FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Hitting paths in graphs
Student:	Bc. Radovan Červený
Supervisor:	RNDr. Ondřej Suchý, Ph.D.
Study Programme:	Informatics
Study Branch:	System Programming
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2018/19

Instructions

Get familiar with the known parameterized algorithms for P_3 vertex cover and P_4 vertex cover.

Generalize these algorithms to the P_5 vertex cover problem or identify fundamental obstacles preventing such a generalization.
The running time should beat the running time of known algorithms for Hitting Set with sets of the corresponding size.

After consulting with the supervisor select one of the above mentioned algorithms and implement it in a suitable language.
Furthermore, implement a naive algorithm for the problem.

Test the resulting program on a suitable data, evaluate its performance, and compare it to the naive algorithm.

References

Mingyu Xiao, Shaowei Kou: Kernelization and Parameterized Algorithms for 3-Path Vertex Cover. TAMC 2017: 654-668

Jianhua Tu, Zemin Jin: An FPT algorithm for the vertex cover P4 problem. Discrete Applied Mathematics 200: 186-190 (2016)

Jianhua Tu: A fixed-parameter algorithm for the vertex cover P3 problem. Inf. Process. Lett. 115(2): 96-99 (2015)

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 6, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Hitting paths in graphs

Bc. Radovan Červený

Department of theoretical computer science
Supervisor: RNDr. Ondřej Suchý, Ph.D.

May 2, 2018

Acknowledgements

First, my utmost gratitude belongs to my supervisor Ondřej Suchý, whose invaluable insights and support made this thesis possible. Second, I thank my family which allowed me to study at the university and pursue my passion. Third, I am grateful to my friends, colleagues and coworkers for support during these dire times. And last but not least, I want to thank my beloved Martina for everything she does for me.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 2, 2018 ....

Citation of this thesis

Červený, Radovan. *Hitting paths in graphs.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstract

The problem of d-Path Vertex Cover, d-PVC lies in determining a subset F of vertices of a given graph $G = (V, E)$ such that $G \setminus F$ does not contain a path on d vertices. The paths we aim to cover need not to be induced. It is known that the d-PVC problem is NP-complete for any $d \geq 2$. 5-PVC is known to be solvable in $\mathcal{O}(5^k n^{\mathcal{O}(1)})$ time when parameterized by the size of the solution k. In this thesis we present an iterative compression algorithm that solves the 5-PVC problem in $\mathcal{O}(4^k n^{\mathcal{O}(1)})$ time.

Keywords graph algorithms, Hitting Set, iterative compression, parameterized complexity, d-Path Vertex Cover

Abstrakt

Problém zvaný d-Path Vertex Cover, d-PVC spočívá v nalezení podmnožiny F vrcholů daného grafu $G = (V, E)$ takové, že $G \setminus F$ neobsahuje žádnou cestu na d vrcholech. Cesty, které chceme takto podchytit, nemusí být indukované. Je známo, že problém d-PVC je NP-úplný pro každé $d \geq 2$. Dále je známo, že problém 5-PVC parametrizovaný velikostí řešení k je řesitelný v čase $\mathcal{O}(5^k n^{\mathcal{O}(1)})$. V této práci přicházíme s algoritmem používající iterativní kompresi, který řeší problém 5-PVC v čase $\mathcal{O}(4^k n^{\mathcal{O}(1)})$.

Klíčová slova grafové algoritmy, Hitting Set, iterativní komprese, parametrizovaná složitost, d-Path Vertex Cover

Contents

List of Figures

List of Tables

Introduction

The problem of d-Path Vertex Cover, d-PVC lies in determining a subset F of vertices of a given graph $G = (V, E)$ such that $G \setminus F$ does not contain a path on d vertices. The paths we aim to cover need not to be induced. The problem was first introduced by Brešar et al. [1], but its NP-completeness was already proven by the meta-theorem of Lewis and Yannakakis [9] for any $d \geq 2$. The 2-PVC problem corresponds to the well known Vertex Cover problem and the 3-PVC problem is also known as Maximum Dissociation Set. The d-PVC problem is motivated by the field of designing secure wireless communication protocols [10] or in route planning and speeding up shortest path queries [7].

Several efficient (faster than trivial enumeration) exact algorithms are known for 2-PVC which can be solved in $\mathcal{O}^*(1.1996^n)$ time and polynomial space due to Xiao and Nagamochi [17], and for 3-PVC which can be solved in $\mathcal{O}^*(1.4656^n)$ time and polynomial space due to Xiao and Kou [15].

When parameterized by the size of the solution k, the d-PVC problem has a trivial FPT algorithm that runs in $\mathcal{O}^*(d^k)$ time (FPT and the $\mathcal{O}^*$ notation are properly introduced in Chapter 1). In order to find more efficient solutions, the problem has been extensively studied in a setting where d is a small constant. For the 2-PVC problem, the algorithm of Chen, Kanj, and Xia [2] has currently best known running time $\mathcal{O}^*(1.2738^k)$. For the 3-PVC problem, Tu [13] used iterative compression to achieve a running time $\mathcal{O}^*(2^k)$, which was later improved by Katrenič [8] to $\mathcal{O}^*(1.8127^k)$ and further by Xiao and Kou [16] to $\mathcal{O}^*(1.7485^k)$ by using a branch-and-reduce approach. For the 4-PVC problem, Tu and Jin [14] again used iterative compression and achieved a running time $\mathcal{O}^*(3^k)$. For $d \geq 5$ no non-trivial algorithms are known.

Our contribution. We present an algorithm that solves the 5-PVC problem parameterized by the size of the solution k in $\mathcal{O}^*(4^k)$ time by employing the iterative compression technique.

Organization of this thesis. We introduce the notation and properly define the 5-PVC problem in Chapter 1. The technique of iterative compression is then described in Chapter 2. Our main algorithm (the disjoint compression routine) together with its proof of correctness is exposed in Chapter 3. In Chapter 4 we experimentally evaluate our algorithm against the trivial one. We conclude this thesis with a few open questions.

Chapter 1

Preliminaries

We use the $\mathcal{O}^*$ *notation* as described by Fomin and Kratsch [6]. The notation is derived from the classical big-O notation. Big-O notation is defined as follows. For function $f(n)$ and $g(n)$ we write $f(n) = \mathcal{O}(g(n))$ if there are positive numbers n_0 and c such that for every $n > n_0$ we have $f(n) < c \cdot g(n)$. The $\mathcal{O}^*$ notation is a modification of big-O notation which suppresses all polynomially bounded factors. Formally, for functions $f(n)$ and $g(n)$ we write $f(n) = \mathcal{O}^*(g(n))$ if $f(n) = \mathcal{O}(g(n)\mathit{poly}(n))$, where $\mathit{poly}(n)$ is a polynomial.

We use the notation of parameterized complexity as described by Cygan et al. [3]. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed finite alphabet. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the *parameter*. A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* (FPT) if there exists an algorithm A, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm A correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$, where $|(x, k)|$ is the size of the problem instance (x, k). The algorithm A is then called an FPT algorithm.

We use standard graph notation and consider simple and undirected graphs unless otherwise stated. Vertices of graph G are denoted by $V(G)$, edges by $E(G)$. By $G[X]$ we denote the subgraph of G induced by vertices of $X \subseteq V(G)$. By $N(v)$ we denote the set of neighbors of $v \in V(G)$ in G. Analogically, $N(X) = \bigcup_{x \in X} N(x)$ denotes the set of neighbors of vertices in $X \subseteq V(G)$. The degree of vertex v is denoted by $\mathit{deg}(v) = |N(v)|$. For simplicity, we write $G \setminus v$ for $v \in V(G)$ and $G \setminus X$ for $X \subseteq V(G)$ as shorthands for $G[V(G) \setminus \{v\}]$ and $G[V(G) \setminus X]$, respectively.

Definition 1. A *k-path* denoted as an ordered k-tuple $P_k = (p_1, p_2, \ldots, p_k)$ is a graph with vertices $V(P_k) = \{p_1, p_2, \ldots, p_k\}$ and edges $E(P_k) = \{\{p_i, p_{i+1}\} \mid i \in \{1, 2, \ldots, k-1\}\}$. A path P_k *starts* at vertex x when $p_1 = x$.

A k-cycle is a cycle on k vertices. A triangle is a 3-cycle. A P_5-free graph is a graph that does not contain a P_5 as a subgraph (the P_5 needs not to be induced).

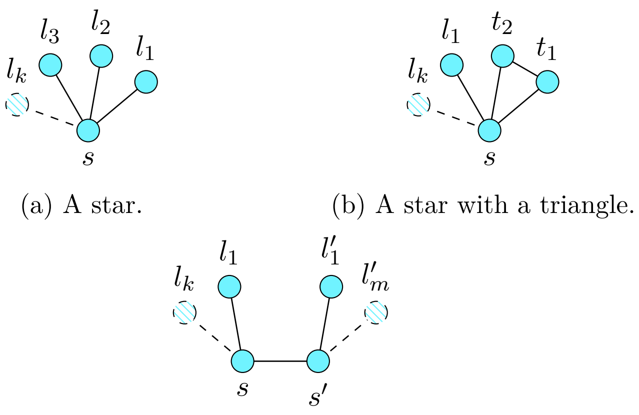
The 5-Path Vertex Cover problem is formally defined as follows:

5-Path Vertex Cover, 5-PVC	
Input:	A graph $G = (V, E)$, an integer $k \in Z_0^+$.
Output:	A set $F \subseteq V$, such that $\|F\| \leq k$ and $G \setminus F$ is a P_5-free graph.

Definition 2. A *star* is a graph S with vertices $V(S) = \{s\} \cup \{l_1, \dots, l_k\}$, $k \geq 3$ and edges $E(S) = \{\{s, l_i\} \mid i \in \{1, \dots, k\}\}$ (see Figure 1.1a). Vertex s is called a center, vertices $L = \{l_1, \dots, l_k\}$ are called leaves.

Definition 3. A *star with a triangle* is a graph $S^\triangle$ with vertices $V(S^\triangle) = \{s, t_1, t_2\} \cup \{l_1, \dots, l_k\}$, $k \geq 1$ and edges $E(S^\triangle) = \{\{s, t_1\}, \{s, t_2\}, \{t_1, t_2\}\} \cup \{\{s, l_i\} \mid i \in \{1, \dots, k\}\}$ (see Figure 1.1b). Vertex s is called a center, vertices $T = \{t_1, t_2\}$ are called triangle vertices and vertices $L = \{l_1, \dots, l_k\}$ are called leaves.

Definition 4. A *di-star* is a graph D with vertices $V(D) = \{s, s'\} \cup \{l_1, \dots, l_k\} \cup \{l'_1, \dots, l'_m\}$, $k \geq 1, m \geq 1$ and edges $E(D) = \{\{s, s'\}\} \cup \{\{s, l_i\} \mid i \in \{1, \dots, k\}\} \cup \{\{s', l'_j\} \mid j \in \{1, \dots, m\}\}$ (see Figure 1.1c). Vertices s, s' are called centers, vertices $L = \{l_1, \dots, l_k\}$ and $L' = \{l'_1, \dots, l'_m\}$ are called leaves.



(a) A star.

(b) A star with a triangle.

(c) A di-star.

Figure 1.1

Lemma 1. *If a connected graph is P_5-free and has more than 5 vertices, then it is a star, a star with a triangle, or a di-star.*

Proof. Suppose we have a P_5-free graph G on at least 5 vertices. Firstly, G does not contain a k-cycle, $k \geq 5$ as a subgraph, since P_5 is a subgraph of such a k-cycle. Secondly, G does not contain a 4-cycle as a subgraph, since G has at least 5 vertices and it is connected which implies that there is at least one vertex connected to the 4-cycle which in turn implies a P_5 in G. Finally, G does not contain two edge-disjoint triangles as a subgraph, since G is connected, the two triangles are either sharing a vertex or are connected by some path, which in both cases implies a P_5 in G. Consequently, G contains either exactly one triangle or is acyclic.

Consider the first case where G contains exactly one triangle. Label the vertices of the triangle with $\{t_1, t_2, t_3\}$. Then we claim that all vertices outside the triangle are connected by an edge to exactly one vertex of that triangle, let that vertex be t_1. Indeed, for contradiction suppose they are not. Since we have at least 5 vertices in G, label the two existing vertices outside the triangle x and y. Then we either have x and y connecting to two different vertices of the triangle, let them be t_1, t_2, which immediately implies a $P_5 = (x, t_1, t_3, t_2, y)$ in G, or we have a $P_3 = (x, y, t_1)$ connected to the triangle, which again implies a $P_5 = (x, y, t_1, t_2, t_3)$. Hence, if G contains a triangle, then it is a star with a triangle.

Consider the second case where G is acyclic. Then we claim that there is a dominating edge in G, i.e. an edge $e = \{x, y\}$ such that $V(G) = N(\{x, y\}) \cup \{x, y\}$. Indeed, for contradiction suppose that there is no such edge. Then we have that for each edge $e = \{x, y\}$ in G there must be a vertex v that is adjacent neither to x, nor to y. Assume that v is connected to y through some vertex u. The same also holds for the edge $\{y, u\}$, so assume that there is a vertex $v' \neq x$ that is connected to u through some vertex $u' \neq y$. But then we have a $P_5 = (x, y, u, u', v')$ in G.

Label the dominating edge $e = \{s, s'\}$. Here, if only one of the vertices s, s' has degree greater than one, we have a star, otherwise we have a di-star. $\square$

Chapter **2**

Iterative compression

Iterative compression is a technique which enables us to design FPT algorithms. It was first introduced by Reed et al. [12] to solve the Odd Cycle Traversal problem. The main idea of iterative compression lies in the *compression routine*, which takes a solution F and returns a solution F' such that $|F| < |F'|$ or proves that the solution F is already optimal in size.

2.1 Algorithm

We start with an empty vertex set $V' = \emptyset$ and empty solution $F = \emptyset$ and work with the graph $G[V']$. Surely, an empty set F is a solution for a currently empty graph $G[V']$. One by one we add vertices $v \in V \setminus V'$ to V' and F until $V' = V$ and if at any time the solution becomes too big, i.e. if $|F| = k + 1$, we start the compression routine.

The compression routine takes F and goes through every partition of F into two sets X, Y such that $Y \neq \emptyset$. Here, X is the part of F that we want to keep in the solution and Y is the part of F that we want to replace with vertices from $V' \setminus F$. Since X are vertices we already decided to keep in the solution, we remove them from $G[V']$, i.e. we continue with $G' = G[V'] \setminus X$. Now the problem is to find a solution F' for G' such that $|F'| \leq |Y| - 1$ and F' is disjoint from Y. We consider this partition only if $G[Y]$ is P_5-free. Indeed, we require that F' is disjoint from Y so we cannot have any P_5 paths in $G[Y]$. To find this smaller disjoint solution F' for G' we use the *disjoint compression routine*. The smaller solution for $G[V']$ is then constructed as $\hat{F} = X \cup F'$ and it follows from construction of $\hat{F}$ that $|\hat{F}| \leq k$.

If after going through all partitions of F we did not find a smaller solution for $G[V']$, then we know that F was optimal in size and signalize that there is no solution (see Algorithm 1 for illustration).

The disjoint compression routine is typically the only part that must be designed specifically for the problem. In our case the disjoint compression routine is called DISJOINT and the problem it solves is called 5-PVC WITH P_5-FREE BIPARTITION. We describe the routine and the problem in Chapter 3.

Algorithm 1 Pseudocode of the iterative compression algorithm

procedure ALGO$(G = (V, E), k)$
  $V' \leftarrow \emptyset$, $F \leftarrow \emptyset$
  while $V \setminus V' \neq \emptyset$ **do**
    with $v \in V \setminus V'$
    $V' \leftarrow V' \cup \{v\}$, $F \leftarrow F \cup \{v\}$
    if $|F| = k + 1$ **then**
      $\hat{F} \leftarrow$ *no solution*
      for each $X \subsetneq F$ **do**
        $Y \leftarrow F \setminus X$
        if $G[Y]$ is P_5-free **then**
          $G' \leftarrow G[V'] \setminus X$
          $F' \leftarrow$ DISJOINT$(G', Y, V(G') \setminus Y, |Y| - 1)$
          if $F' \neq$ *no solution* **then**
            $\hat{F} = X \cup F'$
            break
          end if
        end if
      end for
      if $\hat{F} \neq$ *no solution* **then**
        $F \leftarrow \hat{F}$
      else
        return *no solution*
      end if
    end if
  end while
  return F
end procedure

Chapter **3**

5-PVC with P_5-free bipartition

3.1 Problem definition

Definition 5. A P_5-free bipartition of graph $G = (V, E)$ is a pair (V_1, V_2) such that $V = V_1 \cup V_2, V_1 \cap V_2 = \emptyset$ and $G[V_1], G[V_2]$ are P_5-free.

The problem that is solved by our disjoint compression routine DISJOINT is formally defined as follows:

5-PVC with P_5-free Bipartition, 5-PVCwB	
Input:	A graph $G = (V, E)$ with P_5-free bipartition (V_1, V_2), an integer $k \in Z_0^+$.
Output:	A set $F \subseteq V_2$, such that $\lvert F\rvert \leq k$ and $G \setminus F$ is a P_5-free graph.

Throughout this thesis the vertices from V_1 will be also referred to as "red" vertices and vertices from V_2 will be also refereed to as "blue" vertices. The same colors will also be used in figures with the same meaning.

3.2 Algorithm

Our algorithm is a recursive procedure DISJOINT_R(G, V_1, V_2, F, k), where G is the input graph, V_1, V_2 are the partitions of the P_5-free bipartition of G, F is the solution being constructed, and k is the maximum number of vertices we can still add to F. The procedure repeatedly tries to apply a series of rules with a condition that a rule (RI) can be applied only if all Rules that come before (RI) cannot be applied (see Algorithm 2 for illustration). It is paramount that in every call of DISJOINT_R at least one rule can be applied. The main work is done in Rules of two types: *reduction rules* and *branching rules*. To make it easier for the reader we also use rules called *context rules*, which only describe the configuration we are currently in and serve as some sort of a parent rules for their subrules.

Definition 6. A *reduction rule* is used to simplify a problem instance, i.e. remove some vertices or edges from G and possibly add some vertices to a solution, or to halt the algorithm.

Definition 7. A *branching rule* splits the problem instance into at least two subinstances. The branching is based on subsets of vertices that we try to add to a solution and by adding them to the solution we also remove them from G.

The notation we use to denote the individual branches of a branching rule is as follows: $\langle X_1 \mid X_2 \mid \dots \mid X_l \rangle$. Such a rule has l branches and $X_1, X_2, \dots, X_l$ are subsets of V_2 which we try to add to the solution. This rule is translated into the following l calls of the procedure:

$$\text{DISJOINT\_R}(G \setminus X_i, V_1, V_2 \setminus X_i, F \cup X_i, k - |X_i|) \text{ for } i \in \{1, \dots, l\}$$

Definition 8. A rule is *applicable* if the conditions of the rule are satisfied and if there is no other rule that comes before that is applicable.

If a context rule is not applicable, it means that none of its subrules is applicable.

Definition 9. A reduction rule is *correct* if it satisfies that the problem instance has a solution if and only if the simplified problem instance has a solution.

A branching rule is *correct* if it satisfies that if the problem instance has a solution, then at least one of the branches of the rule will return a solution.

Definition 10. When we say we *delete* a vertex, we mean that we remove it from G and also add it to the solution F. When we say we *remove* a vertex, we mean that we remove it from G and do not add it to the solution F.

The fact that among Rules (R0) – (R18) there is always at least one that is applicable is proven in Theorem 19, Section 3.12.

In the following sections assume that the parameters of the current call of DISJOINT_R are G, V_1, V_2, F, k.

3.3 Preprocessing

Reduction rule (R0). This rule stops the recursion of DISJOINT_R. It has three stopping conditions:

1. If $k < 0$, return *no solution*;
2. else if G is P_5-free, return F;
3. else if $k = 0$, return *no solution*.

Algorithm 2 Illustrative pseudocode of the recursive procedure

```
procedure DISJOINT(G, V_1, V_2, k)
    return DISJOINT_R(G, V_1, V_2, ∅, k)
end procedure

procedure DISJOINT_R(G, V_1, V_2, F, k)
    F_result ← no solution
    R ← the first rule that is applicable
    if R is (R0) then
        F_result ← either F or no solution based on which stopping condition
                   of (R0) was triggered
    else if R is a reduction rule then
        let G', V_1', V_2' be simplified by R and let X be the vertices that R
        wants to add to F
        F_result ← DISJOINT_R(G', V_1', V_2', F ∪ X, k − |X|)
    else
        let the branches of R be ⟨ X_1 | X_2 | ... | X_l ⟩
        for i ← 1, ..., l do
            F_candidate ← DISJOINT_R(G \ X_i, V_1, V_2 \ X_i, F ∪ X_i, k − |X_i|)
            if F_candidate ≠ no solution and
              (F_result = no solution or |F_candidate| ≤ |F_result|) then
                F_result ← F_candidate
            end if
        end for
    end if
    return F_result
end procedure
```

Reduction rule (R1). Let $v \in V(G)$ be a vertex such that there is no P_5 in G that uses v. Then remove v from G.

Proof of correctness. Let $v \in V(G)$ be a vertex that is not used by any P_5 in G and let F be a solution to the 5-PVCwB instance $(G \setminus v, V_1 \setminus \{v\}, V_2 \setminus \{v\}, k)$. Then F is also a solution to (G, V_1, V_2, k) since v is not used by any P_5 in G.

If $(G \setminus v, V_1 \setminus \{v\}, V_2 \setminus \{v\}, k)$ does not have a solution, then we claim that (G, V_1, V_2, k) also does not have a solution. Indeed, adding vertices can only create new P_5 paths. ⋈

Branching rule (R2). Let P be a P_5 in G with $X = V(P) \cap V_2$ such that $|X| \leq 3$. Then branch on $\langle x_1 \mid x_2 \mid \ldots \rangle, x_i \in X$, i.e. branch on the blue vertices of P.

Proof of correctness. We have to delete at least one blue vertex in P, thus branching on the blue vertices of P is correct. ⋈

Lemma 2. *Assume that Rules (R0) – (R2) are not applicable. Then for each vertex $v \in V(G)$ there exists a P_5 in G that uses v; every P_5 in G uses exactly one red vertex; and there are only isolated vertices in $G[V_1]$.*

Proof. If Rule (R1) is not applicable, then for each vertex $v \in V(G)$ there exists a P_5 in G that uses v. If Rule (R2) is not applicable, then every P_5 in G uses at most one red vertex and since (V_1, V_2) is a P_5-free bipartition we cannot have a P_5 in G that uses no red vertex.

To prove that there are only isolated vertices in $G[V_1]$, assume for contradiction that there is an edge e in $G[V_1]$. Since each P_5 in G uses exactly one red vertex there cannot be a P_5 that uses e. Which means that at least one of the vertices of e is not used by any P_5 in G and we get a contradiction with Rule (R1) not being applicable. □

3.4 Dealing with isolated vertices in $G[V_2]$

Lemma 3. *Assume that Rules (R0) – (R2) are not applicable. Let v be an isolated vertex in $G[V_2]$ and let F be a solution to* 5-PVCwB *which uses vertex v. Then there exists a solution F' that does not use vertex v and $|F'| \leq |F|$.*

Proof. From Lemma 2 we get that each P_5 in G which contains v must also start in v, otherwise it would imply a P_5 that uses more than one red vertex. Suppose that there exists a path $P = (v, w, x, y, z)$ where w is a red vertex and $\{x, y, z\} \cap F = \emptyset$ (see Figure 3.1). If there is no such P, then we have that each P_5 starting in v has at least one of the vertices x, y, z in F or there is no P_5 starting in v. In both cases we can put $F' = F \setminus \{v\}$ and the lemma holds.

There cannot exist another path $P' = (v, w, x', y', z')$ such that $x' \neq x$ and $\{x', y', z'\} \cap F = \emptyset$, otherwise we would have a $P_5 = (x', w, x, y, z)$ in G that is not hit by F. Consequently, each P_5 that is hit only by vertex v also contains vertex x, which implies that $F' = (F \setminus \{v\}) \cup \{x\}$ is a solution and $|F'| \leq |F|$, thus the lemma holds. □

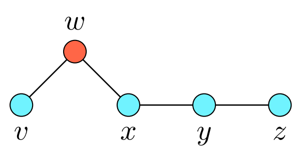


Figure 3.1: Configuration in rule (R3).

Branching rule (R3). Let v be an isolated vertex in $G[V_2]$ and let $P = (v, w, x, y, z)$ be a P_5 where w is a red vertex. Then branch on $\langle\, x \mid y \mid z \,\rangle$.

Proof of correctness. From Lemma 3 we know that if there exists a solution, then there exists a solution that does not contain v. Therefore branching on $\langle\, x \mid y \mid z \,\rangle$ is correct. $\bowtie$

Lemma 4. *Assume that Rules (R0) – (R3) are not applicable. Then there are no isolated vertices in $G[V_2]$.*

Proof. For contradiction assume that Rules (R0) − (R3) are not applicable and there is an isolated vertex v in $G[V_2]$. If there is no P_5 that uses v, then Rule (R1) is applicable on v. So suppose that there is a P_5 path P that uses v. If there are at least two red vertices connected to v, then there also exists a P_5 path P' that uses v and at least two red vertices and Rule (R2) is applicable. So suppose that there is only one red vertex w connected to v. Then Rule (R3) is applicable. $\square$

3.5 Dealing with isolated edges in $G[V_2]$

Lemma 5. *Assume that Rules (R0) – (R3) are not applicable. Let v be a blue vertex to which at least two red vertices are connected and let C_v be a connected component of $G[V_2]$ which contains v. Then for each red vertex w connected to v we have that $N(w) \subseteq V(C_v)$.*

Proof. Let w_1, w_2 be red vertices connected to v. For contradiction assume that w_1 is connected to some vertex v' in $G[V_2]$ such that $v' \notin V(C_v)$. From Lemma 4 we know that v' has degree at least one in $G[V_2]$. Label some neighbor of v' in $G[V_2]$ as u'. We obtained a $P_5 = (u', v', w_1, v, w_2)$ which contradicts Lemma 2. $\square$

Lemma 6. *Assume that Rules (R0) – (R3) are not applicable. Let $e = \{u, v\}$ be a blue edge to which at least two red vertices are connected in a way that to both u and v there is at least one red vertex connected. Let C_e be a connected component of $G[V_2]$ which contains e. Then for each red vertex w connected to e we have that $N(w) \subseteq V(C_e)$.*

Proof. Let w_1, w_2 be red vertices connected to e and assume that w_1 is connected to u and w_2 is connected to v. For contradiction assume that w_1 is connected to some vertex v' in $G[V_2]$ such that $v' \notin V(C_e)$. We obtain a $P_5 = (v', w_1, u, v, w_2)$ which contradicts Lemma 2. $\square$

Lemma 7. *Let X be a subset of V_2 such that $N(X) \cap V_1 = \emptyset$ and $|N(X) \cap V_2| = 1$. If there exists a solution F such that $F \cap X \neq \emptyset$, then there exists a solution F' such that $F' \cap X = \emptyset$ and $|F'| \leq |F|$.*

Proof. Assume that $N(X) \cap V_2 = \{v\}$. Then each P_5 that uses some vertex in X must also use vertex v, otherwise it would be contained in X which contradicts $G[V_2]$ being P_5-free. Consequently, any P_5 that is hit by a vertex from X in the solution F can be also hit by vertex v and thus $F' = (F \setminus X) \cup \{v\}$ is also a solution and $|F'| \leq |F|$. □

Definition 11. We say that two nodes x, y are symmetric if $N(x) \setminus \{y\} = N(y) \setminus \{x\}$.

Lemma 8. *Let x, y be blue vertices that are symmetric. Let F be a solution and $x \in F$. Then at least one of the following holds:*

(1) $y \in F$

(2) $F' = (F \setminus \{x\}) \cup \{y\}$ *is a solution*

Proof. Assume that $x \in F$ and $y \notin F$. Since x, y are symmetric, for each path $P = (p_1, p_2, p_3, p_4, p_5)$ with $p_i = x$ and $y \notin P$, there also exists a path $P' = (p'_1, p'_2, p'_3, p'_4, p'_5)$ such that $p'_j = p_j$ for $j \in (\{1, 2, 3, 4, 5\} \setminus \{i\})$ and $p'_i = y$. Firstly, if there is no P_5 containing x, then trivially *(2)* holds. Secondly, if all P_5 paths that contain x are hit by some other vertex $z, z \neq x, z \in F$, then again *(2)* holds. So suppose that there exists a P_5 path P that is hit only by x. If $y \notin P$, then we know that there is a path P' as described above and we get a contradiction with F being a solution since P' is not hit by F and *(1)* must hold. Otherwise, all P_5 paths that contain x also contain y and *(2)* holds. □

Branching rule (R4). Let $e = \{u, v\}$ be an isolated edge in $G[V_2]$. We know from Lemmata 5 and 6 that there is only one red vertex w connected to e, because if there were at least two red vertices connected to e, then there would be no P_5 that uses vertices from e. Let there be a red vertex w connected to at least one vertex in e. If w is connected only to one vertex in e, let that vertex be v (see Figure 3.2). Then branch on $\langle\, v \mid x \mid y \,\rangle$.

Proof of correctness. Firstly, assume that w is connected only to one vertex of e. Then from Lemma 7 we know that we do not have to try vertex u. Secondly, assume that w is connected to both vertices of e. Since u, v are symmetric, from Lemma 8 it follows that we can try deleting only one of them. Thus branching on $\langle\, v \mid x \mid y \,\rangle$ is correct. ⋈

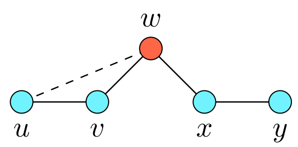


Figure 3.2: Configuration in rule (R4).

Lemma 9. *Assume that Rules (R0) – (R4) are not applicable. Then there are no isolated edges in* $G[V_2]$.

Proof. For contradiction assume that Rules (R0) – (R4) are not applicable and there is an isolated edge $e = \{x, y\}$ in $G[V_2]$. If there is no P_5 that uses vertices from e, then Rule (R1) is applicable on e. If there are at least two red vertices connected to e, then from Lemmata 5 and 6 we know that those red vertices are not connected to any other vertices outside e and there again cannot be a P_5 that uses vertices from e and Rule (R1) is applicable on e.

So suppose that there is a P_5 that uses vertices from e and there is only one red vertex w connected to e. But then Rule (R4) is applicable in both cases where w is connected to both vertices in e or to exactly one vertex in e. □

3.6 Dealing with isolated P_3 paths in $G[V_2]$

Context rule (R5). Let P be a $P_3 = (t, u, v)$ in $G[V_2]$. From Lemmata 2, 5 and 6 we know that there is only one red vertex w connected to P. We further know that w must be connected to some component of $G[V_2]$ other than P, otherwise no P_5 could be formed. Assume that x is some vertex to which w connects outside P and let y be a neighbor of x in $G[V_2]$. This rule is split into four subrules (R5.1), (R5.2), (R5.3) and (R5.4) based on how w is connected to P.

Branching rule (R5.1). Vertex w is connected only to v (see Figure 3.3a). Then branch on $\langle\, v \mid x \,\rangle$.

Proof of correctness. If we do not delete vertex x, then we have to delete something in P. From Lemma 7 we know that we do not have to try vertices t, u. Thus branching on $\langle\, v \mid x \,\rangle$ is correct. ⋈

Branching rule (R5.2). Vertex w is connected only to u, v (see Figure 3.3b). Then branch on $\langle\, u \mid v \mid x \,\rangle$.

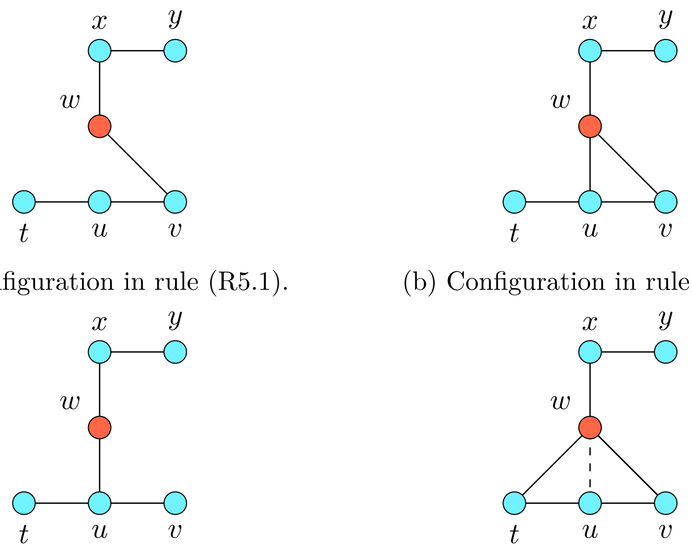
Proof of correctness. If we do not delete vertex x, then we have to delete something in P. From Lemma 7 we know that we do not have to try vertex t. Thus branching on $\langle\, u \mid v \mid x \,\rangle$ is correct. ⋈

Branching rule (R5.3). Vertex w is connected only to u (see Figure 3.3c). Then branch on $\langle\, u \mid x \mid y \,\rangle$.

Proof of correctness. If none of the vertices x, y is deleted, then we have to delete something in P. From Lemma 7 we know that we do not have to try vertices t, v. Thus branching on $\langle\, u \mid x \mid y \,\rangle$ is correct. ⋈

Branching rule (R5.4). Vertex w is connected to t, v and w can be also connected to u (see Figure 3.3d). Then branch on $\langle\, u \mid v \mid x \,\rangle$.

Proof of correctness. If we do not delete vertex x, then we have to delete something in P. In both cases, when w is connected to u and when not, t, v



(a) Configuration in rule (R5.1).

(b) Configuration in rule (R5.2).

(c) Configuration in rule (R5.3).

(d) Configuration in rule (R5.4).

Figure 3.3

are symmetric and from Lemma 8 we know that we have to try only one of t, v. Thus branching on $\langle\, u \mid v \mid x \,\rangle$ is correct. $\bowtie$

Lemma 10. *Assume that Rules (R0) – (R5) are not applicable. Then there are no isolated P_3 paths in $G[V_2]$.*

Proof. For contradiction assume that Rules (R0) – (R5) are not applicable and there is an isolated P_3 path $P = (t, u, v)$ in $G[V_2]$. If there is no P_5 that uses vertices from P, then Rule (R1) is applicable on P. Suppose there are at least two red vertices connected to P. If they are connected to vertices t, v, then Rule (R2) is applicable, since there is a P_5 that uses at least two red vertices. So suppose the red vertices are connected to a single vertex or a single edge in P. Then from Lemmata 5 and 6 we know that those red vertices are not connected to any other vertices outside P. Consequently, there cannot be a P_5 that uses vertices from P and again Rule (R1) is applicable on P.

So suppose that there is a P_5 that uses vertices from P and there is only one red vertex w connected to P. There are seven possibilities how w can be connected to P from which only five are not mutually isomorphic. Table 3.1 summarizes which rule should be applied in each situation (for clarity the isomorphic cases are omitted). $\square$

Table 3.1: Possible configurations of w and P in Lemma 10.

$N(w) \cap V(P)$	Rule to apply
$\{u\}$	(R5.3)
$\{v\}$	(R5.1)
$\{t, v\}$	(R5.4)
$\{u, v\}$	(R5.2)
$\{t, u, v\}$	(R5.4)

3.7 Dealing with isolated triangles in $G[V_2]$

Context rule (R6). Let T be a $K_3 = \{t, u, v\}$ in $G[V_2]$. From Lemmata 2 and 5 we know that there is only one red vertex w connected to T. We further know that w must be connected to some component of $G[V_2]$ other than T, otherwise no P_5 could be formed. Assume that x is some vertex to which w connects outside T and let y be a neighbor of x in $G[V_2]$. This rule is split into three subrules (R6.1), (R6.2) and (R6.3) based on how w is connected to T.

Branching rule (R6.1). Vertex w is connected only to one vertex in T, let that vertex be v (see Figure 3.4a). Then branch on $\langle\, v \mid x \,\rangle$.

Proof of correctness. If we do not delete vertex x, then we have to delete something in T. From Lemma 7 we know that we do not have to try vertices t, u. Thus branching on $\langle\, v \mid x \,\rangle$ is correct. ⋈

Branching rule (R6.2). Vertex w is connected to exactly two vertices in T, let those vertices be u, v (see Figure 3.4b). Then branch on $\langle\, t \mid v \mid x \,\rangle$.

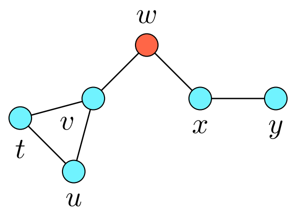
Proof of correctness. As in Rule (R6.1), if we do not delete vertex x, then we have to delete something in T. Since u, v are symmetric, from Lemma 8 we know that we have to try only one of u, v. Thus branching on $\langle\, t \mid v \mid x \,\rangle$ is correct. ⋈

Branching rule (R6.3). Vertex w is connected to all vertices in T (see Figure 3.4c). Then branch on $\langle\, v \mid x \,\rangle$.

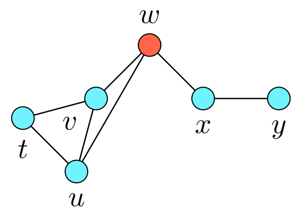
Proof of correctness. As in Rule (R6.1), if we do not delete vertex x, then we have to delete something in T. Since vertices in T are pairwise symmetric, from Lemma 8 we know that we have to try only one of t, u, v. Thus branching on $\langle\, v \mid x \,\rangle$ is correct. ⋈

Lemma 11. *Assume that Rules (R0) – (R6) are not applicable. Then there are no isolated triangles in* $G[V_2]$*.*

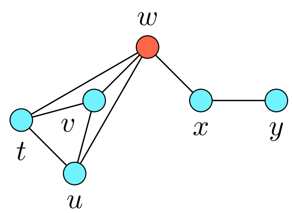
Proof. For contradiction assume that Rules (R0) − (R6) are not applicable and there is an isolated triangle $T = \{t, u, v\}$ in $G[V_2]$. If there is no P_5 that



(a) Configuration in rule (R6.1).



(b) Configuration in rule (R6.2).



(c) Configuration in rule (R6.3).

Figure 3.4

uses vertices from T, then Rule (R1) is applicable on T. Suppose there are at least two red vertices connected to T. If the red vertices are not connected to a single vertex in T, then Rule (R2) is applicable, since there is a P_5 that uses at least two red vertices. So suppose the red vertices are connected to a single vertex in T. Then from Lemma 5 we know that those red vertices are not connected to any other vertices outside T. Consequently, there cannot be a P_5 that uses vertices from T and again Rule (R1) is applicable on T.

So suppose that there is a P_5 that uses vertices from T and there is only one red vertex w connected to T. There are seven possibilities how w can be connected to T from which only three are not mutually isomorphic. Table 3.2 summarizes which rule should be applied in each situation (for clarity the isomorphic cases are omitted). □

Table 3.2: Possible configurations of w and T in Lemma 11.

$N(w) \cap V(T)$	Rule to apply
$\{v\}$	(R6.1)
$\{u, v\}$	(R6.2)
$\{t, u, v\}$	(R6.3)

3.8 Dealing with 4-cycles in $G[V_2]$

Lemma 12. *Let C be a connected component of $G[V_2]$ and $X = V(C) \cap N(V_1)$. Let F be a solution that deletes at least $|X|$ vertices in C. Then $F' = (F \setminus$*

$V(C)) \cup X$ *is also a solution and* $|F'| \leq |F|$.

Proof. Each P_5 that uses some vertex in C must also use some vertex $x \in X$, otherwise it would be contained in C which contradicts $G[V_2]$ being P_5-free. Consequently, any P_5 that is hit by a vertex from C in the solution F can be also hit by some vertex $x \in X$ and thus $F' = (F \setminus V(C)) \cup X$ is also a solution and $|F'| \leq |F|$. □

Context rule (R7). Let Q be a subgraph of K_4 such that 4-cycle is a subgraph of Q, label the vertices of the 4-cycle (v_1, v_2, v_3, v_4). We will call pairs of vertices $\{v_1, v_3\}$ and $\{v_2, v_4\}$ *diagonal*, all other pairs will be called *non-diagonal*. Edges corresponding to diagonal (non-diagonal) pairs are called diagonal (non-diagonal) edges, respectively. This rule is split into two subrules (R7.1), (R7.2) based on the number of red vertices connected to Q.

Reduction rule (R7.1). Assume that there are at least two red vertices connected to Q. Then delete v_1 and add it to the solution F.

Proof of correctness. We have to delete something in Q. From Lemmata 2, 5 and 6 we know that if there are at least two red vertices connected to Q, then they must be connected either to a single vertex or a single edge in Q and these vertices are not connected to any component in $G[V_2]$ other than Q.

Firstly, consider the case *(a)* when the red vertices are connected to a single vertex, let it be v_1 (see Figure 3.5a). Then from Lemma 12 we know that we only have to try deleting v_1. Thus deleting v_1 and adding it to the solution F is correct.

Secondly, consider the case *(b)* when the red vertices are connected to the vertices of a single edge, let them be v_1, v_2 (see Figure 3.5b). Observe that there are no diagonal edges in Q, since they would allow a P_5 that uses at least two red vertices, which would contradict Lemma 2. Also observe that the red vertices are connected to v_1 or v_2 by exactly one edge, i.e. there is not a red vertex among them connected to both v_1 and v_2, otherwise we would contradict Lemma 2 again. Consequently, after deleting v_1 there can be no P_5 formed in the component containing Q. Thus deleting v_1 and adding it to the solution F is correct. ⋈

Context rule (R7.2). Assume that there is only one red vertex w connected to Q and $X = V(Q) \cap N(w)$. This rule is split into five subrules (R7.2a), (R7.2b), (R7.2c), (R7.2d) and (R7.2e) based on how w is connected to Q and whether w is connected to other components.

Reduction rule (R7.2a). Vertex w is connected only to one vertex in Q, let it be v_1 (see Figure 3.6a). Then delete v_1 and add it to the solution F.

Proof of correctness. We have to delete something in Q and Lemma 12 implies that we have to try only v_1, thus deleting v_1 and adding it to the solution F is correct. ⋈

(a) Case *(a)* configuration in rule (R7.1). (b) Case *(b)* configuration in rule (R7.1).

Figure 3.5

Branching rule (R7.2b). Set X contains at least one diagonal pair, let that pair be $\{v_1, v_3\}$ (see Figure 3.6b). Then branch on $\langle\, v_1 \mid v_2 \mid v_4 \,\rangle$.

Proof of correctness. We have to delete something in Q. Since v_1, v_3 are symmetric, from Lemma 8 we know that we have to try only one of v_1, v_3. Thus branching on $\langle\, v_1 \mid v_2 \mid v_4 \,\rangle$ is correct. ⋈

Observation. Rule (R7.2b) also covers configurations where $|X| \geq 3$, since the conditions of the rule would be satisfied in that case.

Branching rule (R7.2c). Set X contains exactly one non-diagonal pair, let that pair be $\{v_1, v_2\}$, and case *(a)* either both diagonal edges are in Q (see Figure 3.6c), or case *(b)* none of them is (see Figure 3.6d). Then branch on $\langle\, v_1 \mid v_3 \mid v_4 \,\rangle$

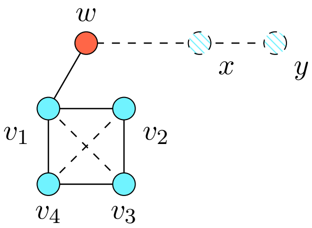
Proof of correctness. We have to delete something in Q. Vertices v_1, v_2 are symmetric and Lemma 8 applies. Thus branching on $\langle\, v_1 \mid v_3 \mid v_4 \,\rangle$ is correct. ⋈

Reduction rule (R7.2d). Set X contains exactly one non-diagonal pair, let that pair be $\{v_1, v_2\}$ and exactly one diagonal edge is in Q, let that edge be $\{v_1, v_3\}$. Furthermore, w is connected only to Q, i.e. $N(w) \subseteq V(Q)$ (see Figure 3.6e). Then delete any vertex v_i in Q and add it to the solution F.

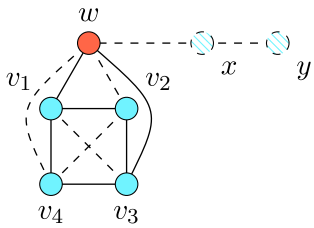
Proof of correctness. Since w is connected only to Q, after deleting some vertex in Q, there can be no P_5 formed in the component containing Q. Thus deleting any vertex v_i in Q and adding it to the solution F is correct. ⋈

Branching rule (R7.2e). Set X contains exactly one non-diagonal pair, let that pair be $\{v_1, v_2\}$ and exactly one diagonal edge is in Q, let that edge be $\{v_1, v_3\}$. Furthermore, w is connected to at least one more component of $G[V_2]$ other than Q, label the vertex to which w connects outside Q as x and let y be a neighbor of x in $G[V_2]$ (see Figure 3.6f). Then branch on $\langle\, \{v_1, v_2\} \mid x \mid y \,\rangle$.

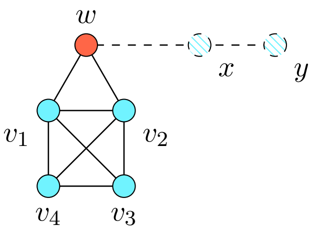
Proof of correctness. If none of the vertices x, y is deleted, then we have to delete at least two vertices in Q. From Lemma 12 we know that we only have to try deleting vertices $\{v_1, v_2\}$. Thus branching on $\langle\, \{v_1, v_2\} \mid x \mid y\,\rangle$ is correct. $\bowtie$



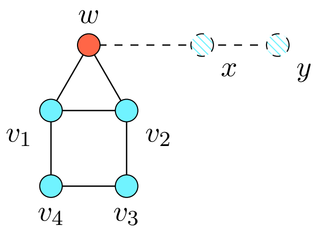
(a) Configuration in rule (R7.2a).



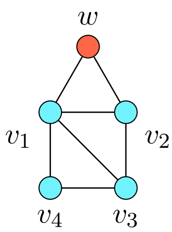
(b) Configuration in rule (R7.2b).



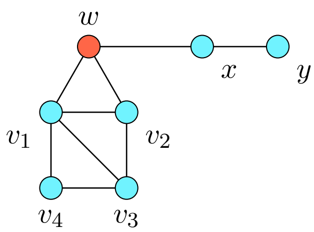
(c) Case *(a)* configuration in rule (R7.2c).



(d) Case *(b)* configuration in rule (R7.2c).



(e) Configuration in rule (R7.2d).



(f) Configuration in rule (R7.2e).

Figure 3.6

Lemma 13. *Assume that Rules (R0) – (R7) are not applicable. Then there is no component of $G[V_2]$ that contains a 4-cycle as a subgraph.*

Proof. For contradiction assume that Rules (R0) – (R7) are not applicable and there is a component Q in $G[V_2]$ that contains a 4-cycle as a subgraph, label the vertices of the 4-cycle (v_1, v_2, v_3, v_4). Observe that Q is a subgraph of K_4, as otherwise there would be a P_5 in $G[V_2]$.

If there is no P_5 that uses vertices from Q, then Rule (R1) is applicable on Q. Suppose there are at least two red vertices connected to Q. If the red vertices are not connected to a single vertex or a single edge in Q, then Rule (R2) is applicable, since there is a P_5 that uses at least two red vertices. So suppose the red vertices are connected to a single vertex or a single edge in Q. Then from Lemmata 5 and 6 we know that those red vertices are not connected to any other vertices outside Q and in both cases Rule (R7.1) is applicable.

So suppose that there is a P_5 that uses vertices from Q and there is only one red vertex w connected to Q. We consider only not mutually isomorphic possibilities how w is connected to Q. Firstly, in the case where there are no diagonal edges in Q the possibilities are summarized in Table 3.3a. Secondly, in the case where there are both diagonal edges in Q the possibilities are summarized in Table 3.3b.

Finally, in the case where there is only one diagonal edge in Q, let that edge be $\{v_1, v_3\}$, there is one exception when w is connected to both v_1 and v_2 where we need to consider whether w is connected only to Q ($N(w) \subseteq V(Q)$), or w is connected also outside Q ($N(w) \not\subseteq V(Q)$). The case with only one diagonal edge is summarized in Table 3.3c. □

3.9 Dealing with stars in $G[V_2]$

Recall the definition of a *star*. A star is a graph S with vertices $V(S) = \{s\} \cup \{l_1, \ldots, l_k\}$, $k \geq 3$ and edges $E(S) = \{\{s, l_i\} \mid i \in \{1, \ldots, k\}\}$. Vertex s is called a center, vertices $L = \{l_1, \ldots, l_k\}$ are called leaves.

Context rule (R8). Let S be a star in $G[V_2]$. This rule is divided into three subrules (R8.1), (R8.2) and (R8.3) based on how w is connected to S.

Lemma 14. *Assume that Rules (R0) – (R7) are not applicable. Then there is only one red vertex connected to S.*

Proof. For contradiction assume that Rules (R0) – (R7) are not applicable and that there are at least two red vertices connected to S. If they are connected to two different leaves, then we get a contradiction with Lemma 2. So suppose they are connected to the set $\{s, l_i\}$ for some $i \in \{1, \ldots, k\}$. From Lemmata 5 and 6 we know, that the red vertices are not connected to a component of $G[V_2]$ other than S and therefore there can be no P_5 formed in the component containing S. Thus the vertices of the component containing S are not used by any P_5 in G and the rule (R1) is applicable. □

Branching rule (R8.1). A red vertex w is connected to at least two leaves of S, let those two leaves be l_1, l_2 (see Figure 3.7a). Then branch on $\langle\, l_1 \mid s \mid L \setminus \{l_1, l_2\}\,\rangle$.

Proof of correctness. We have to delete something in S, since there is a path $P_5 = (l_1, w, l_2, s, l_i)$ for some $i \in \{3, \ldots, k\}$. From Lemma 14 we know that w is the only red vertex connected to S.

Suppose that we do not delete any vertex from $\{s, l_1, l_2\}$. Then the only thing we can do is to delete each vertex in $L \setminus \{l_1, l_2\}$, otherwise we would not hit all paths in S.

Table 3.3: Possible configurations of w and Q in Lemma 13.

(a) No diagonal edges in Q.

$N(w) \cap V(Q)$	Rule to apply
$\{v_1\}$	(R7.2a)
$\{v_1, v_2\}$	(R7.2c)
$\{v_1, v_3\}$	(R7.2b)
$\{v_1, v_2, v_3\}$	(R7.2b)
$\{v_1, v_2, v_3, v_4\}$	(R7.2b)

(b) Both diagonal edges in Q.

$N(w) \cap V(Q)$	Rule to apply
$\{v_1\}$	(R7.2a)
$\{v_1, v_2\}$	(R7.2c)
$\{v_1, v_2, v_3\}$	(R7.2b)
$\{v_1, v_2, v_3, v_4\}$	(R7.2b)

(c) One diagonal edge in Q, let it be $\{v_1, v_3\}$.

$N(w) \cap V(Q)$	Rule to apply
$\{v_1\}$	(R7.2a)
$\{v_2\}$	(R7.2a)
$\{v_1, v_2\}$ and $N(w) \subseteq V(Q)$	(R7.2d)
$\{v_1, v_2\}$ and $N(w) \not\subseteq V(Q)$	(R7.2e)
$\{v_1, v_3\}$	(R7.2b)
$\{v_2, v_4\}$	(R7.2b)
$\{v_1, v_2, v_3\}$	(R7.2b)
$\{v_1, v_2, v_4\}$	(R7.2b)
$\{v_1, v_2, v_3, v_4\}$	(R7.2b)

Now, assume that we did not delete all vertices from $L \setminus \{l_1, l_2\}$, label x a vertex from $L \setminus \{l_1, l_2\}$ that is not deleted. Suppose that we do not delete any vertex from $\{l_1, l_2\}$. Then we have to delete s, otherwise a path (l_1, w, l_2, s, x) would remain.

Finally, assume that we did not even delete s, now we have to delete something in $\{l_1, l_2\}$. Since l_1, l_2 are symmetric, from Lemma 8 we know that we have to try only one of l_1, l_2. Therefore branching on $\langle\, l_1 \mid s \mid L \setminus \{l_1, l_2\}\,\rangle$ is correct. $\bowtie$

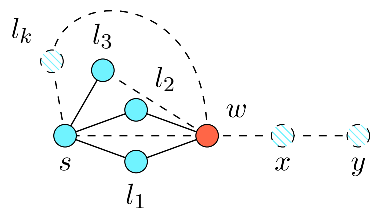
Observation. Assume that Rules (R0) – (R8.1) are not applicable. Then the red vertex w connected to S is connected only to a subset of $\{s, l_i\}$ for some $i \in \{1, \dots, k\}$, assume that the set w connects to is a subset of $\{s, l_1\}$. Also observe that w must be connected to at least one component of $G[V_2]$ other than S.

Branching rule (R8.2). A red vertex w is connected only to s and w is connected to some other vertex x in $G[V_2]$ outside S and y is a neighbor of x in $G[V_2]$ (see Figure 3.7b). Then branch on $\langle\, s \mid x \mid y \,\rangle$.

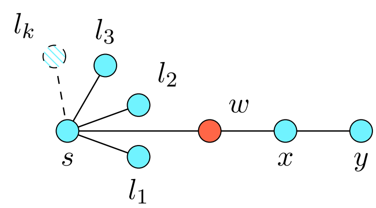
Proof of correctness. If none of the vertices x, y is deleted, then we have to delete something in S. From Lemma 7 we know, that we do not have to try any vertex in L. Thus branching on $\langle\, s \mid x \mid y \,\rangle$ is correct. $\bowtie$

Branching rule (R8.3). A red vertex w is connected to l_1, w can be connected also to s, and w is connected to some other vertex x in $G[V_2]$ outside S (see Figure 3.7c). Then branch on $\langle\, s \mid l_1 \mid x \,\rangle$.

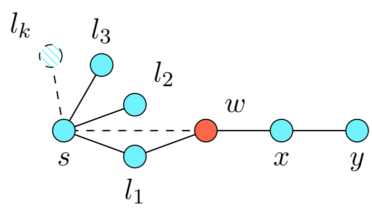
Proof of correctness. If we do not delete vertex x, then we have to delete something in S. From Lemma 7 we know, that we do not have to try any vertex in $L \setminus \{l_1\}$. Thus branching on $\langle\, s \mid l_1 \mid x \,\rangle$ is correct. $\bowtie$



(a) Configuration in rule (R8.1).



(b) Configuration in rule (R8.2).



(c) Configuration in rule (R8.3).

Figure 3.7

Lemma 15. *Assume that Rules (R0) – (R8) are not applicable. Then there are no stars in* $G[V_2]$.

Proof. For contradiction assume that Rules (R0) – (R8) are not applicable and there is a star S in $G[V_2]$.

If there is no P_5 that uses vertices from S, then Rule (R1) is applicable on S. Suppose there are at least two red vertices connected to S. If the red vertices are not connected to a single vertex or a single edge in S, then Rule (R2) is applicable, since there is a P_5 that uses at least two red vertices. So suppose the red vertices are connected to a single vertex or a single edge in S. Then from Lemmata 5 and 6 we know that those red vertices are not connected to any other vertices outside S. Consequently, there cannot be a P_5 that uses vertices from S and again Rule (R1) is applicable on S.

So suppose that there is a P_5 that uses vertices from S and there is only one red vertex w connected to S. If w is connected to two leaves, then Rule (R8.1) is applicable. So suppose that w is not connected to two leaves. There are three not mutually isomorphic possibilities how w can be connected to S and they are summarized in Table 3.4. □

Table 3.4: Possible configurations of w and S in Lemma 15.

$N(w) \cap V(S)$	Rule to apply
$\{l_1\}$	(R8.3)
$\{s\}$	(R8.2)
$\{l_1, s\}$	(R8.3)

3.10 Dealing with stars with a triangle in $G[V_2]$

Recall the definition of *star with a triangle*. A star with a triangle is a graph $S^\triangle$ with vertices $V(S^\triangle) = \{s, t_1, t_2\} \cup \{l_1, \dots, l_k\}$, $k \geq 1$ and edges $E(S^\triangle) = \{\{s, t_1\}, \{s, t_2\}, \{t_1, t_2\}\} \cup \{\{s, l_i\} \mid i \in \{1, \dots, k\}\}$. Vertex s is called a center, vertices $T = \{t_1, t_2\}$ are called triangle vertices and vertices $L = \{l_1, \dots, l_k\}$ are called leaves.

Context rule (R9). Let $S^\triangle$ be a star with a triangle in $G[V_2]$. This rule is divided into four subrules (R9.1), (R9.2), (R9.3) and (R9.4) based on how w is connected to $S^\triangle$.

Branching rule (R9.1). There is a red vertex w such that $\{t_1, t_2\} \subseteq N(w)$ (see Figure 3.8a). Then branch on $\langle\, t_1 \mid s \mid L \,\rangle$.

Proof of correctness. The proof follows the same logic as in Rule (R8.1) where w was connected to l_1, l_2 instead of t_1, t_2. ⋈

Branching rule (R9.2). There is a red vertex w such that $|\{t_1, t_2\} \cap N(w)| = 1$, assume that w is connected to t_1 (see Figure 3.8b). Then branch on $\langle\, t_1 \mid s \mid L \,\rangle$.

Lemma 16. *Assume that Rules (R0) – (R9.1) are not applicable and the assumptions of Rule (R9.2) are satisfied. Let F be a solution that contains t_2, then at least one of the following holds:*

(1) $t_1 \in F$

(2) $F' = (F \setminus \{t_2\}) \cup \{t_1\}$ *is a solution*

Proof. If there is no P_5 containing t_2, then *(2)* trivially holds. Suppose that every P_5 that contains t_2 also contains t_1, then again *(2)* trivially holds. So assume that there is a P_5 labeled P that contains t_2 but does not contain t_1. If for each such P there is some vertex x such that $x \neq t_2$ and $x \in F$, then *(2)* holds, since t_2 is not needed in the solution. Finally assume that $V(P) \cap F = \{t_2\}$, then, since P does not contain t_1, P must start at t_2 and $P = (t_2, p_1, p_2, p_3, p_4)$. But then there also exists a path $P' = (t_1, p_1, p_2, p_3, p_4)$ and P' is not hit, which is a contradiction with F being a solution and *(1)* must hold. □

Proof of correctness. We have to delete something in $S^{\triangle}$. Similarly as in Rule (R8.1) suppose that we do not delete any vertex from $\{s, t_1, t_2\}$. Then the only thing we can do is to delete each vertex in L.

So assume that we did not delete all vertices from L, label some remaining vertex from L as x. If we do not delete anything in $\{t_1, t_2\}$, then we have to delete s.

Finally, from Lemma 16 we see that deleting only t_1 is sufficient and thus branching on $\langle\, t_1 \mid s \mid L \,\rangle$ is correct. ⋈

Branching rule (R9.3). There is a red vertex w connected to a leaf, let that leaf be l_1, i.e. $l_1 \in N(w)$ (see Figure 3.8c). Then branch on $\langle\, l_1 \mid s \,\rangle$.

Proof of correctness. We have to delete something from $\{l_1, s, t_1, t_2\}$. Since there is no red vertex connected to any of $\{t_1, t_2\}$, Lemma 7 applies on $\{t_1, t_2\}$ and we have to try only vertices from $\{l_1, s\}$, therefore branching on $\langle\, l_1 \mid s \,\rangle$ is correct. ⋈

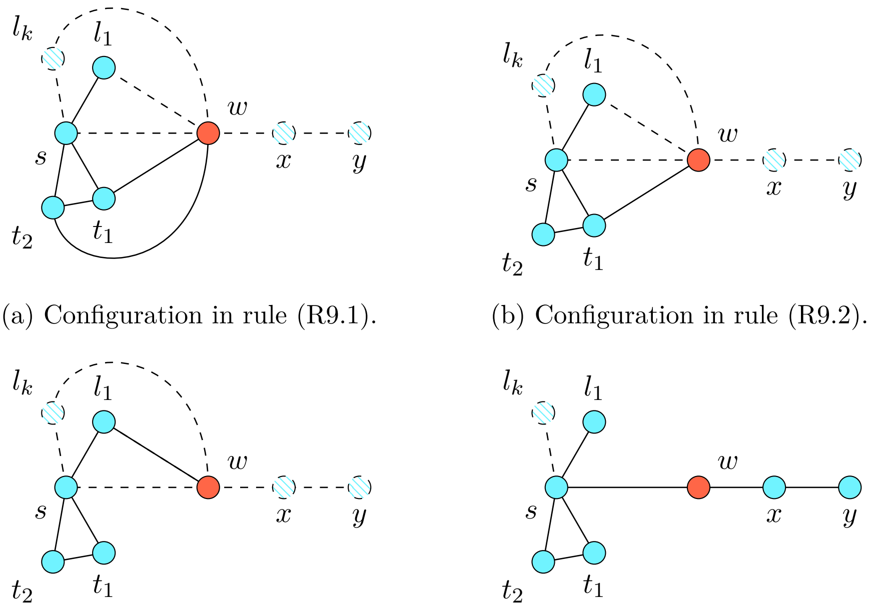
Branching rule (R9.4). A red vertex w is connected only to s. Also w must be connected to some component of $G[V_2]$ other than $S^{\triangle}$, otherwise no P_5 would occur in the component containing $S^{\triangle}$. Label the vertex to which w connects outside $S^{\triangle}$ as x (see Figure 3.8d). Then branch on $\langle\, s \mid x \,\rangle$.

Proof of correctness. If we do not delete vertex x, then we have to delete something in $S^{\triangle}$. Since there is no red vertex connected to $L \cup V(T)$, by Lemma 7 we have to try only s. Thus branching on $\langle\, s \mid x \,\rangle$ is correct. ⋈

Lemma 17. *Assume that Rules (R0) – (R9) are not applicable. Then there are no stars with a triangle in* $G[V_2]$.

Proof. For contradiction assume that Rules (R0) – (R9) are not applicable and there is a star with a triangle $S^{\triangle}$ in $G[V_2]$.

If there is no P_5 that uses vertices from $S^{\triangle}$, then Rule (R1) is applicable on $S^{\triangle}$. So suppose that there is a P_5 that uses vertices from $S^{\triangle}$, which implies



(a) Configuration in rule (R9.1). (b) Configuration in rule (R9.2).

(c) Configuration in rule (R9.3). (d) Configuration in rule (R9.4).

Figure 3.8

that there is at least one red vertex connected to $S^\triangle$, label one of those red vertices as w.

If w is connected to both t_1, t_2, then Rule (R9.1) is applicable. So suppose that w is not connected to both t_1, t_2. If w is connected to one of t_1, t_2, then Rule (R9.2) is applicable. So suppose that w is not connected to any of t_1, t_2. If w is connected to a leaf, then Rule (R9.3) is applicable.

Now we are in the situation in which the red vertices can be connected only to the center of $S^\triangle$. Firstly, if there are at least two red vertices connected to the center of $S^\triangle$, then from Lemma 5 these vertices are not connected to any other vertices outside $S^\triangle$. Consequently, there is no P_5 that uses vertices from $S^\triangle$ and again Rule (R1) is applicable on $S^\triangle$. Rule (R1) is also applicable if there is only one red vertex connected to the center of $S^\triangle$ and that vertex is connected to no other component in $G[V_2]$.

Finally, if there is only one red vertex w connected to the center of $S^\triangle$ and w is also connected to some vertices outside $S^\triangle$, then Rule (R9.4) is applicable. $\square$

3.11 Dealing with di-stars in $G[V_2]$

Recall the definition of a *di-star*. A di-star is a graph D with vertices $V(D) = \{s, s'\} \cup \{l_1, \dots, l_k\} \cup \{l'_1, \dots, l'_m\}$, $k \geq 1, m \geq 1$ and edges $E(D) = \{\{s, s'\}\}$

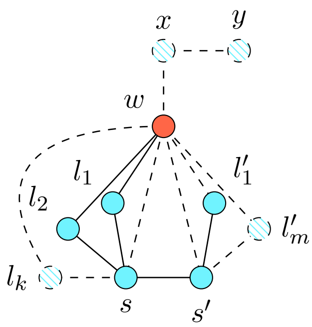


Figure 3.9: Configuration in rule (R10).

$\cup \{\{s, l_i\} \mid i \in \{1, \ldots, k\}\} \cup \{\{s', l'_j\} \mid j \in \{1, \ldots, m\}\}$. Vertices s, s' are called centers, vertices $L = \{l_1, \ldots, l_k\}$ and $L' = \{l'_1, \ldots, l'_m\}$ are called leaves.

Branching rule (R10). Let D be a di-star in $G[V_2]$ and let there be a red vertex w connected to at least two leaves on the same side of the di-star, i.e. $|N(w) \cap L| \geq 2$ or $|N(w) \cap L'| \geq 2$. Assume that those leaves are from L and l_1, l_2 are among them (see Figure 3.9). Then branch on $\langle\, l_1 \mid s \mid s' \,\rangle$.

Proof of correctness. We have to delete something in $\{l_1, l_2, s, s'\}$ and since l_1, l_2 are symmetric, from Lemma 8 we know that we have to try only one of them, thus branching on $\langle\, l_1 \mid s \mid s' \,\rangle$ is correct. $\bowtie$

Observation. Assume that Rules (R0) – (R10) are not applicable. In the following rules we have to consider only configurations where the red vertices are connected to a subset of $\{l_1, s, s', l'_1\}$.

Context rule (R11). Let D be a di-star in $G[V_2]$ and let there be a red vertex w connected to both s, s'. This rule is split into three subrules (R11.1), (R11.2), and (R11.3) based on the degrees of s and s'.

Context rule (R11.1). Assume that both s, s' have degree two in $G[V_2]$, i.e. the di-star D is actually a P_4. This rule is split into four subrules (R11.1a), (R11.1b), (R11.1c), and (R11.1d) based on how w is connected to D and whether w is connected to other components.

Branching rule (R11.1a). Vertex w is connected only to s, s' (see Figure 3.10a). Then branch on $\langle\, s \mid s' \,\rangle$.

Proof of correctness. We have to delete something in D and from Lemma 7 we know that we do not have to try vertices in L and L'. Thus branching on $\langle\, s \mid s' \,\rangle$ is correct. $\bowtie$

Branching rule (R11.1b). Vertex w is connected to s, s' and to one leaf, let that leaf be l_1 (see Figure 3.10b). Then branch on $\langle\, l_1 \mid s \mid s' \,\rangle$.

Proof of correctness. We have to delete something in D and from Lemma 7 we know that we do not have to try vertex l_1'. Thus branching on $\langle\, l_1 \mid s \mid s' \,\rangle$ is correct. $\bowtie$

Branching rule (R11.1c). Vertex w is connected to l_1, l_1', s, s' and to at least one other component of $G[V_2]$, label the vertex w connects to outside D as x and the neighbor of x in $G[V_2]$ as y (see Figure 3.10c). Then branch on $\langle\, x \mid y \mid \{l_1, s'\} \mid \{s, l_1'\} \mid \{s, s'\} \,\rangle$.

Proof of correctness. If none of the vertices x, y is deleted, then we have to delete at least two vertices in D. Assume that we want to delete only two vertices in D. Out of six possible pairs of vertices only $\{l_1, s'\}, \{s, l_1'\}, \{s, s'\}$ lead to a solution. Deleting more than two vertices in D also deletes at least one of the pairs $\{l_1, s'\}, \{s, l_1'\}, \{s, s'\}$. Thus branching on $\langle\, x \mid y \mid \{l_1, s'\} \mid \{s, l_1'\} \mid \{s, s'\} \,\rangle$ is correct. $\bowtie$

Reduction rule (R11.1d). Vertex w is connected only to l_1, l_2, s, s' and to no other component of $G[V_2]$ (see Figure 3.10d). Then delete any vertex v in D and add it to the solution F.

Proof of correctness. From Lemma 2 we know that there is no red vertex other than w connected to D, thus after deleting any vertex v in D and adding it to the solution F there is not enough vertices in D to from a P_5 with w. $\bowtie$

Context rule (R11.2). Assume that exactly one of s, s' has degree at least 3 in $G[V_2]$, let it be s. This rule is split into four subrules (R11.2a), (R11.2b), (R11.2c), and (R11.2d) based on how w is connected to D.

Branching rule (R11.2a). Vertex w is connected only to s, s' (see Figure 3.11a). Then branch on $\langle\, s \mid s' \,\rangle$.

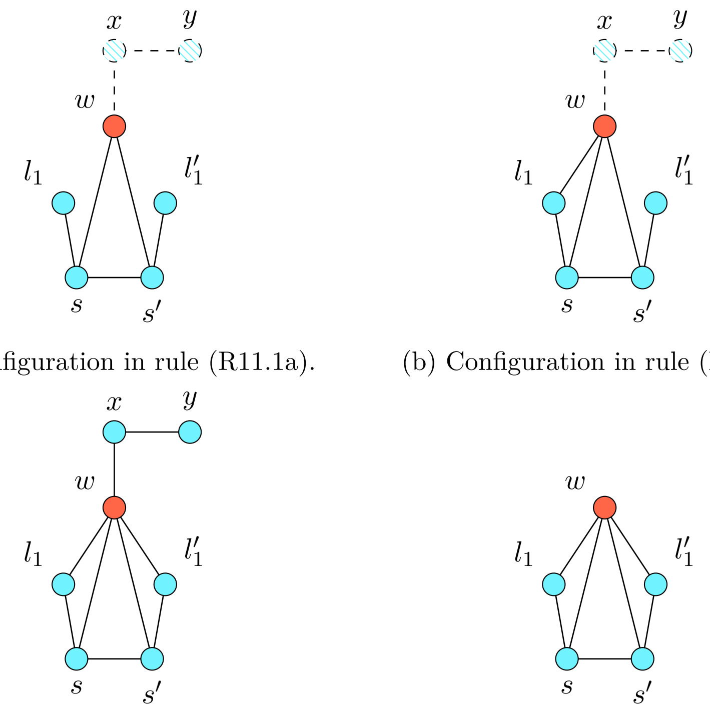
Proof of correctness. We have to delete something in D and from Lemma 7 we know that we do not have to try vertices in L and L'. Thus branching on $\langle\, s \mid s' \,\rangle$ is correct. $\bowtie$

Branching rule (R11.2b). Vertex w is connected to s, s' and exactly one leaf from L, let that leaf be l_1 (see Figure 3.11b). Then branch on $\langle\, l_1 \mid s \mid s' \,\rangle$.

Proof of correctness. We have to delete something in D and from Lemma 7 we know that we do not have to try vertices in $L \setminus \{l_1\}$ and L'. Thus branching on $\langle\, l_1 \mid s \mid s' \,\rangle$ is correct. $\bowtie$

Branching rule (R11.2c). Vertex w is connected to s, s', l_1' (see Figure 3.11c). Then branch on $\langle\, l_1' \mid s \mid s' \,\rangle$.

Proof of correctness. We have to delete something in D and from Lemma 7 we know that we do not have to try vertices in L. Thus branching on $\langle\, l_1' \mid s \mid s' \,\rangle$ is correct. $\bowtie$



(a) Configuration in rule (R11.1a). (b) Configuration in rule (R11.1b). (c) Configuration in rule (R11.1c). (d) Configuration in rule (R11.1d).

Figure 3.10

Branching rule (R11.2d). Vertex w is connected to s, s', l'_1 and exactly one leaf from L, let that leaf be l_1 (see Figure 3.11d). Then branch on $\langle\, l_1 \mid s \mid s' \,\rangle$.

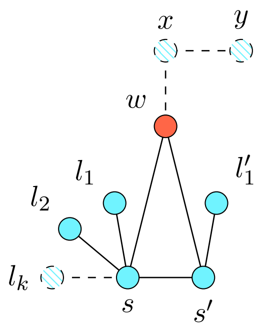
Proof of correctness. Let l_2 be some other leaf from L, $l_2 \neq l_1$. We have to delete something in $\{l_1, l_2, s, s'\}$ and from Lemma 7 we know that we do not have to try l_2. Thus branching on $\langle\, l_1 \mid s \mid s' \,\rangle$ is correct. ⋈

Branching rule (R11.3). Assume that both s, s' have degree at least 3 in $G[V_2]$ (see Figure 3.12). Then branch on $\langle\, L \mid s \mid s' \mid L' \,\rangle$.

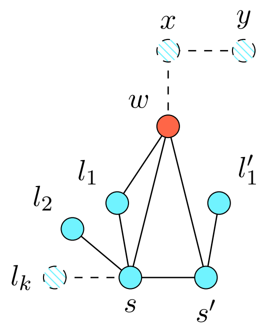
Proof of correctness. Assume that none of the vertices s, s' is deleted and that neither whole L, nor whole L' is deleted. Let l_1 be a not deleted leaf from L and l'_1 not deleted leaf from L'. That implies a $P_5 = (l_1, s, w, s', l'_1)$ in D and hence at least one whole side of the di-star must be deleted to get a solution. Thus branching on $\langle\, L \mid s \mid s' \mid L' \,\rangle$ is correct. ⋈

Observation. Assume that Rules (R0) – (R11) are not applicable. In the following rules we have to consider only configurations where the red vertices are connected to a subset of $\{l_1, s, s', l'_1\}$, but not to both s and s'.

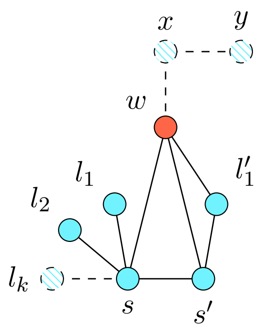
Context rule (R12). Let D be a di-star in $G[V_2]$ and let there be a red vertex w connected by two edges to D. We know that w is connected to



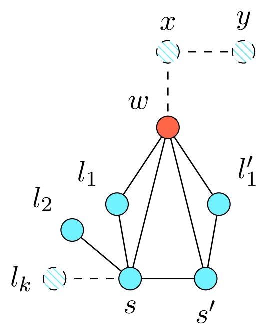
(a) Configuration in rule (R11.2a).



(b) Configuration in rule (R11.2b).



(c) Configuration in rule (R11.2c).



(d) Configuration in rule (R11.2d).

Figure 3.11

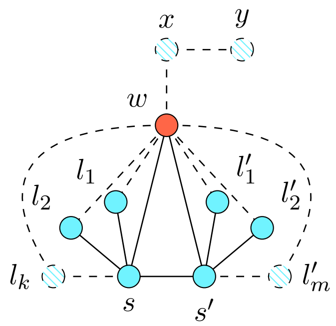
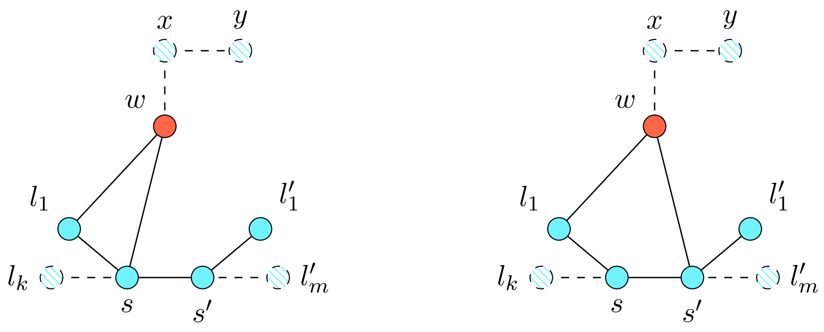


Figure 3.12: Configuration in rule (R11.3).

a subset of $\{l_1, s, s', l'_1\}$, but not to both s and s'. This rule is split into three subrules (R12.1), (R12.2), and (R12.3) based on how w is connected to D.

Branching rule (R12.1). Vertex w is connected to a center and its leaf, let them be s and l_1 (see Figure 3.13a). Then branch on $\langle\, l_1 \mid s \,\rangle$.

Proof of correctness. We have to delete something in D and from Lemma 7 we do not have to try vertices other than l_1 and s, thus branching on $\langle\, l_1 \mid s \,\rangle$ is correct. $\bowtie$



(a) Configuration in rule (R12.1). (b) Configuration in rule (R12.2).

Figure 3.13

Branching rule (R12.2). Vertex w is connected to a center and to a leaf of the other center, let them be s' and l_1 (see Figure 3.13b). Then branch on $\langle\, l_1 \mid s \mid s' \,\rangle$.

Proof of correctness. We have to delete something in D and from Lemma 7 we do not have to try vertices other than l_1, s and s', thus branching on $\langle\, l_1 \mid s \mid s' \,\rangle$ is correct. ⋈

Context rule (R12.3). Vertex w is connected to two opposite leaves, let them be l_1 and l_1'. This rule is split into four subrules (R12.3a), (R12.3b), (R12.3c), and (R12.3d) based on the degrees of s and s' and whether w is connected to other components.

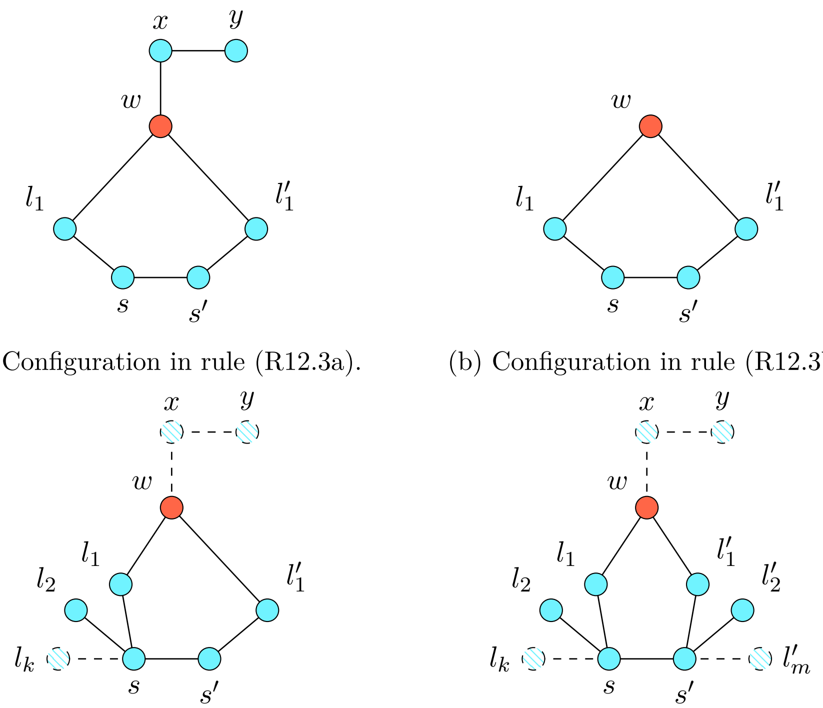
Branching rule (R12.3a). Both s, s' have degree 2 in $G[V_2]$ and w is connected to a component of $G[V_2]$ other than D, let x be the vertex w connects to outside D and let y be a neighbor of x in $G[V_2]$ (see Figure 3.14a). Then branch on $\langle\, x \mid y \mid \{l_1, l_1'\} \,\rangle$.

Proof of correctness. If none of the vertices x, y is deleted, then we have to delete at least two vertices in D and from Lemma 12 we know that we only have to try to delete $\{l_1, l_1'\}$. Therefore branching on $\langle\, x \mid y \mid \{l_1, l_1'\} \,\rangle$ is correct. ⋈

Reduction rule (R12.3b). Both s, s' have degree 2 in $G[V_2]$ and w is not connected to a component of $G[V_2]$ other than D (see Figure 3.14b). Then delete any vertex v in D and add it to the solution F.

Proof of correctness. We have to delete something in D and after deleting any vertex v in D and adding it to the solution F, there is not enough vertices in the component containing D to form a P_5. ⋈

Branching rule (R12.3c). Exactly one of s, s' has degree at least 3 in $G[V_2]$, let it be s (see Figure 3.14c). Then branch on $\langle\, l_1 \mid s \mid l_1' \,\rangle$.



(a) Configuration in rule (R12.3a). (b) Configuration in rule (R12.3b).

(c) Configuration in rule (R12.3c). (d) Configuration in rule (R12.3d).

Figure 3.14

Proof of correctness. Let l_2 be some leaf from $L \setminus \{l_1\}$. We have to delete something in $\{l_1, l_2, s, l'_1\}$ and from Lemma 7 we know that we do not have to try vertex l_2. Thus branching on $\langle\, l_1 \mid s \mid l'_1 \,\rangle$ is correct. ⋈

Branching rule (R12.3d). Both s, s' have degree at least 3 in $G[V_2]$ (see Figure 3.14d). Then branch on $\langle\, s \mid s' \mid \{l_1, l'_1\} \,\rangle$.

Proof of correctness. Assume that none of the vertices s, s' is deleted. If we do not delete both l_1, l'_1, then at least one of L or L' must be wholly deleted. Since both L and L' have size at least 2, we would delete at least two vertices in D and by Lemma 12 we can choose $\{l_1, l'_1\}$ instead. Thus branching on $\langle\, s \mid s' \mid \{l_1, l'_1\} \,\rangle$ is correct. ⋈

Context rule (R13). Let D be a di-star in $G[V_2]$ and let there be a red vertex w connected by three edges to D. We know that w is connected to a subset of $\{l_1, s, s', l'_1\}$, but not to both s and s'. Assume that w is connected to l_1, s, l'_1. This rule is split into four subrules (R13.1), (R13.2), (R13.3) and (R13.4) based on the degrees of s and s' and whether w is connected to other components.

Branching rule (R13.1). Both s, s' have degree 2 in $G[V_2]$ and w is connected to at least one other component of $G[V_2]$, label the vertex w connects to outside D as x and the neighbor of x in $G[V_2]$ as y (see Figure 3.15a). Then branch on $\langle\, x \mid y \mid \{l_1, s'\} \mid \{s, l_1'\}\,\rangle$.

Proof of correctness. If none of the vertices x, y is deleted, then we have to delete at least two and from Lemma 12 at most three vertices in D. Suppose we wanted to delete exactly two vertices. Out of six possible pairs, only $\{l_1, s'\}, \{s, s'\}, \{s, l_1'\}$ lead to a solution. We do not have to try $\{s, s'\}$, since if we delete s, then Lemma 7 becomes applicable and we may delete l_1' instead of s'. Finally, if we wanted to delete three vertices, then by Lemma 12 those vertices would be $\{l_1, s, l_1'\}$, but this is already covered by branching on $\{s, l_1'\}$. Thus branching on $\langle\, x \mid y \mid \{l_1, s'\} \mid \{s, l_1'\}\,\rangle$ is correct. ⋈

Reduction rule (R13.2). Both s, s' have degree 2 in $G[V_2]$ and w is not connected to other component of $G[V_2]$ (see Figure 3.15b). Then delete any vertex v in D and add it to the solution F.

Proof of correctness. We have to delete something in D and after deleting any vertex v of D and adding it to the solution F, there is not enough vertices in D to form a P_5 with w. ⋈

Branching rule (R13.3). Vertex s has degree at least 3 in $G[V_2]$ (see Figure 3.15c). Then branch on $\langle\, l_1 \mid s \mid l_1'\,\rangle$.

Proof of correctness. We have to delete something in $\{l_1, l_2, s, l_1'\}$ and from Lemma 7 we know that we do not have to try vertex l_2, thus branching on $\langle\, l_1 \mid s \mid l_1'\,\rangle$ is correct. ⋈

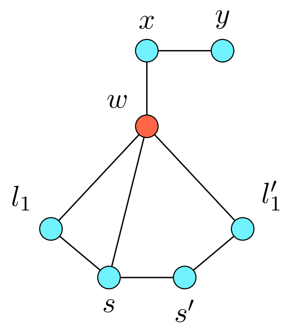
Branching rule (R13.4). Vertex s' has degree at least 3 in $G[V_2]$ (see Figure 3.15d). Then branch on $\langle\, l_1 \mid s' \mid l_1'\,\rangle$.

Proof of correctness. We have to delete something in $\{l_1, s', l_1', l_2'\}$ and from Lemma 7 we know that we do not have to try vertex l_2', thus branching on $\langle\, l_1 \mid s' \mid l_1'\,\rangle$ is correct. ⋈

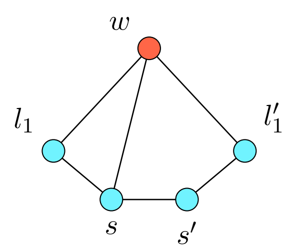
Observation. Assume that Rules (R0) – (R13) are not applicable. In the following rules we have to consider only configurations where the red vertices are connected to exactly one of $\{l_1, s, s', l_1'\}$.

Context rule (R14). There is exactly one red vertex w connected to D by one edge. This rule is split into two subrules (R14.1) and (R14.2) based on how w is connected to D.

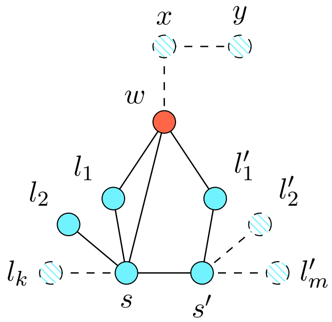
Reduction rule (R14.1). Vertex w is connected to a leaf, let it be l_1 (see Figure 3.16a). Then delete l_1 and add it to the solution F.



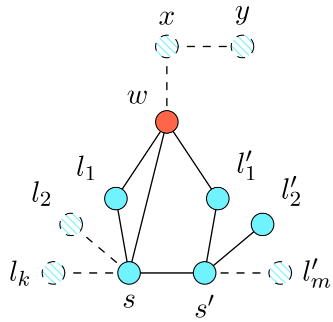
(a) Configuration in rule (R13.1).



(b) Configuration in rule (R13.2).



(c) Configuration in rule (R13.3).



(d) Configuration in rule (R13.4).

Figure 3.15

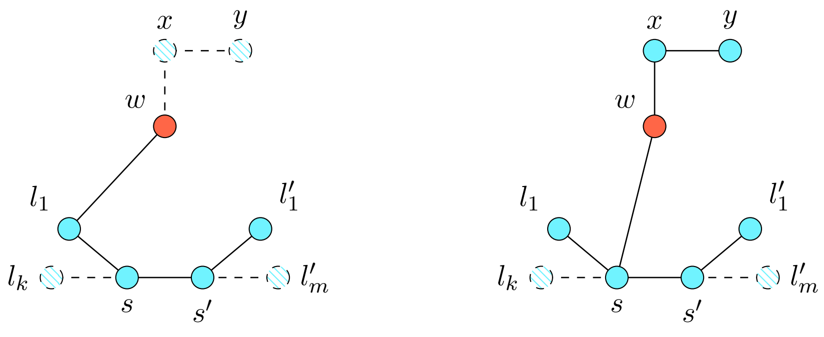
Proof of correctness. We have to delete something in D and from Lemma 12 we know that deleting l_1 and adding it to the solution F is correct. ⋈

Branching rule (R14.2). Vertex w is connected to a center, let it be s, and w is connected to at least one component of $G[V_2]$ other than D, label the vertex w connects to outside D as x (see Figure 3.16b). Then branch on $\langle\, s \mid x \,\rangle$.

Proof of correctness. If we do not delete vertex x, then we have to delete something in D and from Lemma 12 we know that deleting s is sufficient. Thus branching on $\langle\, s \mid x \,\rangle$ is correct. ⋈

Reduction rule (R15). There are at least two red vertices connected to D by exactly one edge and they are connected to a single vertex. From Lemma 5 we know, that the red vertices are not connected to a component of $G[V_2]$ other than D and hence the single vertex must be a leaf, let it be l_1, otherwise no P_5 would be formed and Rule (R1) would be applicable. Then delete l_1 and add it to the solution F.

Proof of correctness. We have to delete something in D and from Lemma 12 we know that deleting l_1 and adding it to the solution F is correct. ⋈



(a) Configuration in rule (R14.1). (b) Configuration in rule (R14.2).

Figure 3.16

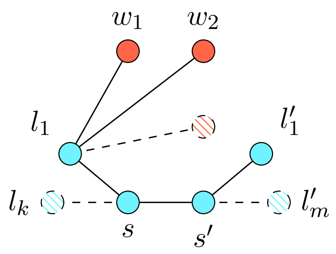


Figure 3.17: Configuration in rule (R15).

Branching rule (R16). There are at least two red vertices connected to D by exactly one edge and they are connected to two opposite leaves, let those leaves be l_1, l'_1. Assume that there is at least one red vertex connected to each one of them. Further assume that the red vertices connected to l_1 are not connected to a component of $G[V_2]$ other than D (see Figure 3.18). Then branch on $\langle\, s' \mid l'_1 \,\rangle$.

Proof of correctness. We have to delete something in D. From Lemma 12 we know, that we will delete at most two vertices from D and those vertices would be $\{l_1, l'_1\}$. Now suppose that we want to delete exactly one vertex from D. From Lemma 7 we know that we have to consider trying only vertices in $\{l_1, s, s', l'_1\}$. Assume that there exists a solution F that deletes either l_1 or s from D. Since F is a solution, if there is a P_5 that uses at least one of $\{s', l'_1\}$, then it must be hit by some vertex outside D.

And with that we know that either $F' = (F \setminus \{l_1, s\}) \cup \{s'\}$ or $F'' = (F \setminus \{l_1, s\}) \cup \{l'_1\}$ is also a solution since all P_5 paths that start in the red vertices connected to l_1 use at least one of $\{l'_1, s'\}$ (they use both if $|L'| = 1$) and $|F'| \leq |F|, |F''| \leq |F|$. Thus branching on $\langle\, s' \mid l'_1 \,\rangle$ is correct. ⋈

Observation. Assume that Rules (R0) – (R16) are not applicable. Then for each di-star component of $G[V_2]$ there are exactly two red vertices connected to

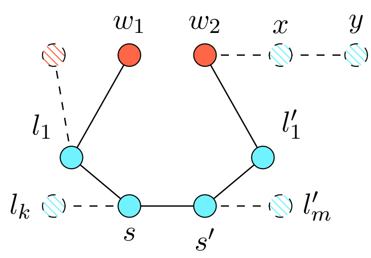


Figure 3.18: Configuration in rule (R16).

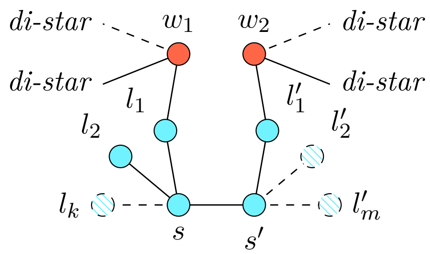


Figure 3.19: Configuration in rule (R17).

two opposite leaves in the di-star. Furthermore, each red vertex is connected to at least two different di-star components of $G[V_2]$.

Branching rule (R17). Let there be a di-star D and the two red vertices w, w' connected to D are connected to leaves l_1, l'_1, respectively, and at least one of the centers has degree at least three, let it be s (see Figure 3.19). Then branch on $\langle\, s \mid s' \mid l'_1 \,\rangle$.

Proof of correctness. We know that we have to delete something in D and we will delete at most two vertices from D. In the case where we delete two vertices from D, we delete vertices l_1, l'_1 by Lemma 12. So suppose that we want to delete exactly one vertex from D. It cannot be vertex l_1, since center s has degree at least three, thus there exists another leaf l_2 connected to s. This implies a $P_5 = (l_2, s, s', l'_1, w')$. Finally, from Lemma 7 we know that we do not have to try vertices in $L \setminus \{l_1\}$ and $L' \setminus \{l'_1\}$. Consequently, branching on $\langle\, s \mid s' \mid l'_1 \,\rangle$ is correct. ⋈

Branching rule (R18). Let there be a di-star D and the two red vertices w, w' connected to D are connected to leaves l_1, l'_1, respectively, and both centers have degree exactly two (see Figure 3.20). Then branch on $\langle\, l_1 \mid l'_1 \,\rangle$.

Proof of correctness. Observe that each di-star component of $G[V_2]$ is actually a P_4 now. Let F be a solution. Label the di-star components of $G[V_2]$ as $D_1, D_2, \ldots, D_r$. Observe that F deletes at least one vertex in each di-star component D_i.

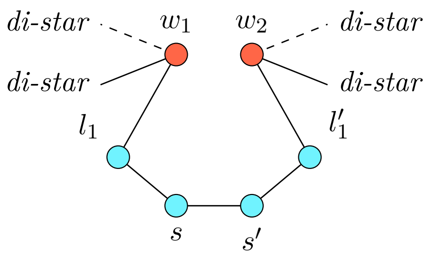


Figure 3.20: Configuration in rule (R18).

Firstly, we construct a directed graph G' such that $V(G') = V_1$ and there is an edge $e_i = (x, y)$ in G' if and only if F deletes exactly one vertex in D_i and the deleted vertex is either s_i^y or l_i^y where l_i^y is a leaf y connects to in D_i and s_i^y is the center of D_i to which l_i^y is connected.

We claim that each vertex in G' has outgoing degree at most one. Indeed, for contradiction assume that vertex w has outgoing degree at least two, which means that there are two di-star components D_i, D_j connected to w such that F does not contain the leaves w is connected to in D_i, D_j, let them be l_i^w, l_j^w and the centers to which these leaves are connected, let them be s_i^w, s_j^w. But that implies a $P_5 = (s_i^w, l_i^w, w, l_j^w, s_j^w)$ in G and F would not be a solution, which is a contradiction.

Secondly, we construct a set F' in the following way: (1) for each di-star component D_i where F deletes at least two vertices, add to F' the two leaves of D_i and (2) for each edge $e_j = (x, y)$ in G' add to F' a leaf connected to y in D_j.

Finally, F' is also a solution because in the di-star D_i where F deleted at least two vertices we know from Lemma 12 that it suffices to delete only the leaves of D_i and we claim that in the graph $G \setminus F'$ there is no P_5. Indeed, for contradiction assume that there is a P_5 in $G \setminus F'$. But that could only happen if there was a vertex w in G' with an outgoing degree at least two, which is a contradiction.

Therefore F' is a solution that uses only leaves of the di-stars in G and from construction of G' and F' we have that $|F'| \leq |F|$. Thus branching on $\langle\, l_1 \mid l'_1 \,\rangle$ is correct. $\bowtie$

Lemma 18. *Assume that Rules (R0) – (R9) are not applicable. Then at least one of Rules (R10) – (R18) is applicable.*

Proof. From Lemma 1 together with Lemmata 4, 9, 10, 11, 13, 15 and 17 we are now in the situation in which all components of $G[V_2]$ are di-stars and there must be a di-star D in $G[V_2]$ such that there is a P_5 that uses the vertices of D which implies there is at least one red vertex connected to D. For contradiction assume that Rules (R10) – (R18) are not applicable, i.e. no rules are applicable.

Let w be some red vertex connected to D. If $|N(w) \cap L| \geq 2$ or $|N(w) \cap L'| \geq 2$, then Rule (R10) is applicable. So for the rest of this proof assume that each red vertex can be connected only to vertices l_1, s, s', or l'_1.

Firstly, assume that there is only one red vertex w connected to D. In Table 3.5 we list all possibilities (omitting several isomorphic cases) based on how w is connected to D, on the degrees of s and s', and whether w is connected only to D ($N(w) \subseteq V(D)$) or w is also connected outside D ($N(w) \nsubseteq V(D)$).

Observe that if there were at least two red vertices connected to D and w was connected to D by at least two edges, then Rule (R2) would be applicable with the only exception in case where w is connected to $\{l_1, s\}$ or $\{s', l'_1\}$ and the other red vertices to s or s', respectively. But this exception is resolved by Rule (R1) since vertices connected only to s or s' in this configuration are not used by any P_5. With this in mind, if there are at least two red vertices connected to D, then they are connected to D by only one edge.

Secondly, assume that there are at least two red vertices connected to D by exactly one edge. Let $X \subseteq V(D)$ be the vertices to which the red vertices are connected in D. If $|X \cap L| \geq 2$ or $|X \cap L'| \geq 2$, then Rule (R2) is applicable, since there is a P_5 that uses at least two red vertices. So suppose that $X \subseteq \{l_1, s, s', l'_1\}$. If $\{l_1, s'\} \subseteq X$ or $\{s, l'_1\} \subseteq X$ (which covers also cases where $|X| \geq 3$), then again Rule (R2) is applicable. If the vertices are connected to a single edge, then at least one of the vertices of such edge is a center and the vertices connected to that center are not used by any P_5 in this configuration and Rule (R1) is applicable. We conclude that the red vertices may be connected only to a single vertex or to two opposite leaves in D.

Thirdly, assume that the red vertices are connected to a single vertex. If that vertex is a leaf, then Rule (R15) is applicable, otherwise Rule (R1) is applicable.

Fourthly, assume that the red vertices are connected to two opposite leaves, let them be l_1 and l'_1, and let W be the red vertices connected to l_1 and W' be the red vertices connected to l'_1. If the vertices in W or in W' (or both) are not connected to any component other than D, then Rule (R16) is applicable.

Observe that now we are in situation in which there are exactly two red vertices w and w' connected to D by exactly one edge and these vertices are connected to l_1 and l'_1, assume that w is connected to l_1 and w' is connected to l'_1. Furthermore, vertices w and w' are connected to at least one other di-star in $G[V_2]$. If at least one of L, L' has size at least two, then Rule (R17) is applicable, otherwise all di-stars in $G[V_2]$ are actually a P_4 paths and Rule (R18) is applicable.

Finally, there is no di-star remaining in $G[V_2]$ which together with Lemmata 1, 4, 9, 10, 11, 13, 15 and 17 implies that $G[V_2] = \emptyset$ and since V_1, V_2 is a P_5-free bipartition, there is no P_5 path remaining in G and Rule (R0) is applicable. $\square$

Table 3.5: Possible configurations of single red vertex w and D in Lemma 18.

$N(w) \cap V(D)$	$N(w) \not\subseteq V(D)$			$N(w) \subseteq V(D)$		
	$\lvert L\rvert = 1$, $\lvert L'\rvert = 1$	$\lvert L\rvert > 1$, $\lvert L'\rvert = 1$	$\lvert L\rvert > 1$, $\lvert L'\rvert > 1$	$\lvert L\rvert = 1$, $\lvert L'\rvert = 1$	$\lvert L\rvert > 1$, $\lvert L'\rvert = 1$	$\lvert L\rvert > 1$, $\lvert L'\rvert > 1$
$\{l_1\}$	(R14.1)	(R14.1)	(R14.1)	(R14.1)	(R14.1)	(R14.1)
$\{s\}$	(R14.2)	(R14.2)	(R14.2)	(R1)	(R1)	(R1)
$\{s'\}$	(R14.2)	(R14.2)	(R14.2)	(R1)	(R1)	(R1)
$\{l'_1\}$	(R14.1)	(R14.1)	(R14.1)	(R14.1)	(R14.1)	(R14.1)
$\{l_1, s\}$	(R12.1)	(R12.1)	(R12.1)	(R12.1)	(R12.1)	(R12.1)
$\{l_1, s'\}$	(R12.2)	(R12.2)	(R12.2)	(R12.2)	(R12.2)	(R12.2)
$\{l_1, l'_1\}$	(R12.3a)	(R12.3c)	(R12.3d)	(R12.3b)	(R12.3c)	(R12.3d)
$\{s, s'\}$	(R11.1a)	(R11.2a)	(R11.3)	(R11.1a)	(R11.2a)	(R11.3)
$\{s, l'_1\}$	(R12.2)	(R12.2)	(R12.2)	(R12.2)	(R12.2)	(R12.2)
$\{s', l'_1\}$	(R12.1)	(R12.1)	(R12.1)	(R12.1)	(R12.1)	(R12.1)
$\{l_1, s, s'\}$	(R11.1b)	(R11.2b)	(R11.3)	(R11.1b)	(R11.2b)	(R11.3)
$\{l_1, s, l'_1\}$	(R13.1)	(R13.3)	(R13.3)	(R13.2)	(R13.3)	(R13.3)
$\{l_1, s', l'_1\}$	(R13.1)	(R13.4)	(R13.3)	(R13.2)	(R13.4)	(R13.3)
$\{s, s', l'_1\}$	(R11.1b)	(R11.2c)	(R11.3)	(R11.1b)	(R11.2c)	(R11.3)
$\{l_1, s, s', l'_1\}$	(R11.1c)	(R11.2d)	(R11.3)	(R11.1d)	(R11.2d)	(R11.3)

3.12 Final remarks

Theorem 19. *For any values of the input parameters of the call to* DISJOINT_R *procedure at least one of Rules (R0) – (R18) is applicable.*

Proof. The theorem directly follows from Lemma 18. □

Theorem 20. *The* DISJOINT *procedure solves the* 5-PVCwB *problem in* $\mathcal{O}^*(3^k)$ *time.*

Proof. We use the technique of analysis of branching algorithms as described by Fomin and Kratsch [6].

Let $T(k)$ be the maximum number of leaves in any search tree of a problem instance with parameter k. We analyze each branching rule separately and finally use the worst-case bound on the number of leaves over all branching rules to bound the number of leaves in the search tree of the whole procedure.

Let $\langle X_1 \mid X_2 \mid \ldots \mid X_l \rangle$ be the branching rule to be analyzed. We have that $l \geq 2$ and $|X_i| \geq 1$. This implies the linear recurrence

$$T(k) \leq T(k - |X_1|) + T(k - |X_2|) + \cdots + T(k - |X_l|).$$

It is well known that the base solution of such linear recurrence is of the form $T(k) = \lambda^k$ where λ is a complex root of the polynomial

$$\lambda^k - \lambda^{k-|X_1|} - \lambda^{k-|X_2|} - \cdots - \lambda^{k-|X_l|} = 0$$

and the worst-case bound on the number of leaves of the branching rule is given by the unique positive root of the polynomial. This positive root λ is called a *branching factor*.

The worst-case upper bound of the number of leaves in the search tree of the whole procedure is the maximal branching factor among the branching factors of all the branching rules. In our case, the worst-case branching factor is 3 (see Table 3.6 for the branching factors), therefore the upper bound of the number of leaves in the search tree is $\mathcal{O}^*(3^k)$.

Now we have to upper bound the number of inner nodes in the search tree. We claim that each path from the root to some leaf of the search tree has at most $\mathcal{O}(|V(G)|)$ vertices. Indeed, each rule removes at least one vertex from G. Therefore the upper bound of the number of inner nodes in the search tree is $\mathcal{O}^*(3^k)$.

Since the running time of each rule (the work that is done in each node of the search tree) is polynomial in $|V(G)|$, we get that the worst-case running time of the whole procedure is $\mathcal{O}^*(3^k)$. □

Theorem 21. *The iterative compression algorithm solves the* 5-PVC *problem and runs in* $\mathcal{O}^*(4^k)$ *time.*

Proof. Take a look again at Algorithm 1. The compression routine on lines 7 – 23 is run at most $|V(G)|$ times and the worst case running time of one run of the compression routine can be computed as

$$\sum_{X \subsetneq F} O^*(3^{k-|X|}) = \sum_{i=0}^{k} \binom{k+1}{i} O^*(3^{k-i}) = O^*(4^k).$$

Therefore the final running time of the algorithm is $\mathcal{O}^*(4^k)$ and the 5-PVC problem is solvable in $\mathcal{O}^*(4^k)$ time when parameterized by the size of the solution k. □

Table 3.6: Branching factors λ of the branching rules.

Rule	λ	Rule	λ	Rule	λ	Rule	λ
(R2)	3	(R7.2b)	3	(R10)	3	(R12.2)	3
(R3)	3	(R7.2c)	3	(R11.1a)	2	(R12.3a)	2.415
(R4)	3	(R7.2e)	2.415	(R11.1b)	3	(R12.3c)	3
(R5.1)	2	(R8.1)	2.415	(R11.1c)	3	(R12.3d)	2.415
(R5.2)	3	(R8.2)	3	(R11.2a)	2	(R13.1)	2.733
(R5.3)	3	(R8.3)	3	(R11.2b)	3	(R13.3)	3
(R5.4)	3	(R9.1)	3	(R11.2c)	3	(R13.4)	3
(R6.1)	2	(R9.2)	3	(R11.2d)	3	(R14.2)	2
(R6.2)	3	(R9.3)	2	(R11.3)	2.733	(R16)	2
(R6.3)	2	(R9.4)	2	(R12.1)	2	(R17)	3
						(R18)	2

Chapter **4**

Experimental evaluation

We implemented both our iterative compression algorithm ALGO running in $\mathcal{O}^*(4^k)$ time and the trivial algorithm TRIVIAL running in $\mathcal{O}^*(5^k)$ time. We ran a few experiments to experimentally show that our algorithm ALGO is indeed faster than the trivial algorithm on instances with sufficiently large parameter k.

4.1 Environment

The experiments were run on a PC running Ubuntu Linux 17.10 with Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz processor and 16GB of RAM. The programming language used was `C++` with the `gcc` compiler version 7.2.0 with `-Ofast` optimizations enabled.

4.2 Datasets

We generated instances with varying numbers of vertices n and parameters k. The types of graphs we generated are: *path*, *random*, and *semi-random*.

- *path* – a simple path on n vertices
- *random* – a random graph on n vertices; each edge has the same probability of being in the graph
- *semi-random* – constructed in the following way: let $b = n - k$ and start with an empty graph G; (1) randomly and with uniform probability add P_5-free components to G until $|V(G)| = b$, the numbers of leaves of stars, stars with a triangle, and di-stars to be generated are also determined uniformly at random between the minimum number (from definition) and the remaining number of vertices; (2) add k isolated vertices $U = \{u_1, u_2, \ldots, u_k\}$ to G; (3) for each $u_i \in U$ pick k random vertices in

$V(G) \setminus U$ and make u_i adjacent to those vertices. This construction ensures that there is a solution for G with size at most k.

4.3 Implementation remark

We simplified finding the P_5 paths in the graph to trivial enumeration since it is still just a polynomial factor in the final running time. But we admit that in this area there is a lot of room to improve the algorithm so that it can process instances with large number of vertices but small parameter k.

4.4 Results

Tables 4.1, 4.2, and 4.3 and Figure 4.1 summarize the results. In the tables the abbreviations A and T stand for ALGO and TRIVIAL, respectively. The times measured are in seconds and they are the average running time of three runs on the same dataset. We set a hard time limit to 3600 seconds. If an algorithm exceeded this time limit, we stopped its execution and in the tables marked this fact with "> 3600".

It can be seen that the TRIVIAL algorithm performs better when the problem instance is small and with small parameter k. That is expected since the ALGO algorithm is far more complex and this significantly increases the multiplicative constant in its running time. But when parameter k gets sufficiently large, we see that the TRIVIAL algorithm is exponentially slower than the ALGO algorithm.

An unexpected phenomenon occurred in the *semi-random* datasets, where TRIVIAL algorithm performed much better than ALGO. We attribute this poor performance of ALGO algorithm to the complexity of determining which rule should be applied in given situation.

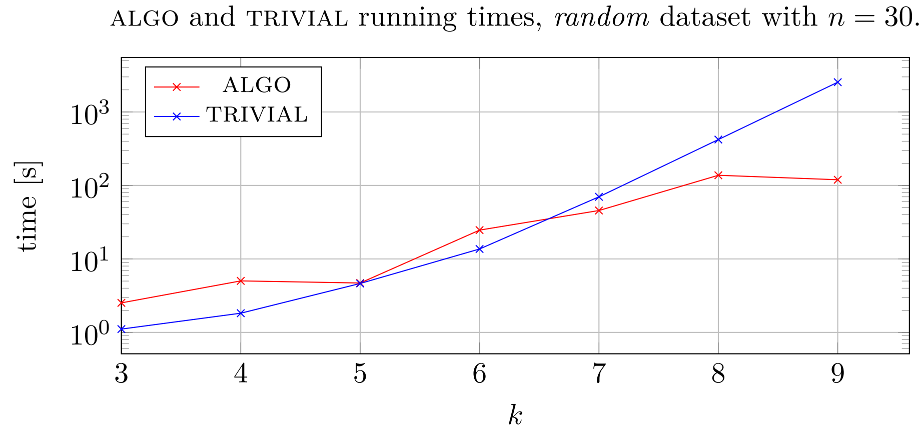


Figure 4.1: ALGO and TRIVIAL running times, *random* dataset with $n = 30$.

Table 4.1: Experimental results for path graphs.

n \ k	3		4		5		6		7		8		9	
	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]
25	0.17	**0.16**	0.18	**0.17**	**0.18**	0.21	**0.18**	0.34	**0.18**	0.71	**0.16**	1.98	**0.17**	4.05
30	**0.42**	0.50	0.46	**0.46**	0.55	**0.54**	**0.97**	1.06	**0.57**	2.69	**0.90**	6.95	**0.56**	20.16
35	**1.59**	1.61	**1.74**	2.10	**1.75**	1.82	**1.51**	2.52	**1.43**	5.37	**2.05**	16.96	**1.77**	61.05
40	**3.11**	3.59	**3.21**	3.63	**3.20**	3.75	**2.71**	4.47	**3.00**	10.26	**3.47**	36.29	**2.47**	137.98
45	5.39	**3.65**	4.67	**3.72**	4.33	**3.83**	5.25	**4.33**	6.60	**5.99**	**8.01**	18.27	**6.39**	85.48
50	10.14	**10.07**	**9.21**	9.95	**9.42**	10.25	**10.52**	11.29	**10.53**	14.38	**10.84**	29.87	**17.17**	108.36

Table 4.2: Experimental results for random graphs.

n \ k	3		4		5		6		7		8		9	
	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]
25	1.05	**0.36**	1.50	**0.59**	2.94	**1.37**	7.55	**7.01**	**14.92**	43.81	**67.90**	259.63	**48.37**	1445.88
30	2.52	**1.11**	5.02	**1.83**	4.70	**4.63**	24.67	**13.65**	**45.58**	70.03	**137.53**	421.14	**119.54**	2540.32
35	5.14	**4.95**	9.79	**5.18**	22.66	**9.30**	59.15	**36.33**	**84.39**	144.39	**224.00**	1143.30	**513.14**	> 3600
40	11.05	**4.03**	19.02	**4.81**	43.91	**8.45**	101.92	**35.33**	202.61	**170.73**	**748.84**	> 3600	**1966.01**	> 3600
45	34.98	**8.85**	62.65	**12.03**	186.45	**24.36**	504.62	**81.29**	**413.42**	420.22	**1187.00**	> 3600	> 3600	> 3600
50	33.05	**14.05**	76.87	**21.85**	113.35	**37.95**	230.77	**143.20**	**505.49**	667.61	**1240.12**	> 3600	> 3600	> 3600

Table 4.3: Experimental results for semi-random graphs.

n \ k	3		4		5		6		7		8		9	
	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]	A[s]	T[s]
25	**0.17**	0.19	**0.20**	0.21	0.56	**0.34**	**0.86**	1.08	13.11	**4.82**	**9.17**	37.98	**81.28**	265.24
30	0.76	**0.41**	0.64	**0.45**	1.19	**0.51**	3.73	**0.95**	8.20	**4.08**	**14.30**	29.15	721.98	**239.81**
35	1.58	**1.38**	1.77	**1.41**	**1.33**	1.70	4.87	**2.43**	19.60	**6.90**	**1.74**	33.59	240.13	**215.86**
40	**2.38**	2.85	**2.55**	3.07	2.97	**2.63**	6.78	**4.30**	21.77	**9.05**	48.95	**37.60**	796.34	**228.58**
45	**4.57**	5.38	**5.66**	5.91	**5.44**	6.59	12.18	**7.58**	20.87	**13.47**	134.92	**49.96**	1139.38	**272.54**
50	**10.52**	11.06	10.47	**9.62**	11.25	**10.45**	14.71	**12.27**	64.80	**17.81**	340.84	**56.00**	1224.74	**283.47**

Conclusion

We conclude this thesis with a few open questions.

Firstly, we see the trend of solving 3-PVC, 4-PVC and now 5-PVC with the iterative compression technique, so it is natural to ask whether this approach can be further used for 6-PVC or even to d-PVC in general. However, given the complexity (number of rules) of the algorithm presented in this thesis, it seems more reasonable to first try to find a simpler algorithm for 5-PVC.

Secondly, motivated by the work of Orenstein et al. [11], we ask whether known algorithms for 3-PVC, 4-PVC, 5-PVC can be generalized to work with directed graphs.

Finally, due to Fafianie and Kratsch [5] we know that d-PVC problem has a kernel with $\mathcal{O}(k^d)$ vertices and edges. Dell and van Melkebeek [4] showed that there is no $\mathcal{O}(k^{d-\epsilon})$ kernel for any $\epsilon > 0$ for general d-Hitting Set unless coNP is in NP/poly, which would imply a collapse of the polynomial-time hierarchy. However, for 3-PVC problem, Xiao and Kou [16] presented a kernel with $5k$ vertices. To our knowledge, it is not known whether there exists a linear kernel for 4-PVC or any d-PVC with $d \geq 5$.

Bibliography

[1] Brešar, B., Kardoš, F., Katrenič, J., and Semanišin, G. Minimum k-path vertex cover. *Discrete Applied Mathematics 159*, 12 (2011), 1189–1195.

[2] Chen, J., Kanj, I. A., and Xia, G. Improved upper bounds for vertex cover. *Theor. Comput. Sci. 411*, 40-42 (2010), 3736–3756.

[3] Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. *Parameterized Algorithms*. Springer, 2015.

[4] Dell, H., and van Melkebeek, D. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM 61*, 4 (2014), 23:1–23:27.

[5] Fafianie, S., and Kratsch, S. A shortcut to (sun)flowers: Kernels in logarithmic space or linear time. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II* (2015), pp. 299–310.

[6] Fomin, F. V., and Kratsch, D. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[7] Funke, S., Nusser, A., and Storandt, S. On k-path covers and their applications. *VLDB J. 25*, 1 (2016), 103–123.

[8] Katrenič, J. A faster FPT algorithm for 3-path vertex cover. *Inf. Process. Lett. 116*, 4 (2016), 273–278.

[9] Lewis, J. M., and Yannakakis, M. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci. 20*, 2 (1980), 219–230.

[10] Novotný, M. Design and analysis of a generalized canvas protocol. In *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, 4th IFIP WG 11.2 International Workshop, WISTP 2010, Passau, Germany, April 12-14, 2010. Proceedings* (2010), pp. 106–121.

[11] Orenstein, Y., Pellow, D., Marçais, G., Shamir, R., and Kingsford, C. Designing small universal k-mer hitting sets for improved analysis of high-throughput sequencing. *PLoS Computational Biology 13*, 10 (2017).

[12] Reed, B. A., Smith, K., and Vetta, A. Finding odd cycle transversals. *Oper. Res. Lett. 32*, 4 (2004), 299–301.

[13] Tu, J. A fixed-parameter algorithm for the vertex cover P_3 problem. *Inf. Process. Lett. 115*, 2 (2015), 96–99.

[14] Tu, J., and Jin, Z. An FPT algorithm for the vertex cover P_4 problem. *Discrete Applied Mathematics 200* (2016), 186–190.

[15] Xiao, M., and Kou, S. Exact algorithms for the maximum dissociation set and minimum 3-path vertex cover problems. *Theor. Comput. Sci. 657* (2017), 86–97.

[16] Xiao, M., and Kou, S. Kernelization and parameterized algorithms for 3-path vertex cover. In *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings* (2017), pp. 654–668.

[17] Xiao, M., and Nagamochi, H. Exact algorithms for maximum independent set. *Inf. Comput. 255* (2017), 126–146.

Contents of CD

```
readme.txt ........................ the file with CD contents description
thesis.pdf ......................... the Diploma thesis in PDF format
src ................................... the thesis source code directory
implementation ................ the algorithm implementation directory
   bin ......................................... the executable binaries
   data .................................................. the datasets
   measure ...................................... the measurements
```