Karlsruhe Institute of Technology

Packing Lower Bounds for Cluster Editing

Bachelor Thesis of

Maximilian David Walz

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	T.T.-Prof. Dr. Thomas Bläsius
	PD Dr. Torsten Ueckerdt
Advisors:	Christopher Weyand

Time Period: 16th Mai 2022 – 16th September 2022

Statement of Authorship

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, September 15, 2022

Abstract

CLUSTER EDITING is the problem of finding a minimal set of edge modifications to transform a given graph into a cluster graph - a graph whose connected components are cliques. A set of induced, vertex pair disjoint paths of length two in the given graph is a lower bound for the number of required edge modifications.

To find such sets, we define and examine a type of intersection graph, the *P_3-intersection graph*, that models which induced paths of length two, called P_3s, in a given graph share a vertex pair. Now, any independent set on the P_3-intersection graph yields a lower bound for the CLUSTER EDITING problem on the given graph. Unfortunately, MAXIMUM INDEPENDENT SET generally is NP-hard. However, the complexity on P_3-intersection graphs is unknown.

In this work, we show that the neighbourhood of every vertex in a P_3-intersection graph can be partitioned into three cliques characterized by which vertex pair the contained vertices share with the respective vertex. We also prove that edges between these cliques strictly follow a specific structure. Additionally, we briefly discuss possible approaches to show the NP-hardness of MAXIMUM INDEPENDENT SET on P_3-intersection graphs.
Finally, we transfer the concept of P_3-intersection graphs to stars and paths of arbitrary length. We find that it is infeasible to compute a maximum independent set on these types of intersection graphs.

Deutsche Zusammenfassung

CLUSTER EDITING ist das Problem, eine minimale Menge von Kantenmodifikationen zu finden, um einen gegebenen Graphen in einen Cluster-Graphen umzuwandeln. Dies ist ein Graph, bei dem alle Zusammenhangskomponenten Cliquen sind. Eine Menge an induzierten, Knotenpaar-disjunkten Pfaden der Länge zwei eines gegebenen Graphen, ist dabei eine untere Schranke für die Anzahl benötigter Kantenmodifikationen.

Um eine solche Menge zu finden, definieren und untersuchen wir eine Art Schnittgraph, den *P_3-intersection Graph*. Dieser modelliert welche induzierten Pfade der Länge zwei, genannt P_3s, eines gegebenen Graphen sich ein Knotenpaar teilen. Jede unabhängige Menge im P_3-intersection Graphen gibt uns nun eine untere Schranke für das CLUSTER EDITING Problem auf dem gegebenen Graphen. Leider ist MAXIMUM INDEPENDENT SET im Allgemeinen NP-schwer. Für P_3-intersection Graphen is die Komplextität allerdings noch unbekannt.

In dieser Arbeit zeigen wir, dass sich die Nachbarschaft jedes Knoten eines P_3-intersection Graphen in drei Cliquen partitionieren lässt, die sich darüber definieren, welches Knotenpaar die enthaltenen Knoten sich mit dem jeweiligen Knoten teilen. Wir beweisen außerdem, dass Kanten zwischen diesen Cliquen streng einer bestimmten Struktur folgen. Zusätzlich besprechen wir kurz unsere Versuche, die NP-Schwere von MAXIMUM INDEPENDENT SET auf P_3-intersection Graphen zu zeigen.
Schlussendlich übertragen wir das Konzept von P_3-intersection Graphen auf Sterne und Pfade beliebiger Länge. Wir finden dabei heraus, dass es nicht praktikabel ist, eine größte unabhängige Menge auf diesen Arten von Schnittgraphen zu berechnen.

Contents

1. Introduction

The Parameterized Algorithms and Computational Experiments (PACE) Challenge is an annual competition in which people from all over the world develop algorithms to solve a given problem [pac]. In 2021, the topic of the PACE Challenge was Cluster Editing [KKNZ21]. The goal is to transform a given graph into a cluster graph, a graph consisting solely of fully connected components, using a minimal set of edge insertions and deletions.

Bläsius et al., the winners of 2021's challenge, used a branch and bound approach [BFG$^+$22]. The general concept of this approach is to compute upper and lower bounds for a solution of a given problem, branch on a decision and calculate new lower bounds for the resulting instances. If this new lower bound is higher than a previous upper bound, the respective branch can be discarded. Otherwise, the branching and computing of bounds is repeated until a solution is found. In the context of Cluster Editing, the bounds refer to the number of required edge modifications. By *decision* we mean a problem specific choice needed to be made while solving the problem. In Cluster Editing, that choice could be whether an edge is in- or excluded.
The efficiency of this technique is highly dependent on the quality of the calculated bounds because the better the bounds, the sooner the branches can be discarded. In this work, we concentrate on finding lower bounds for Cluster Editing.

We can obtain lower bounds of a Cluster Editing instance by adding up lower bounds of non-overlapping familiar structures contained in the instance. A path of length two, a P_3, is a good candidate since it is a small structure consisting of only 3 vertices and can always be resolved with exactly one edge modification. Therefore, the number of non-overlapping P_3s in the instance infers an equal lower bound. To obtain a set of such non-overlapping P_3s, we can construct a graph which represents the P_3s of the instance and their intersections, the P_3-intersection graph. This way, we transform the problem of finding a set of non-overlapping P_3s into finding an independent set in the P_3-intersection graph.

In this work, after establishing some basic concepts of graph theory and formally introducing the Cluster Editing problem along with P_3-packings in Chapter 2, we define P_3-intersection graphs in Chapter 3 and cover some basic examples to get a basic understanding for their structure. In Section 3.3, we look at properties of P_3-intersection graphs, where we focus on the neighbourhood of their vertices. We continue with some invariants of P_3-intersection graph construction in Section 3.4 until briefly looking at the computational costs of the construction in Section 3.5 and finishing the chapter with an outline of our approaches to show the hardness of Maximum Independent Set on P_3-intersection graphs in Section 3.6.

In Chapter 4 We proceed to look at intersection graphs of stars. After discussing different definitions, we settle for the more promising one and compare these star-intersection graphs to the previously proposed P_3-intersection graphs.
Chapter 5 addresses another type of intersection graph based on paths of an arbitrary, but fixed, length. They are a generalization of P_3-intersection graphs, so we briefly examine how some properties of P_3-intersection graphs translate to these P_n-intersection graphs, as we will call them.
We conclude this work in Chapter 6, with a discussion of our results and the proposal of some problems, questions and approaches that could be pursued in future work.

Related Work

The study of interval graphs can be considered the origin of intersection graphs as a research topic [Gol88]. They are a simple type of intersection graph, representing the intersections of intervals on a line. For intersection graphs in general, consider a family of nonempty subsets of some set S. Then, its intersection graph has a vertex for each subset, with two vertices adjacent if and only if their corresponding subsets intersect. Additionally, there may be certain constraints for the subsets and their intersections. Other than a line, the set S could be a circle [Kle69], a plane [EET76] or a tree [GJ85]. There is, for example, the class of EPT graphs, which are the edge intersection graphs of collections of paths in trees [Gav78], [MW86] or the class of chordal graphs, which are the edge intersection graphs of subtrees in trees [Gav74]. As far as we know, there is little to no research concerning intersection graphs of paths with a fixed length in arbitrary graphs.

There is, however, very much research dedicated to Cluster Editing. In 1964, Zahn [Zah64] already solved the problem for a certain type of input graphs. Since then there have been many findings concerning parameterized complexity for several variants of the problem including, for example, Bicluster Graph Editing [PDdSS09], Temporal Cluster Editing [CMSS17] and Dynamic Cluster Editing [LMNN21]. Even tough Cluster Editing was proven to be NP-complete on arbitrary graphs [SST02], good heuristic [BFG$^+$21] and exact solvers [Bö12] have been developed over the last years. The currently fastest solver runs in $O(1.62^k)$ time, where k is the solution size, implying that Cluster Editing is fixed parameter tractable.

2. Preliminaries

In this chapter we introduce some general definitions and concepts of graph theory which will be used throughout this work as well as establish definitions specific to our topic. See [Die05] for a broader and more in-depth overview of graph theory basics.

2.1 Fundamentals

A *graph* $G = (V, E)$ is a tuple of *vertices* V and *edges* $E = \{\{u, v\} \mid u, v \in V\}$. An edge $\{u, v\}$ will be abbreviated by uv. Two vertices u, v are called *adjacent* if $uv \in E$. Unless stated otherwise, the graphs in this work are *simple graphs* meaning they are undirected, without parallel edges and loopless. An *induced subgraph* of a graph G is a graph $G[V'] \coloneqq (V', E')$ with $V' \subseteq V$ and $E' \coloneqq \{uv \in E \mid u, v \in V'\}$. In this work, the term *subgraph* always refers to an induced subgraph. The *neighbourhood* $N(v)$ of a vertex v is the set of vertices adjacent to v. Formally, $N(v) \coloneqq \{u \in V \mid uv \in E\}$. A *path* P_n with $n \in \mathbb{N}$ is a graph whose vertices can be seen as a sequence $(v_1, \ldots, v_n)$ in which every two consecutive vertices are adjacent. Note, that $v_1 v_n$ is a non-edge if the path is induced. See Figure 2.1 for an example. A *circle* C_n with $n \in \mathbb{N}$ is a graph whose vertices can be seen as a sequence $(v_1, \ldots, v_n)$ where every two consecutive vertices as well as the first and the final vertex are adjacent. The *complete graph* K_n with $n \in \mathbb{N}$ is a graph of n vertices where every two vertices are adjacent. Figure 2.2 shows the complete graph K_4 as an example. The *complete bipartite graph* $K_{n,m}$ with $n, m \in \mathbb{N}$ is a graph in which the vertices can be divided into two disjoint and independent sets U, V with $|U| = n, |V| = m$ where every vertex of U is adjacent to every vertex of V. The *k-star* with $k \in \mathbb{N}$ is the complete bipartite graph $K_{1,k}$, as seen with $k = 4$ in Figure 2.3. We hereby call the single vertex the center of the star and the k other vertices the leaves.

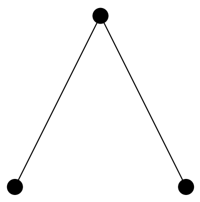
Figure 2.1: The P_3.

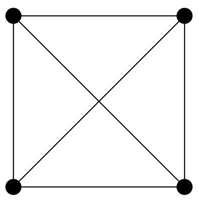
Figure 2.2: The K_4.

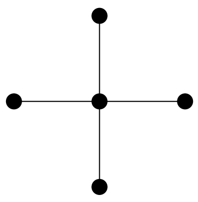
Figure 2.3: The 4-star.

A maximal subgraph where every two nodes are connected through a path is called a *connected component*. A set of pairwise adjacent vertices is called a *clique*, while a set of pairwise non-adjacent vertices is called an *independent set*. The set $\{1, \ldots, n\}$ including all positive integers from 1 to n will be denoted by $[n]$, and for a set M and a positive integer n, $\binom{M}{n} \coloneqq \{M' \subseteq M \mid |M'| = n\}$.

2.2 Cluster Editing

A graph whose connected components are cliques is called a *cluster graph*. If we add an edge to a graph or delete an existing one, we call that an *edge modification*. The CLUSTER EDITING problem asks to find a set of edge modifications of minimum size so that a given graph is transformed into a cluster graph.

Figure 2.4 shows a CLUSTER EDITING instance and the graph where the edge modifications of a possible solution are applied. The continuous red edge was added and the dashed red edges were deleted. This results in a cluster graph with two components: a clique of size four and a clique of size three.

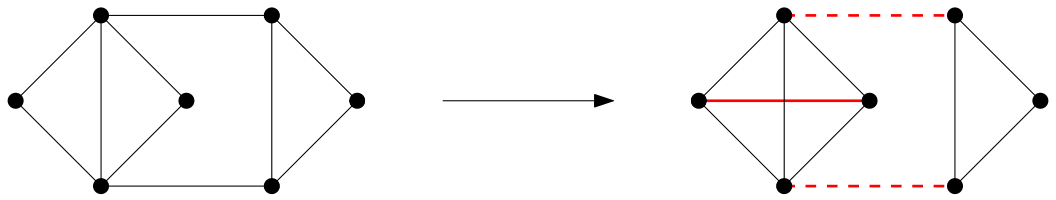

Figure 2.4: A CLUSTER EDITING instance and the resulting graph after applying a solution.

The problem of CLUSTER EDITING is equivalent to the problem of eliminating all induced P_3s of a graph [BFG+22]. If there is an induced P_3 in a CLUSTER EDITING instance, we can resolve it by exactly one edge-modification: We either add the non-edge or delete one of the two edges. P_3s can overlap, so one edge-modification might eliminate multiple P_3s. Therefore, the number of induced, vertex pair disjoint P_3s, gives us a lower bound for the number of edge-modifications of a given CLUSTER EDITING instance. We will refer to a set of induced, vertex pair disjoint P_3s as a *P_3-packing* or *packing*, for short. MAXIMUM INDEPENDENT SET is the problem of finding an independent set of maximum size in a given arbitrary graph.
In the following chapters, we will introduce and examine structures on which we can solve known problems like MAXIMUM INDEPENDENT SET in order to get lower bounds for CLUSTER EDITING through packings.

3. P_3-intersection graphs

The first section of this chapter covers the definition of a graph that represents the vertex pair sharing behaviour of all induced P_3s of a Cluster Editing instance, the P_3-intersection graph (see Section 3.1). As explained in Section 2.2, the size of a packing is a lower bound for the number of edge modifications needed to solve the Cluster Editing instance. Since a P_3 has two edges, there can be up to $|E|/2$ edge-disjoint induced P_3s in a graph $G = (V, E)$. Therefore, we can get lower bounds up to $|E|/2$ through this method. Before we examine the properties of P_3-intersection graphs in Section 3.3, we first look at some noteworthy examples in Section 3.2. As we gain a better understanding of P_3-intersection graphs, we see that there are certain properties they share with their origin graph. We cover these invariants in Section 3.4. While the focus of this work is not on the computation of P_3-intersection graphs, Section 3.5 still briefly outlines a way to compute them in polynomial time. At last, in Section 3.6, we look at the complexity of Maximum Independent Set on P_3-intersection graphs.

3.1 Defining P_3-intersection graphs

For a graph G we call $I(G)$ its P_3-intersection graph. The vertices of $I(G)$ correspond to the induced P_3s of G. We want two vertices to be adjacent if their corresponding P_3s share a vertex pair. By *share a vertex pair*, we mean that the P_3s have two vertices in common and therefore share an edge *or* non-edge. We call G an *origin graph* of $I(G)$. See Definition 3.1 for a formal definition and Figure 3.1 for an example.

Definition 3.1. *For a graph G, we define the* ***P_3-intersection graph*** $I(G) := (\tilde{V}, \tilde{E})$ *with* $\tilde{V} := \{\{a, b, c\} \mid a, b, c \in V \wedge ab, bc \in E \wedge ac \notin E\}$ *and* $\tilde{E} := \{\alpha\beta \mid \alpha, \beta \in \tilde{V} \wedge \alpha \neq \beta \wedge |\alpha \cap \beta| = 2\}$. *We will denote a vertex* $\{a, b, c\} \in \tilde{V}$ *with* $ac \notin E$ *as* (a, b, c). *Note that* $\tilde{V}$ *corresponds exactly to the set of induced P_3s of G.*

When talking about graphs and their P_3-intersection graph we often need to refer to a P_3 corresponding to a specific vertex η in the P_3-intersection graph. To this end, we define $\tilde{I}(\eta)$.

Definition 3.2. *The* ***P_3 corresponding to a vertex*** η ***of a P_3-intersection graph*** *is denoted by* $\tilde{I}(\eta)$.

Given a vertex η of a P_3-intersection graph with $\tilde{I}(\eta) = (a, b, c)$, there can be multiple P_3s sharing a particular vertex pair with (a, b, c). Without loss of generality, let ab be the shared vertex pair. Since the P_3s sharing ab with (a, b, c) contain a and b as well, they also share ab with each other. Therefore, they form a clique in the P_3-intersection graph, as we will show in Lemma 3.4. We will refer to such a clique as a *sharing clique*.

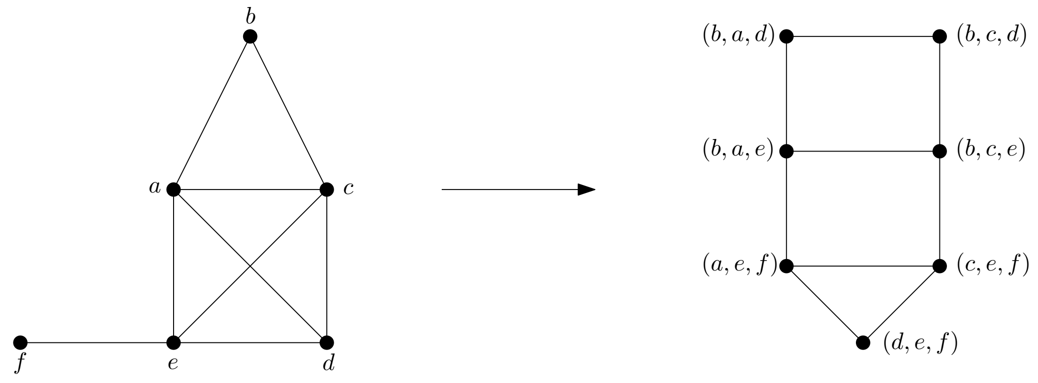

Figure 3.1: An example graph and its P$_3$-intersection graph.

Definition 3.3. *Let $G = (V, E)$ be a graph, $uv \in \binom{V}{2}$ and η a vertex of $I(G) = (\tilde{V}, \tilde{E})$. Then, $N_{uv}(\eta) := \{\rho \in \tilde{V} \mid \rho \cap \eta = \{u, v\}\}$ is called a* ***sharing clique*** *of η.*

Lemma 3.4. *Let η be a vertex of a P_3-intersection graph $I(G)$ with a graph G and a vertex pair uv of $\tilde{I}(\eta)$. $N_{uv}(\eta)$ is a clique in $I(G)$.*

Proof. Let η be a vertex of a P$_3$-intersection graph $I(G) = (\tilde{V}, \tilde{E})$ with a graph G and a vertex pair uv of $\tilde{I}(\eta)$. By definition, every $\rho \in N_{uv}(\eta)$ contains the vertices u and v as well. Therefore, $\binom{N_{uv}(\eta)}{2} \subseteq \tilde{E}$. $\square$

3.2 Noteworthy P$_3$-intersection graphs

To get a better understanding of P$_3$-intersection graphs, we will first look at the P$_3$-intersection graphs of some common graphs. See Figure 3.2 for a visual display.
A P$_3$-intersection graph of a complete graph has no vertices, since it has no non-edges. A path can be interpreted as a sequence of P$_3$s, each overlapping an edge with its predecessor and successor. Therefore, $I(P_{n+2}) = P_n$. A circle can be interpreted similarly: It can be seen as a circular sequence of overlapping P$_3$s. Therefore, at least for circles C_n with $n > 4$, its P$_3$-intersection graph is a circle of the same size. C_4 and C_3 are special cases: In C_4, each P$_3$ also shares its non-edge with the P$_3$ on the opposite side, implying $I(C_4) = K_4$. C_3 is equivalent to K_3, so $I(C_3) = I(K_3) = \emptyset$.
Every two edges of a n-star form a P$_3$. Thus, the 2-element subsets of the n-element set of edges are the vertices of the P$_3$-intersection graph and two of them are adjacent if the intersection of both 2-element subsets has one element. This is exactly the definition of the Johnson graph $J(n, 2)$ [Ter86].
It is also possible to get an arbitrary K_n as a P$_3$-intersection graph. In fact, there are multiple constructions, implying that there are multiple graphs G to a P$_3$-intersection graph $I(G)$.

Lemma 3.5. *Two distinct graphs G and G' can have the same P_3-intersection graph.*

Proof. Let G be a K_n where one arbitrary edge is removed and let G' be a K_{n-1} where an additional vertex v is added along with an edge uv for an arbitrary vertex u of the K_{n-1} with $u \neq v$. See Figure 3.2 for an example of both constructions with $n = 5$.
Both graphs contain $n - 2$ P$_3$s which all share one vertex pair. In G its the non-edge, in G' the edge uv. Therefore, $K_{n-2} = I(G) = I(G')$ is the P$_3$-intersection graph of G and G'. $\square$

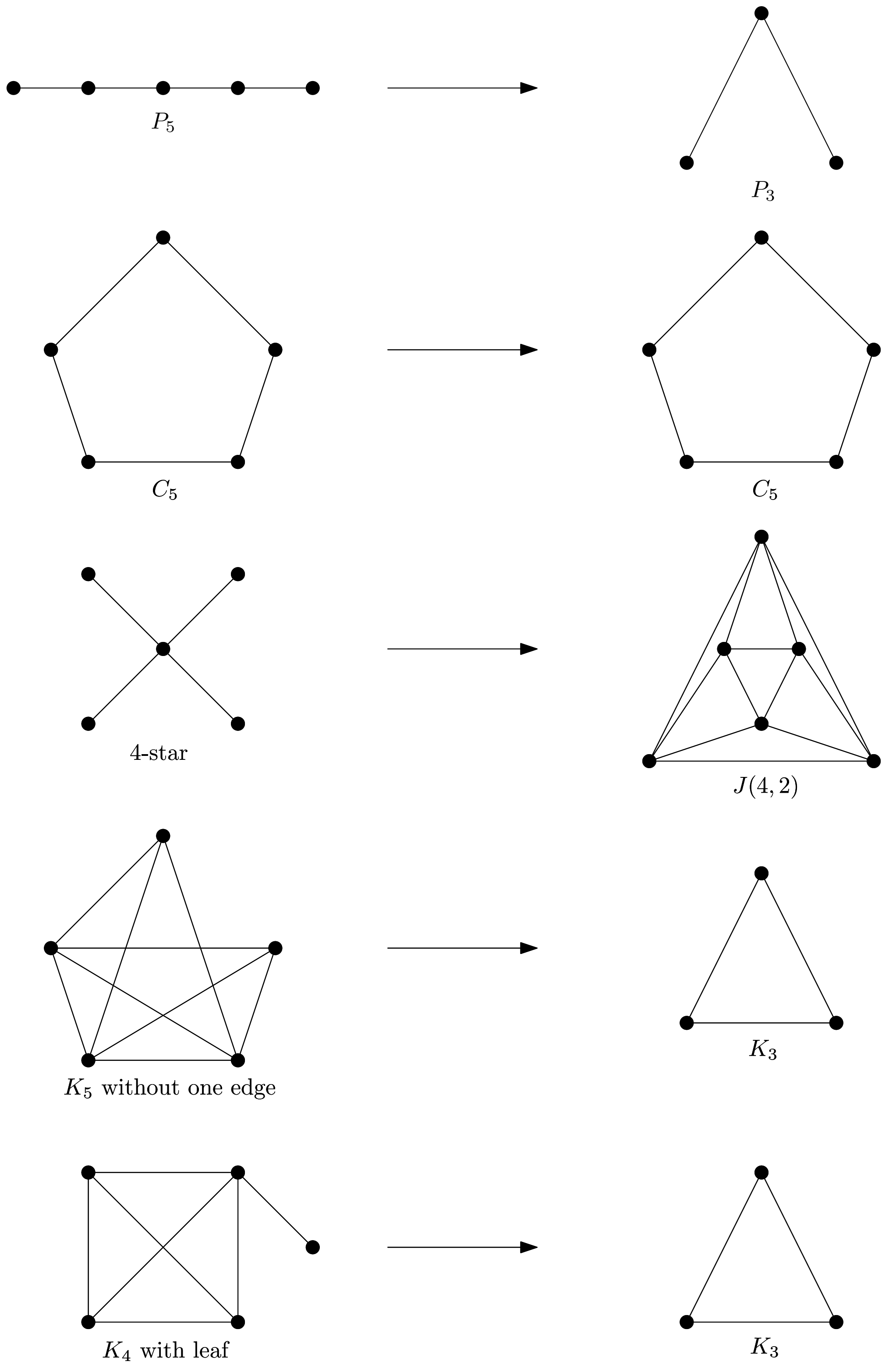

Figure 3.2: Examples of graphs and their corresponding P$_3$-intersection graph.

3.3 Properties of $\mathbf{P_3}$-intersection graphs

The P_3-intersection graph $I(G)$ of a graph G essentially models how the P_3s of G share vertex pairs. To learn about structural properties of P_3-intersection graphs, we first need to understand in which ways two P_3s can share vertex pairs and what that means for the neighbourhood of vertices in the P_3-intersection graph.

3.3.1 The neighbourhood of vertices in $\mathbf{P_3}$-intersection graphs

A P_3 consists of three vertices. Thus, there are $\binom{3}{2} = 3$ vertex pairs. Naturally, two P_3s do not share any vertex pairs if they are vertex-disjoint and are identical if they share all three. If they only share two vertex pairs, they already have three vertices in common, making them identical again. So assuming two P_3s are distinct, they can share at most one vertex pair.

Lemma 3.6. *Two distinct induced P_3s can share at most one vertex pair.*

Proof. Let a and b with $a \neq b$ be two induced P_3s of a graph. a and b share a vertex pair, if two vertices of a and b are identical. In order two share another vertex pair, one additional vertex needs to be identical. Then, all three vertices of the P_3s would be equal which contradicts $a \neq b$. $\square$

Now we know that every two adjacent vertices in a P_3-intersection graph $I(G)$ correspond to two P_3s in G which share exactly one vertex pair.

Definition 3.7. *Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the P_3-intersection graph of a graph $G = (V, E)$ and $\alpha, \beta \in \tilde{V}$ with $\alpha\beta \in \tilde{E}$. Then, the* ***shared vertex pair*** *$s(\alpha, \beta) := \alpha \cap \beta$ is the vertex pair shared by the P_3s corresponding to α and β.*

For every vertex v we can therefore characterize an adjacent vertex by the vertex pair it shares with v. This allows us to partition the neighbourhood of v into up to three sets, since v has three vertex pairs to share.

Theorem 3.8. *The neighbourhood of a vertex in a P_3-intersection graph can be partitioned into at most three non-empty pairwise independent sets.*

Proof. Let $\tilde{G}$ be a P_3-intersection graph, η an arbitrary vertex of $\tilde{G}$ and ρ an arbitrary vertex in $N(\eta)$. The corresponding P_3 $\tilde{I}(\rho)$ has to share a vertex pair with $\tilde{I}(\eta)$. According to Lemma 3.4, every P_3 sharing that edge with $\tilde{I}(\eta)$ is in the same set, in order to comply with the pairwise independence.
Thus, there can only be one such independent set per vertex pair of $\tilde{I}(\eta)$, which makes a total maximum of three non-empty pairwise independent sets. $\square$

It becomes apparent that the neighbours of a vertex in a P_3-intersection graph form cliques. In fact, there are only three cliques needed to cover the whole neighbourhood of every vertex in a P_3-intersection graph, which we will shown in Theorem 3.9.

Theorem 3.9. *The neighbourhood of a vertex in a P_3-intersection graph can be covered by at most three cliques.*

Proof. Let $\tilde{G}$ be a P$_3$-intersection graph and η an arbitrary vertex of $\tilde{G}$ with $\tilde{I}(\eta) = (u, v, w)$. $N_{uv}(\eta)$ is the set of P$_3$s sharing uv with $\tilde{I}(\eta)$. According to Lemma 3.4, that is a clique. The same applies for the sets $N_{vw}(\eta)$ and $N_{uw}(\eta)$.
Since $N(\eta) = N_{uv}(\eta) \cup N_{vw}(\eta) \cup N_{uw}(\eta)$, these three cliques cover the whole neighbourhood. □

The proof of Theorem 3.9 already shows a feasible choice for a clique cover of the neighbourhood of any P$_3$-intersection graph vertex consisting of three well-defined cliques. We define this formally in the following:

Definition 3.10. *Let η be a vertex of a P_3-intersection graph with $\tilde{I}(\eta) = (a, b, c)$. Then, the **neighbourhood clique partition** $NCP(\eta) \coloneqq \{N_{ab}(\eta), N_{bc}(\eta), N_{ac}(\eta)\}$ is the partition of $N(\eta)$ in the three cliques specified by which vertex pair their P_3s share with $\tilde{I}(\eta)$.*

We now consider the edges within the neighbourhood of a P$_3$-intersection graph vertex η. Let $\tilde{I}(\eta) = (a, b, c)$. Each edge $\alpha\beta$ with $\alpha, \beta \in N(\eta)$ is either inside one of the sharing cliques of $NCP(\eta)$ or between two of those cliques. We want to look at the edges between two cliques so assume α and β are in different sharing cliques of η. Since $\alpha, \beta \in N(\eta)$, they both have exactly two vertices in common with η, meaning that they also have exactly one vertex apart from (a, b, c). Let this vertex be $d(\alpha)$ and $d(\beta)$ respectively. Now since α and β are in different cliques and thus share different vertex pairs with η, $d(\alpha) \neq d(\beta)$ implies that $|\alpha \cap \beta| = 1$, meaning α and β are not adjacent. Hence, $d(\alpha) = d(\beta)$. Therefore, we can enumerate all possibilities for how a vertex d might be connected to a,b and c, to exhaustively define all inter-clique edges.
These possibilities are shown in Figure 3.3. The symmetrical cases are left out for simplicity. In the following, we will look at each of them in detail.

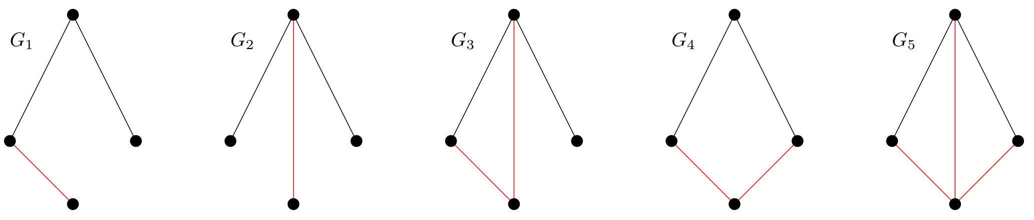

Figure 3.3: A P$_3$ with an additional vertex and all possibilities to connect that vertex to it. The connections of the additional vertex are marked in red.
1

The addition of the edge in G_1 lead to the creation of exactly one P$_3$, as seen in Figure 3.4. Therefore, this P$_3$ will not have an edge to another sharing clique of (a, b, c) in the P$_3$-intersection graph.

In G_2, two P$_3$s are created through the addition of one edge (see Figure 3.5). They both share the edge bd, resulting in an edge between the sharing cliques N_{ab} and N_{bc} in the P$_3$-intersection graph.
We see in Figure 3.6 that G_3 is similar to G_2 but the P$_3$ (a, b, d) is omitted due to the edge ad. Therefore, (d, b, c) will not have an edge to another sharing clique of (a, b, c).

G_4 has three new P$_3$s, which are added through the addition of the edges ad and cd. This can be seen in Figure 3.7. There is a P$_3$ for every vertex pair of (a, b, c) and any two of these P$_3$s share a vertex pair leading to a triangle between the three sharing cliques of (a, b, c).
G_5 (Figure 3.8), is G_4 with an added edge, preventing the creation of (d, a, b) and (b, c, d).

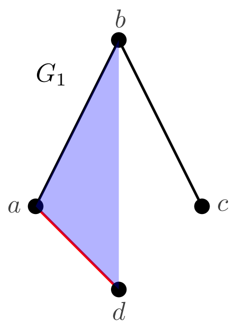

Figure 3.4: A P_3 (a, b, c) with an additional vertex d connected with a, which creates one additional P_3 highlighted in blue.

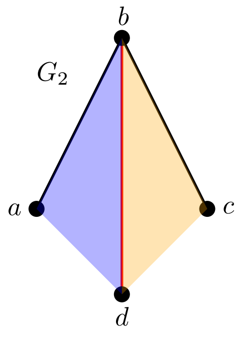

Figure 3.5: A P_3 (a, b, c) with an additional vertex d connected with b, which creates two additional P_3s highlighted in blue and yellow.

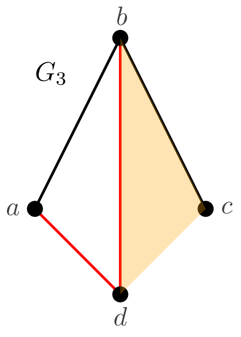

Figure 3.6: A P_3 (a, b, c) with an additional vertex d connected with a and b, which creates one additional P_3 highlighted in yellow.

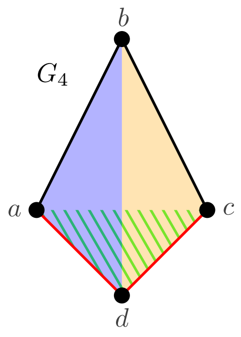

Figure 3.7: A P_3 (a, b, c) with an additional vertex d connected with a and c, which creates three additional P_3s highlighted in blue, yellow and green.

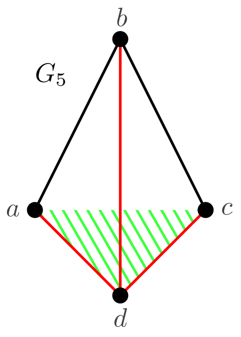

Figure 3.8: A P_3 (a, b, c) with an additional vertex d connected with a, b and c, which creates one additional P_3 highlighted in green.

So this time, only (a, d, c) exists. Therefore, the sharing clique N_{ac} has no edges to the sharing clique N_{ab} and N_{bc} similar to what we saw in G_1 and G_3.

We notice that there is a small number of possible constellations for the sharing cliques of a vertex in a P_3-intersection graph and the connections between them. Hence, we gain a reasonable understanding of the structure of its neighbourhood which we will formalize in

Theorem 3.11. A structural visualization of the neighbourhood of a P$_3$-intersection graph vertex can be seen Figure 3.9.

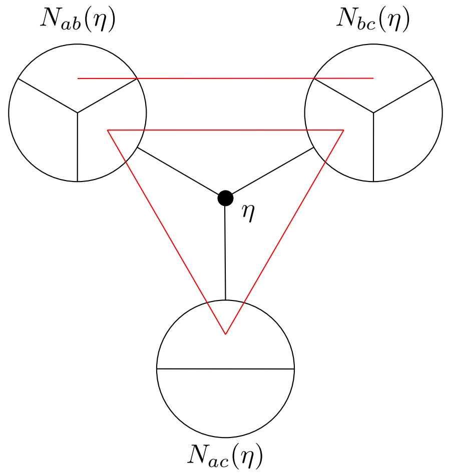

Figure 3.9: Depiction of the neighbourhood of a vertex η in a P$_3$-intersection graph with $\tilde{I}(\eta) = (a, b, c)$. All possible edges between neighbourhood cliques are either between $N_{ab}(\eta)$ and N_{bc} or part of a triangle between all three of them. Both cases are highlighted in red.

Theorem 3.11. *Let η be a vertex of a P$_3$-intersection graph with $\tilde{I}(\eta) = (a, b, c)$. Then, given the partition NCP(η), every edge between two of the cliques is either between $N_{ab}(\eta)$ and $N_{bc}(\eta)$ or part of a triangle between all three cliques.*
In both cases, the involved vertices do not have any other edges to one of the other cliques.

Proof. Let η be a vertex of a P$_3$-intersection graph $I(G)$ of a graph G with $\tilde{I}(\eta) = (a, b, c)$. Let $\alpha \in N_{ab}(\eta)$, $\beta \in N_{bc}(\eta)$ and $\gamma \in N_{ac}(\eta)$.
Note that α can either be $(\cdot, a, b)$ or $(a, b, \cdot)$, β can either be $(\cdot, b, c)$ or $(b, c, \cdot)$ and γ can only be $(a, \cdot, c)$. With $\cdot$ being a place holder for another vertex. Since adjacent vertices in a P$_3$-intersection graph require two shared vertices in their corresponding P$_3$s, edges between any two of α, β and γ can only exist, if both have the same vertex inserted for $\cdot$.

Assume $\alpha\beta$ exists. Then there is a vertex $d \in \alpha \cap \beta$. If $\alpha = (d, a, b)$, the edge db does not exist. Therefore, (d, b, c) is not a P$_3$ and $\beta = (b, c, d)$. Now, ad and dc are edges and ac is a non-edge. Hence, the P$_3$ $(a, d, c) \in N_{ac}$ exists. Implying that $\gamma = (a, d, c)$, $\alpha\gamma$ and $\beta\gamma$ have to be edges in $I(G)$.
If $\alpha = (a, b, d)$, the edge bd exists. Therefore, $(b, c, d) \neq \beta = (d, b, c)$. In this case, the P$_3$ (a, d, c) does not exist, since ad and dc are non-edges. So there are no edges from α or β to a vertex of $N_{ac}(\eta)$.

Assume $\alpha\gamma$ exists. Then there is a vertex $d \in \alpha \cap \gamma$. Since $\gamma = (a, d, c)$, ad has to be an edge. Therefore $(a, b, d) \neq \alpha = (d, a, b)$. This time, bc and ab are edges and bd is a non-edge. Hence, $\beta = (b, c, d)$ leading to the existence of $\alpha\beta$ and $\beta\gamma$ in $I(G)$.

Assume $\beta\gamma$ exists. Analog to the case above, there is a vertex $d \in \beta \cap \gamma$. Once again, $\gamma = (a, d, c)$ and thus $(a, b, d) \neq \alpha = (d, a, b)$. Here, da and ab are edges and bd is a non-edge. Therefore, $\alpha = (d, a, b)$ and $\alpha\beta$ as well as $\alpha\gamma$ are edges in $I(G)$. □

3.3.2 4-star-free graphs

Theorem 3.8 leads to an important characteristic of P$_3$-intersection graphs: Their vertices cannot have four or more neighbours which are pairwise non-adjacent. In other words, P$_3$-intersection graphs do not induce 4-stars.

Theorem 3.12. *P_3-intersection graphs are 4-star-free.*

Proof. Let $\tilde{G}$ be a P_3-intersection graph and S an induced 4-star in $\tilde{G}$ with root s_0 and leaves s_1, s_2, s_3 and s_4. By definition, the leaves are pairwise independent. Then $\{\{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{N(s_0) \setminus S\}\}$ is a partition of the neighbourhood of s_0 containing more than three non-empty pairwise independent sets. That contradicts Theorem 3.8. □

Finding a maximum weight independent set on arbitrary 4-star-free graphs is NP-complete [Min80]. However, P_3-intersection graphs cannot be arbitrary 4-star-free graphs. The graph shown in Figure 3.10 is an example of a graph that is 4-star-free but cannot be a P_3-intersection graph due to Theorem 3.11, because there is a vertex in C_2, which is adjacent to two vertices of the other cliques, which in turn are not adjacent.

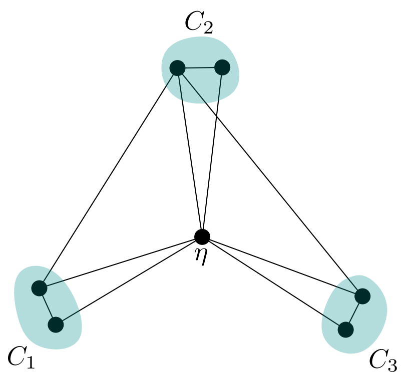

Figure 3.10: A graph with a vertex η, three cliques C_1, C_2, C_3 covering its neighbourhood and two inter-clique edges.

Theorem 3.13. *Not all 4-star-free graphs are P_3-intersection graphs.*

Proof. Let $\tilde{G}$ be the graph shown in Figure 3.10. $\tilde{G}$ is in the graph class of 4-star-free graphs. Assuming $\tilde{G}$ is a P_3-intersection graph, there is a clique cover of $N(\eta)$ with a maximum of three cliques according to Theorem 3.9. The only choice for such a clique cover is $\{C_1, C_2, C_3\}$, as shown in Figure 3.10. With $\tilde{I}(\eta) = (a, b, c)$, one of the three cliques has to be equal to N_{ac}. Then, there is a edge between N_{ac} and one of the other cliques which is not part of a triangle between all three cliques. That contradicts Theorem 3.11. It follows that $\tilde{G}$ cannot be a P_3-intersection graph. □

Corollary 3.14. *P_3-intersection graphs are a proper subset of 4-star-free graphs.*

If the class of P_3-intersection graphs did not contain 3-stars as well, there would be a polynomial-time algorithm for MAXIMUM INDEPENDENT SET on these graphs [Min80]. Even though the 3-star itself is not a P_3-intersection graph, it can be contained as an induced subgraph. See for example Figure 3.1.

3.3.3 3-mino graphs

In a 3-mino graph, every vertex is contained in at most three maximal cliques [MT03]. At first glance, this definition seems similar to the properties of P_3-intersection graphs given by theorems 3.8 and 3.9. However, the graph class of 3-mino graphs does not contain P_3-intersection graphs as made evident by the example in Figure 3.11: The vertex in the center of the P_3-intersection graph $I(G)$ of G is contained in four maximal cliques of $I(G)$.

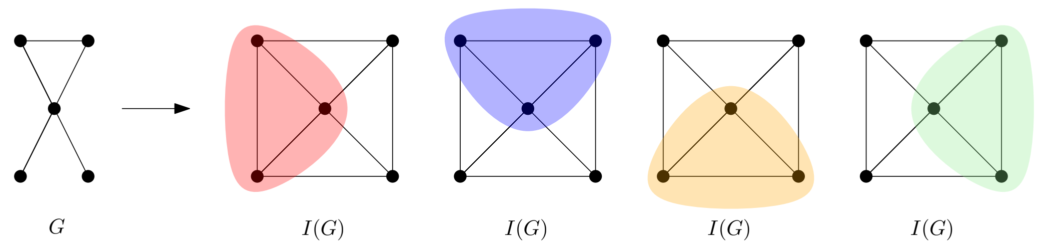

Figure 3.11: A graph G with its P$_3$-intersection graph $I(G)$ and four highlighted maximal cliques of $I(G)$

3.4 Invariants of P$_3$-intersection graph construction

When looking at graphs and their P$_3$-intersection graph it can be observed that a given induced subgraph always behaves the same way. For example in Figure 3.12, we can see that the induced C_5 is preserved and the induced P_5 turns into a P_3, so $I(C_5)$ and $I(P_5)$ are both induced subgraphs in the P$_3$-intersection graph of G. This applies for any induced subgraph in arbitrary graphs, as we show in Theorem 3.15.

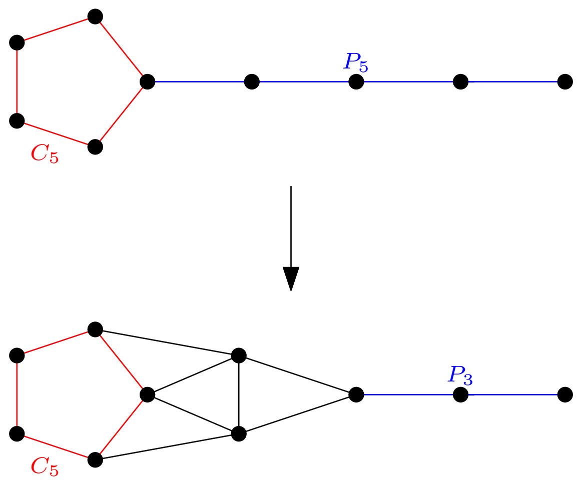

Figure 3.12: A graph G with its P$_3$-intersection graph $I(G)$ below it. The red highlighted C_5 also exists in $I(G)$ and the blue highlighted P_5 turns into a P_3.

Theorem 3.15. *If a graph G contains a graph H as an induced subgraph, the P$_3$-intersection graph of G contains the P$_3$-intersection graph of H as an induced subgraph.*

Proof. We will show that if we construct a graph H' by adding vertices and edges to a graph H in such a way that H' contains H as an induced subgraph, the following properties hold:

1. Every P$_3$ of H is preserved in H'.
2. Two P$_3$s sharing a vertex pair in H, share a vertex pair in H' as well.
3. Two P$_3$s not sharing a vertex pair in H, do not share a vertex pair in H'.

The only way to remove a P$_3$ by adding edges and vertices is by adding the non-edge. Since H' contains H as an induced subgraph, there are no edges added between already existing vertices. Therefore, the first and third property apply. The second property follows directly from the first one.

H' is an arbitrary graph which contains H as an induced subgraph. Since the vertices of $I(H')$ are exactly the P_3s of H' and the edges model the sharing of vertex pairs, it follows from the properties above that $I(H')$ contains $I(H)$ as an induced subgraph. $\square$

By Theorem 3.15 we are now able to predict what happens to subgraphs of a graph in its P_3-intersection graph, if we know the P_3-intersection graph of the subgraph beforehand. Next, we will consider single vertices of a graph and how they are represented in the P_3-intersection graph. More precisely, *if* they have a representation in the P_3-intersection graph. By definition they do, if they are part of an induced P_3. Since a clique has no non-edges between its vertices and no outgoing edges if it is isolated, the vertices of an isolated clique cannot have a representation in a P_3-intersection graph. In fact, this is the only case in which a vertex is not represented in the P_3-intersection graph, as shown in Lemma 3.16.

Lemma 3.16. *Every vertex of a graph G is either contained in an isolated clique or in an induced P_3.*

Proof. Let v be an arbitrary vertex v of a graph G.

Assume v is not in an isolated clique. Then, if v forms a clique with $N(v)$, there has to be another vertex u which is adjacent to a vertex w of the clique. Without loss of generality let $v \neq w$. Therefore, (u, w, v) is a P_3. If v does not form a clique with $N(v)$, there are $u, w \in N(v)$ with the non-edge uw. So v is again contained in a P_3 (u, v, w).

Now assume that v is not contained in an induced P_3. Then, there cannot be two vertices $u, w \in N(v)$ with the non-edge uw or else v would be contained in the P_3 (u, v, w). So $N(v) \cup \{v\}$ is a clique. There cannot be a vertex u adjacent to a vertex $w \neq v$ of the clique but not part of it, because then there would be the P_3 (u, w, v). Thus, the clique is isolated. $\square$

Since vertices of isolated cliques have no representation in the P_3-intersection graph, they do not influence it either. So we can just ignore them when looking at properties of P_3-intersection graph construction. This especially implies that an origin graph of a P_3-intersection graph can be created by just using vertices that are represented in its P_3-vertices.

Corollary 3.17. *For each P_3-intersection graph $\tilde{G} = (\tilde{V}, \tilde{E})$, there is an origin graph of size at most* $3 * |\tilde{V}|$.

Since a cluster editing instance can have multiple connected components, we also explore what happens with the P_3-intersection graph of such a graph.

Theorem 3.18. *Let G be a graph without isolated cliques. The P_3-intersection graph of G is connected if and only if G is connected.*

Proof. Let $I(G)$ be the P_3-intersection graph of a graph G without isolated cliques.

We first show "$\Longrightarrow$":
Assume $I(G)$ is connected. Hence, the vertices of two P_3s which each share a vertex pair with another P_3 are pairwise connected. Therefore, all vertices in G corresponding to a path of *length two* in $I(G)$ are pairwise connected. By transitivity, all vertices in G corresponding to a path of *any length* in $I(G)$ are pairwise connected. Since $I(G)$ is

connected, all vertices of G, which are contained in a P_3, are pairwise connected.
Because of Lemma 3.16 this holds for all vertices of G since G has no isolated cliques.

Now we show "$\Longleftarrow$":
Assume G is connected. So there is a shortest path between each two vertices of G connecting them. This shortest path is always an induced subgraph because otherwise, there would exist a shorter path. A *chain of P_3s* is a sequence of P_3s, where each two consecutive P_3s share a vertex pair. Note, that this translates to a sequence of edges in the P_3-intersection graph. In particular, the first and last vertex of an induced path are connected by a chain of P_3s. So every vertex of a P_3 is connected to every vertex of any other P_3 through a chain of P_3s. Hence, each vertex in $I(G)$ is connected to any other vertex through a sequence of edges. □

Since Theorem 3.18 applies for arbitrary graphs it also applies for every subset of connected components of a graph. Therefore, the number of connected components of a graph and its P_3-intersection graph is the same.

Corollary 3.19. *Let G be a graph without isolated cliques. G and its P_3-intersection graph have the same number of connected components.*

3.5 Computation of P_3-intersection graphs

In order to benefit from the unique properties of P_3-intersection graphs, they first need to be computed. Since its vertices are the induced P_3s of the origin graph $G = (V, E)$, a list of all induced P_3s of G is required. A simple approach is to check for every three-sized subset of vertices of G, if it is an induced P_3. This can be done by checking the existence of exactly two edges and one non-edge between these three vertices, which is possible in constant time. Altogether, it takes $O(\binom{|V|}{3})$ time to compute the full list of induced P_3s of G.
In order to compute the edges, we can check for each pair of P_3s α, β if their corresponding vertices in the P_3-intersection graph are adjacent. A simple method is to compute $\alpha \cap \beta$ and check if it has size two. The intersection of two sets of fixed size can be computed in constant time. Thus, the edges can be computed in $O(\binom{p}{2}) = O(\frac{1}{2}(p-1)p)$ time, with p being the number of P_3s in G.
In the worst case, every three-sized subset of V is indeed a P_3, so $\binom{|V|}{3}$ is a upper bound for the number of P_3s. Therefore, the computation takes

$$\Theta\left(\binom{\binom{|V|}{3}}{2}\right) = \Theta(|V|^6)$$

time. Hence, it is possible to compute the P_3-intersection graph of an arbitrary graph in polynomial time.

3.6 Maximum Independent Set on P_3-intersection graphs

As mentioned in Section 2.2, we want to solve Maximum Independent Set on the P_3-intersection graph to get a packing. Maximum Independent Set is NP-hard on arbitrary graphs [Har82]. We tried to find whether this still holds with all the constraints and properties that come with P_3-intersection graphs. However, all attempted approaches failed. Nonetheless, we outline some promising attempts and why they failed in this section.

Reduction from Exact-3SAT

In order to show the NP-hardness of Maximum Independent Set on P_3-intersection graphs, we can reduce from a known NP-hard problem. One of the most commonly encountered NP-hard problems is certainly 3SAT, where we try to decide whether a set of clauses, with a maximum of three literals each, is satisfiable. It is one of Karp's 21 NP-complete problems [Kar72]. A slight variation of 3SAT called Exact-3SAT, where every clause has exactly three literals, fits better to our case. In this variant we only need to worry about clauses of length three. So we need to construct a graph G from an Exact-3SAT instance C in such a way that Maximum Independent Set on its P_3-intersection graph $I(G)$ gives us a solution for C.
Our approach was to encode each literal incidence in C as a P_3. So for each literal l in clause $c \in C$ we create the P_3 (p, x, c) with x being the variable and p the polarity of l. This way, there cannot be two literal incidences in an independent set which have the same variable but different polarities. However, this encoding comes with two problems: First, if a literal incidence (p, x, c) is in an independent set, no other literal incidence in c with the same polarity can be in the same independent set, so this encoding cannot work. Second, apart from the P_3s representing literal incidences, unintended P_3s are created. If there are, for example, the literal incidences (p, x_1, c) and (p, x_2, c), we also create the P_3s (x_1, p, x_2) and (x_1, c, x_2). Especially the second problem occurs frequently in P_3 constructions. A possible solution for this is to add weights. We want every intentional P_3 to get weight 1 and every unintentional one to get weight 0. Since the instances of the related problem Weighted Cluster Editing regularly have weighted vertex pairs, we need to choose vertex pair weights in a way, that results in the P_3s having the required weights. Supposing that works, we use maximum weight independent set on the now vertex-weighted P_3-intersection graph to get a packing. As a consequence we can at most prove the NP-hardness of Maximum Weight Independent Set on P_3-intersection graphs, using this strategy.

Reduction from the three-dimensional assignment problem

George J. Minty uses a reduction from the *three-dimensional assignment problem* to prove the NP-completeness of Maximum Independent Set on 4-star-free graphs [Min80]. Since P_3-intersection graphs are 4-star-free (see theorem 3.12), our next approach was to adapt the idea for a reduction from the three-dimensional assignment problem. This problem can be pictured as follows: There are three n-sized sets R, G and B and a weight function $w : R \times G \times B \to \mathbb{R}^+$. The goal is to find n triples $t_i = (r_i, g_i, b_i)$ with $r_i \in R$, $g_i \in G$ and $b_i \in B$, such that $\sum_{i=1}^{n} w(t_i)$ is maximal and every $x \in R \cup G \cup B$ is contained in exactly one triple [SW96].
With arbitrary 4-star-free graphs it is possible to create a vertex for each of the n^3 possible triples, with two vertices being adjacent if their corresponding triples have at least one common element $x \in R \cup G \cup B$. The weight of a vertex is exactly the weight of the corresponding triple. We interpret the sets R, G and B as the coordinates from 1 two n in three dimensions. The constructed graph can then be viewed as a n-sized three-dimensional grid, where every vertex is contained in a clique with all the vertices on the same plane in each dimension respectively. Hence, a maximum weight independent set of such a graph contains no two vertices with a common coordinate, as required.
At first glance, it might look promising to construct a graph that has a similar structure in its P_3-intersection graph: For each possible triple (r, g, b) we construct the equivalent P_3 with equal weight. We use weight 0 to deal with unwanted P_3s as mentioned earlier. Due to the properties of P_3-intersection graphs, its vertices are indeed contained in three maximal cliques respectively. But unfortunately they only form cliques with the vertices on the same axes, instead of the same planes, since they need two common vertices to be adjacent instead of one. So this construction does not translate to P_3-intersection graphs.

Reducing CLIQUE to 3-dimensional matching on 3-uniform hypergraphs

Consider a hypergraph with all vertex pairs of a graph G as its vertices. In a hypergraph, an edge can join more than two vertices. In this case we define three vertices to be adjacent, if their corresponding vertex pairs induce a P_3. So every edge has exactly three vertices, hence the resulting hypergraph is called 3-uniform.
A maximum 3-dimensional matching on a 3-uniform hypergraph is a maximum subset of its hyperedges, where every two hyperedges are pairwise vertex disjoint. Since the edges in our case represent the P_3s of G, a maximum 3-dimensional matching is a maximum subset of P_3s of G, where every two P_3s are pairwise vertex pair disjoint, which is a maximum packing of G and therefore an equivalent problem to MAXIMUM INDEPENDENT SET on the P_3-intersection graph of G.
As shown by Kleinberg and Tardos, 3-dimensional matching is NP-hard on arbitrary 3-uniform hypergraphs [KT06]. They use a reduction from 3SAT for the NP-hardness proof using gadgets like the one in figure 3.13 to encode the clauses. We will not go into the details of the proof and the structure of the gadgets but explain why we cannot create such gadgets by hypergraphs constructed in the way described above.

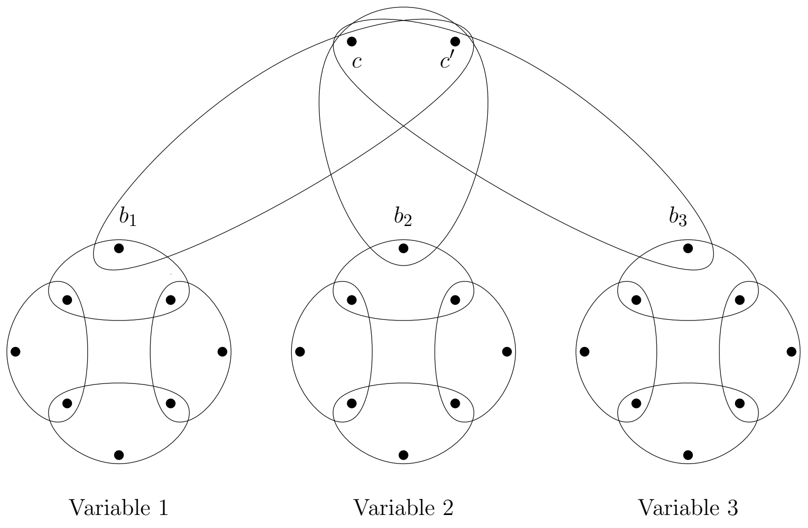

Figure 3.13: Gadget encoding a 3-SAT clause as a 3-uniform hypergraph.

Every variable incidence of a clause is represented by a hyperedge containing the clause core. In the example clause in figure 3.13, these are the vertices c and c'. To construct this, c and c' need to be two vertex pairs of G with a common vertex. Therefore, the third vertex pair of the hyperedge representing a P_3 is already dictated by c and c'. Hence, we cannot have three different hyperedges containing the clause core and b_1, b_2 or b_3 respectively to encode the up to three variable incidences of a clause.

4. Star-intersection graphs

As mentioned in section 2.2, the lower bounds obtained through P_3-intersection graphs cannot exceed $|E|/2$. Bläsius et al. [BFG+22] propose to find structures where the bound to edge ratio is better than for P_3s. They consider stars, which can be viewed as a generalization of P_3s, since a P_3 is just a 2-star. Solving Cluster Editing on a k-star needs $k-1$ edits. So we can obtain lower bounds up to $|E|-1$ by finding vertex pair disjoint stars. To this end, we want to generalize P_3-intersection graphs to star-intersection graphs.
We first define star-intersection graphs in section 4.1 by considering two possible definitions and discussing which one fits better to our use-case. Then, in 4.2, we compare star-intersection graphs to P_3-intersection graphs in terms of the lower bounds they can achieve for Cluster Editing instances as well as the time needed to compute the respective intersection graph.

4.1 Defining star-intersection graphs

Similar to P_3-intersection graphs, we want star-intersection graphs to model how induced stars in a graph share vertex pairs. Therefore, two vertices in a star-intersection graph are adjacent, if and only if their corresponding stars share a vertex pair. However, we have to consider how exactly we define the vertices of a star-intersection graph. The problem here is that for $k > 2$ a k-star always contains k induced $(k-1)$-stars. So we can either define the vertices of a star-intersection graph $I_S(G)$ to just be the stars that are not contained in a bigger star, called the *maximal stars*, or to be all induced stars of G. We will later refer to these as Definition I and II, respectively.

In both cases we want to be able to identify how big a star represented by a vertex in a star-intersection graph is. This can be achieved by introducing a weight function that maps the stars to the number of edits needed to resolve that star, in the sense of Cluster Editing. So every k-star has the weight $k-1$ in the star-intersection graph. Thus, we obtain a lower bound for a Cluster Editing instance by adding the weights of the vertices in a maximum weight independent set of the corresponding star-intersection graph. Since 1-stars have weight zero, we will ignore them from now on.

Now we can compare both definitions by the lower bounds they yield. In Figure 4.1, a maximum independent set of G', which is the star-intersection graph of G using Definition I, has a total weight of 2, whereas the P_3-intersection graph of G gives a lower bound of 3. G'' does also yield a lower bound of 3, suggesting that Definition II is superior to I.

Lemma 4.1. *The star-intersection graph of a graph G using all induced stars contains the star-intersection graph of G using only the maximal stars as an induced subgraph.*

Proof. Let G_I be the star-intersection graph of G using Definition I and G_{II} the star-intersection graph of G using Definition II. Since G_{II} includes all stars of G it in particular includes the ones included in G_I. Additionally, all edges and non-edges of G_I are also contained in G_{II}, because their definitions do not differ. □

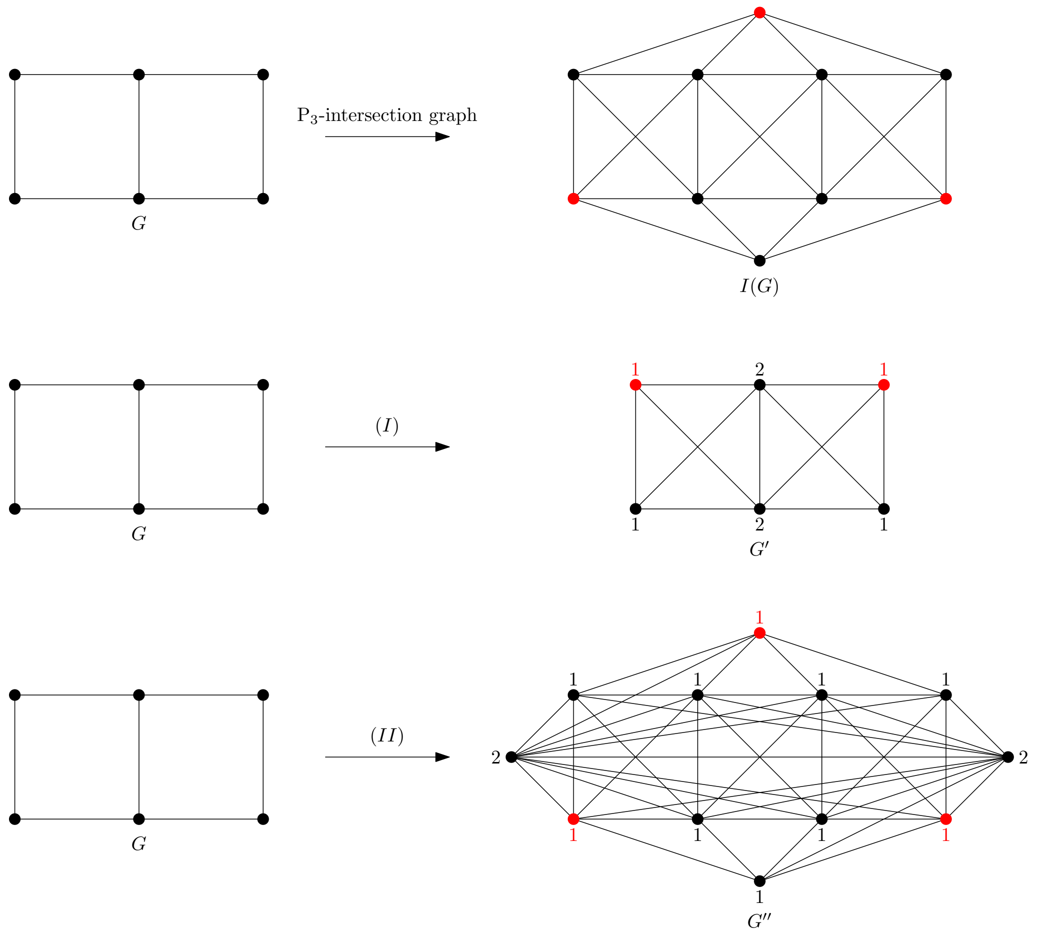

Figure 4.1: A graph G and its P$_3$-intersection graph $I(G)$ as well as its star-intersection graph in two variations G' and G''. Maximum (weight) independent sets of $I(G)$, G' and G'' are highlighted in red.

With Lemma 4.1 and the example in Figure 4.1, where G'' yields a better lower bound than G', we get Theorem 4.2.

Theorem 4.2. *The star-intersection graph of a graph G using all induced stars yields better lower bounds than the star-intersection graph of G using only the maximal stars.*

Proof. Let G_I be the star-intersection graph of an arbitrary graph G using Definition I and G_{II} the star-intersection graph of G using Definition II. Because of Lemma 4.1 G_{II} contains G_I as an induced subgraph. Hence, every maximum weight independent set of G_I is also an independent set in G_{II}. Therefore, the lower bound given by G_{II} is at least as good as the one given by G_I. With the example in Figure 4.4 we see that there are graphs for G, where G_{II} delivers better lower bounds than G_I. □

Naturally, star-intersection graphs using Definition I have far less vertices than the ones using Definition II, so they still could be useful in combination with other methods if their computation is reasonably fast. However, finding all maximal stars is actually NP-hard.

We show this by rephrasing this problem to a decision problem and proving that it is NP-complete (see Theorem 4.3): For a graph G and $k \in \mathbb{N}$, is there an induced k-star in G? We call this problem k-Star.

Theorem 4.3. k-Star *is NP-complete.*

Proof. For a given k-Star instance we can decide non-deterministically in polynomial time, whether a $(k+1)$-sized subset of vertices induces a k-star by assuming one of the vertices to be the center c and checking if every vertex, besides c, is adjacent to c but not adjacent to any other vertex. It follows that k-Star $\in \mathcal{NP}$.

To show that k-Star is NP-hard, we reduce from Clique, which is one of Karp's 21 NP-complete problems [Kar72]: For a graph G and a positive integer k, is there a k-clique in G?
Define the polynomial transformation f from a Clique instance to a k-Star instance as $f((V,E),k) = ((V',E'),k)$ with $V' = V \cup \{\hat{v}\}$ and $E' = \{\binom{V}{2} \setminus E\} \cup \{\{\hat{v}, v\} \mid v \in V\}$. (V',E') is essentially the complement graph of (V,E) with an additional vertex $\hat{v}$, which is adjacent to every vertex of V.
Let (G,k) be a Clique instance with an arbitrary graph $G = (V,E)$ and a positive integer k. $f(G,k) = (G',k)$ gives us the transformed graph $G' = (V',E')$.
If G has a k-clique, there is a subset $C \subseteq V$ of k pairwise adjacent vertices in G. That translates to a set of pairwise non-adjacent vertices in G'. So, C is an independent set of size k in G'. Since $\hat{v}$ is adjacent to every vertex of C, $S = C \cup \{\hat{v}\}$ is a k-star in G'.
If G' has a k-star, there is a subset $I \subseteq V'$ of k independent vertices. Since $\hat{v}$ is adjacent to every vertex of V, $\hat{v} \notin I$. Therefore, $I \subseteq V$. It follows that I is a set of pairwise adjacent vertices in G. Hence, G has a k-clique. $\square$

Now that we know which definition to use for the vertices of a star-intersection graph we can define them formally.

Definition 4.4. *For a graph $G = (V,E)$ we define the* ***star-intersection graph*** $I_S(G) := (V_S, E_S)$ *with* $V_S := \{S \subseteq V \mid S$ *is a k-star in G with* $k \geq 2\}$ *and* $E_S := \{\alpha\beta \mid \alpha, \beta \in V_S \ \wedge\ \alpha \neq \beta \ \wedge\ |\alpha \cap \beta| \geq 2\}$.

4.2 Star-intersection graphs in comparison to P_3-intersection graphs

Since star-intersection graphs are a generalization of P_3-intersection graph we want them to yield lower bounds, which are at least as good as the lower bounds of P_3-intersection graphs. We show that this holds, by proving that the star-intersection graph of a graph contains the P_3-intersection graph of the same graph as an induced subgraph in Lemma 4.5. In Theorem 4.6 we see, that there are examples for which star-intersection graphs give better lower bounds than P_3-intersection graphs, proposing that star-intersection graphs are in fact superior to P_3-intersection graphs in the lower bounds they provide.

Lemma 4.5. *The star-intersection graph of a graph G contains $I(G)$ as an induced subgraph.*

Proof. Let G_S be the star-intersection graph of an arbitrary graph G and $\tilde{G}$ the P_3-intersection graph of G. Since G_S includes all stars of G it particularly includes the 2-stars, also known as P_3s. Additionally, note that the definition of the edge set is effectively the same, since two different P_3s cannot share more than two vertices. Hence, all edges and non-edges of $\tilde{G}$ are also contained in G_S. $\square$

Theorem 4.6. *The star-intersection graph of a graph G yields better lower bounds than the P_3-intersection graph of G.*

Proof. Let G_S be the star-intersection graph of an arbitrary graph G and $\tilde{G}$ the P_3-intersection graph of G. Because of Lemma 4.5, G_S contains $\tilde{G}$ as an induced subgraph. Hence, every maximum weight independent set of $\tilde{G}$ is also an independent set in G_S. Therefore, the lower bound given by G_S is at least as good as the one delivered by $\tilde{G}$. Assuming G is a k-star with $k > 3$, $\tilde{G}$ delivers $\lfloor k/2 \rfloor$ while G_S gives an even better lower bound with $k-1$. $\square$

In Figure 4.1, the star-intersection graph labeled with G'' is rather complex in comparison with the other graphs in the figure. The reason for that is the sheer amount of stars in G. Recall that every k-star with $k > 2$ contains k induced $(k-1)$-stars. So in total that makes

$$\sum_{i=2}^{k} \binom{k}{i} = 2^k - k - 1$$

induced stars contained in a single k-star. Note that we ignore 1-stars, since they do not matter in our application. That means for an arbitrary graph G, with the size of the biggest induced star being $\hat{k}$, that the star-intersection graph has $\Omega(2^{\hat{k}})$ vertices. Therefore, the time needed to compute them or loop through its vertices can be exponential in $\hat{k}$ as well. Thus, it is infeasible to compute a maximum independent set on star-intersection graphs to obtain lower bounds for Cluster Editing.

5. $\mathrm{P_n}$-intersection graphs

Instead of only looking at paths of size three, we can also consider longer paths. In this section, we generalize some of the concepts and properties we examined for $\mathrm{P_3}$s to paths containing n vertices, with n being an arbitrary but fixed positive integer greater than one.

Similar to the $\mathrm{P_3}$-intersection graph, the vertices of a $\mathrm{P_n}$-intersection graph should be the induced $\mathrm{P_n}$s of the origin graph. We want two vertices to be adjacent, if their corresponding $\mathrm{P_n}$s share $n-1$ vertices. So they share exactly $\binom{n-1}{2}$ vertex pairs (see Lemma 5.5). That way, many properties and definitions concerning $\mathrm{P_3}$-intersection graph can easily be generalized for $\mathrm{P_n}$-intersection graph, as shown throughout this section. Formally, we define $\mathrm{P_n}$-intersection graph in definition 5.1. Note that for $n = 2$ the $\mathrm{P_n}$-intersection graph of a graph G is exactly the line graph of G (see [Moo63] for a definition) and for $n = 3$ it is exactly the $\mathrm{P_3}$-intersection graph.

Definition 5.1. *For a graph $G = (V, E)$ and $n \in \mathbb{N}$ we define the **P_n-intersection graph** $I_n(G) := (\tilde{V}, \tilde{E})$ with $\tilde{V} := \{\{v_1, \dots, v_n\} \mid v_i \in V \wedge v_iv_j \in E \iff |i-j| = 1 \wedge i, j \in [n]\}$ and $\tilde{E} := \{\alpha\beta \mid \alpha, \beta \in \tilde{V} \wedge \alpha \neq \beta \wedge |\alpha \cap \beta| = n-1\}$. We will denote a vertex $\{v_1, \dots, v_n\} \in \tilde{V}$ with $v_iv_j \notin E$ for $|i-j| > 1$ and $i, j \in [n]$ as $(v_1, \dots, v_n)$. Note that $\tilde{V}$ corresponds exactly to the set of induced P_ns of G.*

We also extend the definition of $\tilde{I}(\eta)$ to vertices η of $\mathrm{P_n}$-intersection graphs.

Definition 5.2. *The **P_n corresponding to a vertex** η of a P_n-intersection graph is denoted by $\tilde{I}(\eta)$.*

The concept of sharing cliques exists in $\mathrm{P_n}$-intersection graphs as well (see Definition 5.3 and Lemma 5.4). In contrast to $\mathrm{P_3}$-intersection graphs, the vertices of $\mathrm{P_n}$-intersection graphs can share more than two vertices. As mentioned previously, they do in fact share exactly $n-1$ vertices. Therefore, a sharing clique defines itself through the set of $n-1$ shared vertices.

Definition 5.3. *Let $G = (V, E)$ be a graph, $S \subset V$ and η a vertex of $I_n(G) = (\tilde{V}, \tilde{E})$. Then, $N_S(\eta) := \{\rho \in \tilde{V} \mid \rho \cap \eta = S\}$ is called a **sharing clique** of η.*

Lemma 5.4. *Let η be a P_n and S a set of vertex pairs of a graph G. $N_S(\eta)$ is a clique in $I_n(G)$.*

Proof. Let η be a $\mathrm{P_n}$ of a graph $G = (V, E)$ and $S \subset V$. Since every $\mathrm{P_n}$ corresponding to a vertex of $N_S(\eta)$ needs to contain all vertices of S, they also share S with each other. That makes them pairwise adjacent in $I_n(G)$. □

Lemma 5.5. *The P_ns corresponding to two adjacent vertices of a P_n-intersection graph share exactly $\frac{1}{2}(n-1)(n-2)$ vertex pairs.*

Proof. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be a $\mathrm{P_n}$-intersection graph and $\alpha\beta \in \tilde{E}$. By Definition 5.1, $\tilde{I}(\alpha)$ and $\tilde{I}(\beta)$ have $n-1$ common vertices. Therefore, they also share all vertex pairs between these $n-1$ vertices, which are exactly $\binom{n-1}{2} = \frac{1}{2}(n-1)(n-2)$. □

Since adjacent $\mathrm{P_n}$s share $n-1$ vertices, the neighbourhood of a $\mathrm{P_n}$-intersection graph vertex consists of $\binom{n}{n-1} = n$ sharing cliques. So again, similar to $\mathrm{P_3}$-intersection graphs, the neighbourhood of such a vertex can be partitioned into at most n independent sets (see Theorem 5.6) and covered by at most n cliques (see Theorem 5.7). Definition 5.8 formalizes a partition of the neighbourhood into n sharing cliques.

Theorem 5.6. *The neighbourhood of a vertex in a P_n-intersection graph can be partitioned into at most n non-empty pairwise independent sets.*

Proof. Let $\tilde{G}$ be a $\mathrm{P_n}$-intersection graph, η an arbitrary vertex of $\tilde{G}$ and ρ an arbitrary vertex in $N(\eta)$. The corresponding $\mathrm{P_n}$ $\tilde{I}(\rho)$ shares a set S of $n-1$ vertices with $\tilde{I}(\eta)$. According to Lemma 5.4, every $\mathrm{P_n}$ sharing that S with $\tilde{I}(\eta)$ is in the same set, in order to comply with the pairwise independence. Thus, there can only be one such independent set for each choice of the $n-1$ vertices. Since there are $\binom{|\eta|}{n-1} = \binom{n}{n-1} = n$ possible choices, the theorem follows. □

Theorem 5.7. *The neighbourhood of a vertex in a P_n-intersection graph can be covered by at most n cliques.*

Proof. Let $G = (V, E)$ be a graph and η an arbitrary vertex of its $\mathrm{P_n}$-intersection graph. According to Lemma 5.4, $N_S(\eta)$ is a clique for each $S \subset V$. Then,

$$N(\eta) = \bigcup_{S' \in \binom{\eta}{n-1}} N_{S'}(\eta)$$

is the union of n cliques that cover the whole neighbourhood. □

Definition 5.8. *Let $G = (V, E)$ be graph and η an arbitrary vertex of its P_n-intersection graph. Then, the* ***neighbourhood clique partition*** *$NCP(\eta) = \{N_{S'}(\eta) \mid S' \in \binom{\eta}{n-1}\}$ is the partition of $N(\eta)$ into n cliques, specified by which set of vertices their P_ns share with $\tilde{I}(\eta)$.*

We find that the absence of 4-stars in $\mathrm{P_3}$-intersection graphs translates to $\mathrm{P_n}$-intersection graphs as well. With a similar argument as in Theorem 3.12, $\mathrm{P_n}$-intersection graphs are in fact $(n+1)$-star-free.

Theorem 5.9. *P_n-intersection graphs are $(n+1)$-star-free.*

Proof. Let $\tilde{G}$ be a $\mathrm{P_n}$-intersection graph and S an induced $(n+1)$-star in $\tilde{G}$ with root s_0 and pairwise independent leaves $s_1, \ldots, s_{n+1}$. Then, $\{\{s_1\}, \ldots, \{s_{n+1}\}, \{N(s_0) \setminus S\}\}$ is a partition of the neighbourhood of s_0 containing more than n non-empty pairwise independent sets. This contradicts Theorem 5.6, implying that $\tilde{G}$ is in fact $(n+1)$-star-free. □

Lower Bounds with $\mathrm{P_n}$-intersection graphs

In the sense of CLUSTER EDITING, a $\mathrm{P_n}$ can be resolved with $\lfloor \frac{n-1}{2} \rfloor$ edge modifications by deleting every second edge. However, when considering a $\mathrm{P_n}$ as $n-2$ overlapping $\mathrm{P_3}$s, we can choose exactly $\lceil \frac{n-2}{2} \rceil = \lfloor \frac{n-1}{2} \rfloor$ vertex pair disjoint $\mathrm{P_3}$s. So, $\mathrm{P_3}$-intersection graph are capable of delivering equal lower bounds to $\mathrm{P_n}$-intersection graphs. The key difference is that an induced $\mathrm{P_n}$ requires $\binom{n}{2} - (n-1) = \frac{n(n-1)}{2} - (n-1)$ non-edges, while $n-2$ $\mathrm{P_3}$s, covering a $\mathrm{P_n}$, require only $n-2$ non-edges. Therefore, from the perspective of CLUSTER EDITING, the trade-off between the complexity, expressed through the edge requirements, and the delivered lower bounds of $\mathrm{P_n}$-intersection graphs is worse than with $\mathrm{P_3}$-intersection graphs.

6. Conclusion

This work introduced and examined three types of intersection graphs in order to find lower bounds for Cluster Editing through packings.
First, we examined P_3-intersection graphsand concluded that they form a proper subset of 4-star-free graphs. Additionally, the neighbourhood of every vertex η in such a graph can be partitioned into three cliques specified by which vertex pair the corresponding P_3 shares with the P_3 corresponding to η. Moreover, we showed that edges between these cliques only exist as part of a triangle spanning over all three cliques or between the two cliques specified by the vertex pairs that are the edges of the P_3 corresponding to η. We also found that the P_3-intersection graph of a graph contains the P_3-intersection graph of every induced subgraph as an induced subgraph. That means we may not always need to compute the whole P_3-intersection graph to gain knowledge of the local structure, even though we showed that this is possible in polynomial time.
Furthermore, we outlined some of our approaches to show the NP-hardness of Maximum Independent Set on P_3-intersection graphs, which would yield a P_3-packing. However, the strict structure of P_3s proved problematic to construct a polynomial transformation for all approached reductions. An approximation algorithm, on the other hand, could benefit from this strict structure, leading us to suspect that it is indeed possible to construct a good polynomial approximation algorithm or even a polynomial exact algorithm for Maximum Independent Set on P_3-intersection graphs.
We have already seen that multiple distinct graphs can have the same P_3-intersection graph. However, it could be possible to prove that the number of origin graphs to a given P_3-intersection graph, excepting such with isolated cliques, is bounded depending on, for example, the size of the given graph. Future work could also pursue the development of an algorithm which can check if a given graph is a P_3-intersection graph or which aims to find a valid origin graph to a given P_3-intersection graph.
We continued with the examination of other intersection graphs by taking the concept of P_3-intersection graphs and applying it to stars and paths of arbitrary size. We found that there actually is a definition for star-intersection graphs which is able to yield better lower bounds for Cluster Editing than P_3-intersection graphs. The downside however, is the, on the size of the biggest star dependent, exponential time and space needed to compute and store it. P_n-intersection graphs, on the other hand, yield equal lower bounds for Cluster Editing, but, since they come with more restrictions than P_3-intersection graphs, they are less suited to finding lower bounds for Cluster Editing than P_3-intersection graphs. Finding further structures and defining corresponding intersection graphs, where the trade-off between structure complexity and granted lower bound is optimized, could also be pursued in future work.

Bibliography

[BFG⁺21] Thomas Bläsius, Philipp Fischbeck, Lars Gottesbüren, Michael Hamann, Tobias Heuer, Jonas Spinner, Christopher Weyand, and Marcus Wilhelm. PACE Solver Description: KaPoCE: A Heuristic Cluster Editing Algorithm. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:4, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BFG⁺22] Thomas Bläsius, Philipp Fischbeck, Lars Gottesbüren, Michael Hamann, Tobias Heuer, Jonas Spinner, Christopher Weyand, and Marcus Wilhelm. A Branch-And-Bound Algorithm for Cluster Editing. In Christian Schulz and Bora Uçar, editors, *20th International Symposium on Experimental Algorithms (SEA 2022)*, volume 233 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Bö12] Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 16:79–89, 2012. Selected papers from the 22nd International Workshop on Combinatorial Algorithms (IWOCA 2011).

[CMSS17] Jiehua Chen, Hendrik Molter, Manuel Sorge, and Ondrej Suchy. Cluster editing for multi-layer and temporal graphs. *arXiv preprint arXiv:1709.09100*, 2017.

[Die05] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173:33, 2005.

[EET76] Gideon Ehrlich, Shimon Even, and Robert Endre Tarjan. Intersection graphs of curves in the plane. *Journal of Combinatorial Theory, Series B*, 21(1):8–20, 1976.

[Gav74] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[Gav78] Fănică Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Mathematics*, 23(3):211–227, 1978.

[GJ85] Martin Charles Golumbic and Robert E Jamison. Edge and vertex intersection of paths in a tree. *Discrete Mathematics*, 55(2):151–159, 1985.

[Gol88] Martin Charles Golumbic. Algorithmic aspects of intersection graphs and representation hypergraphs. *Graphs and Combinatorics*, 4(1):307–321, 1988.

[Har82] Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[KKNZ21] Leon Kellerhals, Tomohiro Koana, André Nichterlein, and Philipp Zschoche. The pace 2021 parameterized algorithms and computational experiments challenge: Cluster editing. In *16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[Kle69] Victor Klee. What are the intersection graphs of arcs in a circle? *The American Mathematical Monthly*, 76(7):810–813, 1969.

[KT06] Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

[LMNN21] Junjie Luo, Hendrik Molter, André Nichterlein, and Rolf Niedermeier. Parameterized dynamic cluster editing. *Algorithmica*, 83(1):1–44, 2021.

[Min80] George J Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980.

[Moo63] J. W. Moon. On the Line-Graph of the Complete Bigraph. *The Annals of Mathematical Statistics*, 34(2):664 – 667, 1963.

[MT03] Yury Metelsky and Regina Tyshkevich. Line graphs of helly hypergraphs. *SIAM Journal on Discrete Mathematics*, 16(3):438–448, 2003.

[MW86] Clyde L Monma and Victor K Wei. Intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 41(2):141–181, 1986.

[pac] About the Parameterized Algorithms and Computational Experiments Challenge. `https://pacechallenge.org/about/`. Accessed: 2022-08-12.

[PDdSS09] Fábio Protti, Maise Dantas da Silva, and Jayme Luiz Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 44(1):91–104, 2009.

[SST02] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 379–390. Springer, 2002.

[SW96] Frits C.R. Spieksma and Gerhard J. Woeginger. Geometric three-dimensional assignment problems. *European Journal of Operational Research*, 91(3):611–618, 1996.

[Ter86] Paul Terwilliger. The johnson graph j (d, r) is unique if (d, r)$\neq$(2, 8). *Discrete mathematics*, 58(2):175–189, 1986.

[Zah64] CT Zahn, Jr. Approximating symmetric relations by equivalence relations. *Journal of the Society for Industrial and Applied Mathematics*, 12(4):840–847, 1964.