G_R))$.

Assume now that exactly one of $\{t_1,t_2\}$ (say, t_2) belongs to $L(G,u,3)$. Assume first that $N(\{t_1,t_2\}) \setminus MB = \emptyset$. In this case t_1, in order to be of degree 3, is incident to 2 vertices z_4, z_5 of $L(G,u,3) \setminus \{z_1,z_2,t_2\}$. These vertices must be incident to 2 vertices y_4, y_5 outside of MB, which are different from t_1: otherwise $L(G,u,3) \setminus \{z_1,z_2,t_2,z_4,z_5\}$ and together with at most one vertex of y_4, y_5 constitute a good cut. Vertices y_4, y_5 (which belong to $SN(t_1,t_2) \setminus MB$) together with t_1 are the desired 3 vertices removed from $V(C(G_R))$.

Now assume that $N(\{t_1,t_2\}) \setminus MB \neq \emptyset$. There is nothing to prove if $|N(\{t_1,t_2\}) \setminus MB| \geqslant 2$ or $SN(t_1,t_2) \setminus MB \neq \emptyset$. We assume the opposite and derive a contradiction. Since $|N(\{t_1,t_2\}) \setminus MB| = 1$, t_1 is incident to at least one vertex z_4 of $L(G,u,3) \setminus \{z_1,z_2,t_2\}$. This vertex z_4 has at least one neighbor y_4 outside of MB which is different from t_1 and from the only vertex w of $N(\{t_1,t_2\}) \setminus MB$: otherwise, since w is not adjacent to any vertex of $SN(t_1,t_2) \setminus MB$, $L(G,u,3) \setminus \{z_1,z_2,t_2,z_4\}$ constitute a good cut. Vertex y_4 belongs to $SN(t_1,t_2) \setminus MB$ in contradiction to our assumptions.

Finally, we assume that both t_1 and t_2 belong to $L(G,u,3)$. In order to avoid $L(G,u,3) \setminus \{z_1,z_2,t_1,t_2\}$ to be a good cut, we get that both $N(\{t_1,t_2\}) \setminus MB$ and $SN(t_1,t_2) \setminus MB$ are nonempty which contributes at least 2 vertices removed from $V(C(G_R))$ in addition to MB. To complete the proof, we have to show that there one more vertex outside of MB, which is removed from $C(G_R)$. To this end, denote the vertices of $L(G,u,3) \setminus \{z_1,z_2,t_1,t_2\}$ by z_5, z_6, z_7, z_8. Observe that there is at least one edge between $\{w_1,w_2\}$ and $\{z_5,\dots,z_8\}$ (recall that vertices w_1, w_2 are the shown in Fig. 7). Indeed, if not then $|L(G, v_1, 3) \cap L(G, u, 4)| = 0$. Taking into account that $|L(G, u, 4)| \geqslant 5$, there is at least one neighbor v of u such that $|L(G, v, 3) \cap L(G, u, 4)| \geqslant 3$, i.e. $|L(G, u, 3) + L(G, v, 3) \cap L(G, u, 4)| \geqslant 11$ which is an earlier case considered in Section 10.1. Now, assume w.l.o.g. that w_1 is adjacent to z_5. As before, we see that z_5 is adjacent to some vertex y_5 which does not belong to $N(\{t_1, t_2\}) \setminus MB$ as well as to $SN(t_1, t_2) \setminus MB$: otherwise z_6, z_7, z_8 with the only vertex of $SN(t_1, t_2) \setminus MB$ form a good cut. Since this vertex y_5 belongs to $N^+(N^+(\{w_1, w_2\}))$, it is removed from $V(C(G_R))$ by Lemma 8 and this is the desired third vertex in the considered case. □

11. Correctness proof, complexity analysis, and a new upper bound for the parameterized VC-3 problem

Theorem 2. *FindIndep(G) returns a MIS of G.*

Proof. By induction on $|V(G)|$. The statement is trivial for $|V(G)| = 0$. For each branching decision applied by *FindIndep*(G) if $|V(G)| > 0$, we have proven correctness of these decisions given the correctness of recursive calls of *FindIndep*(G) applied to the residual graphs (Lemma 1, Corollary 1, Lemmas 2, 3, 11, 13, the discussion in Section 7.1, Lemmas 18, 21, 23). Since the residual graphs have smaller number of vertices than G, the correctness of *FindIndep*(G) applied to them follows from the induction assumption. □

In the previous sections we proved a number of lemmas regarding the sizes of FNSes of G. These lemmas are summarized in the following lemma.

Lemma 27. *A nontrivial graph G can have at most 3 FNSes. Assume that G has two FNSes G_L and G_R. Then at least one of the following statements happens.*

(1) $|V(C(G_L))| \leqslant |V(C(G))| - 10$, $|V(C(G_R))| \leqslant |V(C(G))| - 8$.
(2) $|V(C(G_L))| \leqslant |V(C(G))| - 11$, $|V(C(G_R))| \leqslant |V(C(G))| - 7$.
(3) $|V(C(G_L))| \leqslant |V(C(G))| - 12$, $|V(C(G_R))| \leqslant |V(C(G))| - 6$.

If G has 3 FNSes G_L, G_M, and G_R then at least one of the following statements happens.

(1) $|V(C(G_L))| \leqslant |V(C(G))| - 10$, $|V(C(G_M))| \leqslant |V(C(G))| - 17$, $|V(C(G_R))| \leqslant |V(C(G))| - 15$.
(2) $|V(C(G_L))| \leqslant |V(C(G))| - 10$, $|V(C(G_M))| \leqslant |V(C(G))| - 10$, *and* $|V(C(G_R))| \leqslant |V(C(G))| - 21$.

Moreover, in the last case $|V(C(G_L))| = 0$ or $|V(C(G_L))| \leqslant |V(C(G))| - 11$ or G_L has at most two FNSes.

Proof. The upper bounds on the sizes of FNSes provided in the present lemma are obtained in Lemmas 9, 12, 14–20, 22, 24–26. These lemmas compute the upper bounds for all possible branching decisions made by *FindIndep*(G). Hence at least one statement in the above two lists is true.

Let us show that the additional requirements hold if the very last statement is satisfied. Assume that $|V(C(G_L))| > 0$. Recall that G_L is obtained by selection of vertex u as shown in Fig. 6. The immediate effect of selection of u to the residual graph is removal of u and the neighbors of u. The degrees of vertices of $L(G, u, 2)$ decrease to 2, the degrees of the rest of the vertices of G remain the same as they were *before* the selection of u, i.e. 3 (recall that in the considered case, graph G is cubic). Also, the vertex u is explicitly selected so that $G[L(G, u, 2)]$ has an isolated vertex v, as stated in Sections 10.1, 10.2, and 10.3.[4] In other words, as a result of selection of u, v is incident to two vertices y_1 and y_2 both of degree 3.

If G_L is obtained as a result of transformation of the residual graph described in Sections 2 and 3 and this transformation removes or decreases the degree of at least one more vertex of degree 3, we have $|V(C(G_L))| \leqslant |V(C(G))| - 11$. Otherwise, if a good cut of degree 3 separating or a rectangle or a triangle is detected in $C(G_L)$ then G_L has two FNSes. If none of these cases happens then observe that y_1 and y_2 are connected in $C(G_L)$ by an odd edge replacing vertex v. Hence, the case considered in Section 8 is satisfied. Consequently, $C(G_L)$ again has two FNSes. □

Let us call a recursive application of *FindIndep* to a graph G' *atomic* if during the processing, *FindIndep*(G') does not apply itself recursively.

Theorem 3. *Let G be a nontrivial graph with $|V(C(G))| = n$. Then the number of atomic recursive calls made during the processing of FindIndep(G) is at most c^n, where $c = 1.0892$.*

Proof. By induction on n. The theorem is clear for $n = 0$ or if *FindIndep*(G) does not apply itself recursively. Hence assume that $n > 0$ and the theorem holds for any nontrivial G' with $|V(C(G'))| < n$.

[4] This is the place we use the first condition of Section 10.1 and the fact the vertex selection in Section 10.3 is good.

Assume that G has only one FNS G'. Graph G is transformed into G' by a sequence of consecutive "one-branch" recursive calls finishing by the call *FindIndep*(G'). Clearly, each of these recursive calls is not an atomic one. Consequently, the number of atomic calls applied during the processing of *FindIndep*(G) equals the number of recursive calls applied during the processing of *FindIndep*(G'). Taking into account that $|V(C(G'))| < |V(C(G))|$ (transformation from a nontrivial graph into its FNS always involves selection or removal of a vertex of degree 3), the statement of the theorem follows by the induction assumption.

Assume now that G has two FNSes G_L and G_R. The transformation of G into each one of them involves an operation of selection or removal of a vertex and a sequence of simplifying recursive call finishing by the call to the respective FNS. Clearly, none of these calls is an atomic one. It follows that the number of atomic recursive calls applied during the execution of *FindIndep*(G) equals the sum of such recursive calls regarding *FindIndep*(G_L) and *FinbdIndep*(G_R). By the induction assumption, at most $c^{n-k_L} + c^{n-k_R}$ atomic recursive calls are applied during the processing of *FindIndep*(G) where $|V(C(G_L))| = n - k_L$, $|V(C(G_R))| = n - k_R$. To prove the theorem for the considered case, one must show that $c^{n-k_L} + c^{n-k_R} \leqslant c^n$ or, simplifying the inequality, that $c^{-k_L} + c^{-k_R} \leqslant 1$.

By Lemma 27, 3 different cases are possible regarding k_L and k_R: [$k_L \geqslant 10$ and $k_R \geqslant 8$] or [$k_L \geqslant 11$ and $k_R \geqslant 7$] or [$k_L \geqslant 12$ and $k_R \geqslant 6$]. A simple computation shows that for each of these cases $c^{-k_L} + c^{-k_R} \leqslant 1$. Thus the theorem holds for the case where G has two FNSes.

Assume that G has 3 FNSes G_L, G_M, G_R. By Lemma 27, there are two possibilities of bounds on the sizes of FNSes. If the first happens then, analogously to the case with two FNSes, the statement follows from the easily verified inequality $c^{-10} + c^{-17} + c^{-15} \leqslant 1$. If G_L, G_M, and G_R are bounded by the set of inequalities $|V(C(G_L))| \leqslant |V(C(G))| - 10$, $|V(C(G_M))| \leqslant |V(C(G))| - 10$, and $|V(C(G_R))| \leqslant |V(C(G))| - 21$ then the reasoning involves additional requirement regarding G_L specified by Lemma 27.

In particular, if $|V(C(G_L))| \leqslant |V(C(G))| - 11$ then, analogously to the previous cases, the theorem follows from the easily verified inequality $c^{-11} + c^{-10} + c^{-21} \leqslant 1$. In the case G_L is the empty graph, to prove the desired statement one has to show that the inequality $1 + c^{n-10} + c^{n-21} \leqslant c^n$ holds. Dividing all the items by c^n, we obtain the inequality $c^{-n} + c^{-10} + c^{-21} \leqslant 1$. Now observe that $n \geqslant 50$ because otherwise *FindIndep*(G) does not apply itself recursively. The last inequality immediately follows from this observation.

If none of the above two cases happens regarding G_L then G_L has 2 FNSes. By Lemma 27 and taking into account that $|V(C(G_L))| \leqslant |V(C(G))| - 10$, [$|V(C((G_L)_L))| \leqslant |V(C(G))| - 20$ and $|V(C((G_L)_R))| \leqslant |V(C(G))| - 18$] or [$|V(C((G_L)_L))| \leqslant |V(C(G))| - 21$ and $|V(C((G_L)_R))| \leqslant |V(C(G))| - 17$] or [$|V(C((G_L)_L))| \leqslant |V(C(G))| - 22$ and $|V(C((G_L)_R))| \leqslant |V(C(G))| - 16$]. Arguing as for the previous cases, one can see that the number of atomic recursive calls made by *FindIndep*(G) equals the sum of the amounts of atomic recursive calls made by *FindIndep*$((G_L)_L)$, *FindIndep*$((G_L)_R)$, *FindIndep*(G_M), *FindIndep*(G_R). Applying the induction assumption and elimination of c^n, one gets that the statement of the theorem immediately follows from the easily verified inequalities $c^{-20} + c^{-18} + c^{-10} + c^{-21} \leqslant 1$, $c^{-21} + c^{-17} + c^{-10} + c^{-21} \leqslant 1$, and $c^{-22} + c^{-16} + c^{-10} + c^{-21} \leqslant 1$.

Thus we have proven the statement of the theorem for the cases where G has no FNSes (where *FindIndep*(G) does not apply itself recursively) as well for the cases where has 1, 2, or 3 FNSes. By Lemma 27, we have covered all the possible cases, which concludes the proof of the theorem. □

Now, the complexity of *FindIndep*(G) can be derived as a corollary from Theorem 3.

Corollary 3. *The runtime of FindIndep*(G) *is bounded by* $O(1.0892^n)$, *where* $n = |V(G)|$.

Proof. We may assume that G is a nontrivial graph because otherwise, it is transformed into a nontrivial graph within a polynomial time. We will show that the time complexity of *FindIndep*(G) polynomially relates to the number of atomic recursive calls made during the execution of *FindIndep*(G), which is at most $1.0892^{|V(C(G))|}$ by Theorem 3. Taking into account that $|V(C(G))| \leqslant n$ and that the constant 1.0892 is obtained by rounding the base of the exponent that eliminates all the polynomial factors, the desired statement will immediately follow.

The operations performed by *FindIndep*(G) can be classified as *decision* operations that select vertices to the MIS being constructed or remove them and auxiliary operations (checking properties of the given graph, updating the residual graph resulting from the last decision operation, etc). It is clear from the description of *FindIndep* that the number of auxiliary operations is polynomially related to the number of the decision operations (auxiliary operations are applied either prior to a decision operation in order to select an appropriate one or as a result of the decision operation; each decision operation is accompanied by a polynomial number of auxiliary operations). Consequently, it is sufficient to show that the number of atomic recursive calls made by *FindIndep*(G) polynomially relates to the number of decision operations.

There is a one-to-one correspondence between the decision operations made by *FindIndep*(G) and the *general* number of recursive calls (not only atomic ones) made during the execution of *FindIndep*(G): each decision operation results in a recursive call as well as each recursive call is a result of some decision operation. For each atomic recursive call, there is a *sequence of consecutive recursive calls* that eventually causes the atomic call. Conversely, each recursive call participates in such a sequence. Each such a sequence has length $O(n)$. Indeed, let *FindIndep*(G_1) be a recursive call in this sequence and let *FindIndep*(G_2) be its successor. Then $|V(G_2)| < |V(G_1)|$. Consequently, each sequence of recursive calls starting from *FindIndep*(G) and leading to an atomic call consists of at most n elements.

Thus the general number of recursive calls made by *FindIndep*(G) is $O(n)$ multiplied by the number of atomic calls. Considering the one-to-one correspondence of the general number of recursive calls to the number of decision operations, the corollary follows. □

Using Corollary 3, we easily obtain a new upper bound for the parameterized Vertex Cover problem for graphs with maximum degree 3 (VC-3). Recall that given a graph G with maximum degree 3 and a constant k, the problem asks whether there is a vertex cover of size at most k. A nice property regarding this problem [4] states that there is an $O(n)$ transformation of G into a maximum degree 3 graph G', $|V(G')| \leqslant 2k$ such that G' admits a vertex cover of size at most k if and only if G does. In terms of parameterized complexity theory, graph G' is called a *kernel* of G.

The parameterized VC-3 problem can be solved regarding G' by checking whether the *minimum* vertex cover of G' is larger than k or not. The minimum vertex cover is a complement of a MIS of G', which can be computed by *FindIndep*(G') in time $O(1.0892^{2k}) < O(1.1864^k)$ (Corollary 3). Taking into account that graph G' can be obtained from G in $O(n)$, the VC-3 problem can be solved for graph G in time $O(1.1864^k + n)$, which improves over the bound $O(k^2 * 1.194^k + n)$ [4], which is currently the smallest one to the best of our knowledge.

Besides improving the upper bound on the VC-3 problem, the above result has a methodological interest. To the best of our knowledge, this is the first time where a good parameterized algorithm is obtained by design and analysis of an exact exponential algorithm. Thus the result connects the areas of Exact Complexity and Parameterized Complexity and opens a new application area of design and analysis of exact algorithms.

References

[1] R. Beigel, Finding maximum independent sets in sparse and general graphs, in: SODA, 1999, pp. 856–857.
[2] N. Bourgeois, B. Escoffier, V. Paschos, An o*(1.0977n) exact algorithm for max independent setin sparse graphs, in: IWPEC, 2008, pp. 55–65.
[3] J. Chen, I. Kanj, W. Jia, Vertex cover: Further observations and further improvements, Journal of Algorithms 41 (2) (2001) 280–301.
[4] J. Chen, I. Kanj, G. Xia, Labeled search trees and amortized analysis: Improved upper bounds for NP-Hard problems, Algorithmica 43 (4) (2005) 245–273.
[5] J. Chen, L. Liu, W. Jia, Improvement on vertex cover for low-degree graphs, Networks 35 (4) (2000) 253–259.
[6] R. Diestel, Graph Theory, second ed., Springer-Verlag, Heidelberg, 1997.
[7] F. Fomin, K. Høie, Pathwidth of cubic graphs and exact algorithms, Information Processing Letters 97 (5) (2006) 191–196.
[8] M. Fürer, A faster algorithm for finding maximum independent sets in sparse graphs, in: LATIN, 2006, pp. 491–501.
[9] A. Kojevnikov, A. Kulikov, A new approach to proving upper bounds for max-2-sat, in: SODA, 2006, pp. 11–17.
[10] I. Razgon, A faster solving of the Maximum Independent Set problem for graphs with maximal degree 3, in: ACiD 2006, 2006, pp. 131–142.
[11] G. Woeginger, Exact algorithms for NP-hard problems: A survey, in: Combinatorial Optimization, 2001, pp. 185–208.