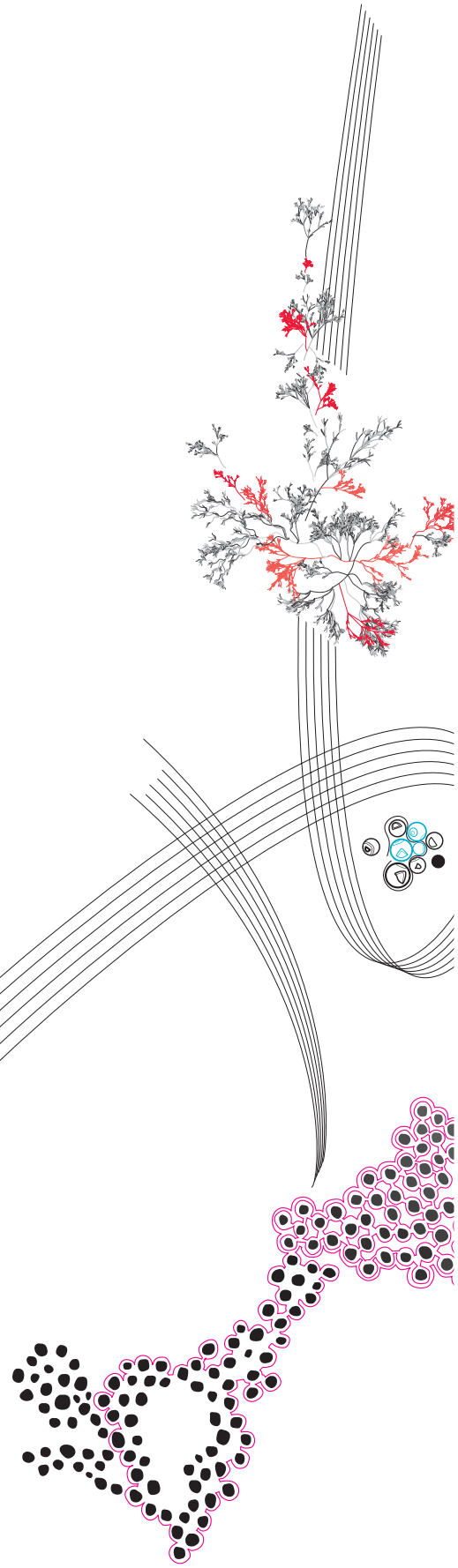

BSc Thesis Applied Mathematics

Counting and enumerating 2-isomorphic graphs

Patrick Heuven

Supervisor: Rolf van der Hulst and Matthias Walter

July, 2023

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Counting and enumerating 2-isomorphic graphs

Patrick Johan Bernard Heuven*

July, 2023

Abstract

This research aims to find every 2-isomorphic graph of a given 2-connected graph G. The main results of this research include a formula for the number of different 2-isomorphic graphs of G and how to enumerate every 2-isomorphic graph efficiently. These results are achieved by utilizing Whitney switches and Whitney's theorem, which provide a way to find any 2-isomorphic graph of G. A Whitney switch can only be applied on a pair of vertices that represent a vertex cut. The SPQR-decomposition is used to find these pairs of vertices. The SPQR-decomposition creates a SPQR-tree of a 2-connected graph and differentiates 2-connected and 3-connected parts of a graph. The pairs of vertices that represent a vertex cut are not found in 3-connected parts of a graph. This property is used to derive the number of different 2-isomorphic graphs based on the SPQR-tree of G. This derivation is also used to design an algorithm that enumerates every 2-isomorphic graph of G with time complexity $O(|V| + N|E|)$, where N is the number of different 2-isomorphic graphs.
Keywords: 2-isomorphic, graph, SPQR-tree, SPQR-decomposition

1 Introduction

This paper talks about 2-isomorphic graphs. 2-isomorphic graphs have applications in graph theory, where it is used to analyze the structure of graphs. Fields where 2-isomorphisms are used include electrical theory, as sketched in [6]. One of the research goals involves finding every 2-isomorphic graph of a given graph, which can possibly be used to enhance techniques and methods in these fields.
The concept of a 2-isomorphic graph is similar to graph isomorphism. However, the main difference between graph isomorphisms and 2-isomorphisms is that graph isomorphism is focused on the vertices and ensuring that adjacent vertices of a vertex can be the same in two graphs, while graph 2-isomorphism is focused on the edges and ensuring that the edges of every cycle in one graph form a cycle in the other graph and vice versa.

Hassler Whitney has found an important result that if two 2-connected graphs G and H are 2-isomorphic, then there is an operation called a Whitney switch that can be applied multiple times on G to create a graph that is 2-isomorphic to H [6]. The Whitney switch can only be applied to certain pairs of vertices in a graph. The SPQR-tree is a data structure that is going to be used to find every pair of vertices where a Whitney switch can be applied.

*Email: p.j.b.heuven@student.utwente.nl

This leads to the research questions of this paper:

1. How many different 2-isomorphic graphs does a given 2-connected graph G have?
2. How can all of these 2-isomorphic graphs be enumerated efficiently?

This paper starts of with definitions and important results from graph theory in Section 2. Then the definition of a 2-isomorphic graph is discussed in Section 3. Furthermore, a definition of different graphs is given in 3.2, which excludes graphs that are, for example, rotations of other graphs. In Section 4, the SPQR-tree is explained, which will be crucial to find every pair of vertices where a Whitney switch can be applied. Section 5 combines all the information to find the number of 2-isomorphic graphs of a given graph G. Lastly, Section 6 discusses an algorithm that enumerates all 2-isomorphic graphs and discusses the efficiency of that algorithm.

2 Preliminaries

Graphs. A graph G is a tuple $(V(G), E(G))$, where $V(G)$ contains vertices of the graph and $E(G)$ contains edges of the graph. An edge $e \in E(G)$ is generally defined as a tuple (u, v), where u and v are vertices of G. Note that a tuple is always ordered. This research will focus on undirected graphs. The order of the tuple of an edge in an undirected graph does not matter. This leads to an edge of an undirected graph being defined as a set of two vertices.
The graphs in this research are finite graphs with labeled edges, which implies that the number of vertices and edges is finite for every graph G. Multi-edges are allowed, which implies that there can be multiple edges between two vertices of the graph. Furthermore, every edge is labeled, which implies that a unique label is assigned to every edge. Lastly, it will be assumed that there are no loops in the graphs of this paper, since these loops do not influence the results.

A *path* in the graph G is a sequence of different vertices $v_1, v_2, ..., v_{n-1}, v_n$ in G such that there is an edge in G that contains both v_i and v_{i+1} for $i \in \{1, 2, ..., n-1\}$. A path that starts at v_1 and ends at v_n is called a (v_1, v_n)-path. A path that starts and ends at the same vertex is called a *cycle*.
A graph is *connected* if there is at least one path between pair of distinct vertices in the graph. This implies that a graph is disconnected whenever there are two vertices u and v in the graph such that there is no (u, v)-path.

Definition 2.1 (Vertex cut). A set of vertices of a connected graph represents a *vertex cut* if the graph is disconnected whenever that set of vertices and all edges incident with these vertices are removed.

The size of a vertex cut is the cardinality of the set of vertices that represent the vertex cut. A disconnected graph has a vertex cut of size 0, since no vertices need to be removed to disconnect the graph.

Definition 2.2 (k-connected). A connected graph is *k-connected* if the size of any vertex cut is at least k.

A 1-connected graph is the same as a connected graph. A 0-connected graph is the same as a disconnected graph. This research will mostly talk about 2-connected graphs.

Suppose that there are two paths between two vertices, then these paths are *internally disjoint* whenever the two paths share no vertices except the starting and the ending vertex of the path. This leads to the following known results:

Lemma 2.3. *[[1]] If a graph is k-connected, then there are at least k internally disjoint paths between any pair of distinct vertices of the graph.*

Corollary 2.4. *Every edge and every pair of vertices of a 2-connected graph is contained in at least one cycle.*

Proof. Suppose G is a 2-connected graph. This implies there are at least two internally disjoint paths between any pair of vertices in the graphs by Lemma 2.3. Thus for every pair of vertices u and v of G there are two internally disjoint (u, v)-paths, which together form a cycle that contain u and v. Furthermore, for every edge $\{x, y\}$ in G there is at least one other internally disjoint path between the vertices x and y, which implies that there is a cycle that contains the internally disjoint path from x to y and the edge $\{x, y\}$. □

A *subgraph* G' of a graph G is a graph such that $V(G') \subseteq V(G)$ and $\{u, v\} \in E(G')$ if $u, v \in E(G)$ and $u, v \in V(G')$.
A *component* of a graph G is a connected subgraph G' of G where for every $v \in V(G')$ there is no $u \in V(G)/V(G')$ such that $\{u, v\} \in E(G)$. In a connected graph, the number of components is equal to 1. However, in a disconnected graph the number of components is at least 2. This definition of a disconnected graph will also be used in this paper.

The components of a graph will be used in this paper whenever the vertices of a vertex cut are removed from a graph. This removal of the vertices u and v of a graph G is denoted as $G - \{u, v\}$. $E(G - \{u, v\})$ contains all edges of G that are not incident with u or v. $V(G - \{u, v\})$ contains every vertex of G except u and v.

3 2-isomorphic graphs

A 2-isomorphic graph G' of a graph G says something about the structure of the graphs, as mentioned in the introduction. 2-isomorphisms are based on the cycles of G and G', where a definition of preserving a cycle is necessary to clearly define what a 2-isomorphism is.

Definition 3.1 (Preserving Cycles). A cycle of graph G is preserved in graph H by a bijective mapping $\tau : E(G) \to E(H)$ when the edges of the cycle in G are mapped to H and form a cycle in H.

One should note that the order of the edges in the cycle does not matter when a cycle is preserved. This means that a cycle in G that is defined by the ordered list of edges $\{e_1, e_2, e_3, e_4\}$ could be preserved in H as the ordered list $\{\phi(e_1), \phi(e_3), \phi(e_2), \phi(e_4)\}$. In this case e_1 and e_3 are not adjacent in G, while $\phi(e_1)$ and $\phi(e_3)$ are adjacent in H. This will be used later in this section to define a cycle-preserving operation.

Definition 3.2 (2-isomorphic graphs). Two 2-connected graphs G and H are 2-isomorphic if and only if there exists a bijective mapping $\tau : E(G) \to E(H)$ such that τ preserves every cycle in G and its inverse τ^{-1} preserves every cycle in H.

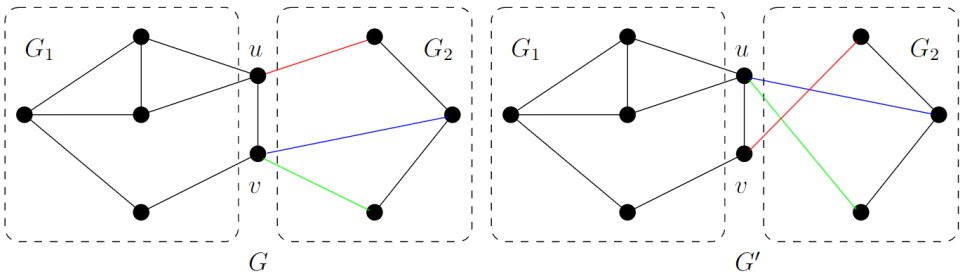

FIGURE 1: G' is a 2-isomorphic graph of G. G' can be made from G by applying one Whitney switch to G_2.

3.1 Whitney switch

The Whitney switch is an important operation that is going to be used to derive the main results of this paper. This operation is used in 2-connected graphs and can only be applied to a pair of vertices $\{u, v\}$ that represent a vertex cut. The most important property of the Whitney switch is that it is a cycle-preserving operation. A Whitney switch is applied on the vertices u and v in Figure 1.

Definition 3.3 (Whitney switch). Let G be a 2-connected graph and let u and v be vertices of G such that $\{u, v\}$ is a vertex cut of G. Let G_1 be a component of $G - \{u, v\}$. Applying a Whitney switch on G_1 in G results in a graph G', where $V(G') = V(G)$. Let $E_1(G, u, v)$ be all edges in G such that no edge in $E_1(G, u, v)$ is incident with u or v and let $E_2(G, u, v)$ be all edges in G such that every edge in $E_2(G, u, v)$ is incident with u or v. For every edge $\{u, x\}, \{v, y\}, \{u, v\}$ in $E_2(G, u, v)$, where $x, y \in V(G_1)$, there is an edge $\{v, x\}, \{u, y\}, \{u, v\}$ in $E_2(G', u, v)$. Let $E(G') = E_1(G, u, v) \cup E_2(G', u, v)$.

In this definition, the component G_1 of the graph $G - \{u, v\}$ will be referred to as *an adjacent component of the vertices* $\{u, v\}$ *that represent a vertex cut.* The definition of the Whitney switch gives the choice to which adjacent component the Whitney switch is applied.
Furthermore, the labels of the edges are important: Suppose $\{u, v\}$ is a vertex cut and edge $\{u, x\}$ is replaced with $\{v, x\}$ after a Whitney switch, then the label of the new edge $\{v, x\}$ should be the same as the label of the edge $\{u, x\}$.
A known result is that applying a Whitney switch on a 2-connected graph results in another 2-connected graph. Another known result is that the cycles of the original graph with labeled edges are preserved in the graph where a Whitney switch has been applied [3]. An important result from these properties is given by Whitney's theorem.

Theorem 3.4 (Whitney's Theorem [6]). *Any 2-isomorphic graph of a 2-connected graph G can be created by applying a sequence of Whitney switches to G.*

Whitney's theorem gives a possibility to find the number of 2-isomorphic graphs of a given 2-connected graph with labeled edges. This number can be found by considering every possible Whitney switch of the given graph and counting the number of different graphs resulting from applying all possible combinations of Whitney switches.

3.2 Equivalent graphs

Graphs can be drawn in multiple different ways while having the same edges and vertices. One could think of rotating a graph 90 degrees or mirroring a graph, where the graph might look different, but is essentially the same.

Definition 3.5. Let G and H be graphs with labeled edges. Let $\tau : E(G) \to E(H)$ be a bijective mapping such that for every edge e in $E(G)$ the label of $\tau(e)$ is the same as the label of e.
G and H are *equivalent* if and only if there is a bijective mapping $\phi : V(G) \to V(H)$ such that for every $v \in V(G)$ and every $e \in E(G)$: $v \in e \iff \phi(v) \in \tau(e)$.

The number of different 2-isomorphic graphs of a given graph G is defined as the same as the number of non-equivalent 2-isomorphic graphs of a given graph G. However, the number of non-equivalent 2-isomorphic graphs of a given graph G can be interpreted as every 2-isomorphic graph of G that is not equivalent to G. Thus a definition of different graphs is given to avoid confusion:
The graphs $H_1, H_2, ..., H_k$ are different graphs whenever H_i is not equivalent to H_j for every $i, j \in \{1, 2, ..., k\}$ and $i \neq j$.

The number of different 2-isomorphic graphs can be found by considering every Whitney switch of a graph. This first requires analysis of equivalent graphs when there is one pair of vertices that represents a vertex cut.

Theorem 3.6. *Suppose G is a 2-connected graph with labeled edges. Furthermore, suppose the set of vertices $\{u, v\}$ represents a vertex cut of G. Let the adjacent components of $\{u, v\}$ be given by $G_1, G_2, ..., G_n$.*
Consider the graph where an even amount of Whitney switches has been applied to every adjacent component $G_1, G_2, ..., G_n$ and consider the graph where an odd amount of Whitney switches has been applied to every adjacent component $G_1, G_2, ..., G_n$. These graphs are equivalent to G and all other graphs that are achieved by applying Whitney switches on the adjacent components, are not equivalent to G.

Proof. Let G be a 2-connected graph with labeled edges. Furthermore, let the set of vertices $\{u, v\}$ represent a vertex cut of G.
Recall that an adjacent component of a pair of vertices $\{u, v\}$ that represent a vertex cut of G, is a component of the graph $G - \{u, v\}$. There must be at least two adjacent components of $\{u, v\}$, since the removal of both u and v results in $G - \{u, v\}$, which only has at least two components if $\{u, v\}$ is a vertex cut.

Suppose that a Whitney switch is applied twice to the adjacent component G_i of $\{u, v\}$. Only edges that are incident with both a vertex from G_i and either u or v, are influenced by the Whitney switch, according to the definition of the Whitney switch. Suppose x is a vertex of G_i and there is an edge $\{u, x\}$ with label L in G. Then after the first Whitney switch has been applied, the resulting graph contains the edge $\{v, x\}$ with label L. After the the second Whitney switch, this resulting graph contains the edge $\{u, x\}$ with label L. The same holds for edges $\{v, x\}$.
Thus the resulting graph after two Whitney switches results in G, since all edges of G have either not been changed or have been changed, but resulted in the labeled edge being incident with exactly the same vertices as before applying the two Whitney switches.
This result can be generalized to: Applying any even amount of Whitney switches to any adjacent component of a vertex cut of size two results in a graph that is the same as the original graph before applying the Whitney switches, which are obviously equivalent.

Now for the case where an odd amount of Whitney switches is applied to every adjacent component of $\{u, v\}$. This case can be simplified to applying exactly one Whitney switch on every adjacent component, since applying two Whitney switches on the same adjacent component results in the exact same graph, which means that applying $2k + 1$ Whitney switches results in the same graph as applying only one Whitney switch. The edges that are incident with either u or v and are incident with a vertex contained in an adjacent component of $\{u, v\}$, are the only edges that change whenever a Whitney switch is applied to every adjacent component of $\{u, v\}$. This means that for every edge $\{u, x\}$ and $\{v, y\}$ in G, where x and y are vertices of any adjacent component, the resulting graph after all Whitney switches have been applied contains the edges $\{v, x\}$ and $\{u, y\}$. All other edges of G remain the same. This implies that if the bijective mapping ϕ from the definition of equivalent graphs is defined as $\phi(u) = v$, $\phi(v) = u$ and $\phi(z) = z$ for every $z \in V(G)\{u, v\}$, then the labels of $\{u, x\}$ and $\{v, y\}$ in G imply that the edges in the resulting graph should be $\{v, x\}$ and $\{u, y\}$, which is the case by the Whitney switch preserving the labels of the edges. Thus these graphs are equivalent.

Lastly, suppose that there is at least two adjacent component where a Whitney switch is applied an odd amount of times to one and an even amount of times to the other adjacent component. Then by the reasoning above, the adjacent component with an even amount of Whitney switches applied requires $\phi(x) = x$ for every vertex x in G to be equivalent to G, while the adjacent component with an odd amount of Whitney switches applied requires $\phi(u) = v$ and $\phi(u) = v$ to be equivalent to G. This implies that every other possible graph achieved by Whitney switches is not equivalent to G. □

This theorem leads to the number of different graphs achievable by Whitney switches for a pair of vertices that represent a vertex cut as shown in Corollary 3.7.

Corollary 3.7. *Suppose G is a 2-connected graph with labeled edges. Furthermore, suppose the set of vertices $\{u, v\}$ are a vertex cut of G. Let n be the number of adjacent components of $\{u, v\}$.*
The number of different graphs that can be achieved by applying any amount Whitney switches on adjacent components of $\{u, v\}$, is given by: 2^{n-1}.

Proof. Let G be a 2-connected graph with labeled edges and let the set of vertices $\{u, v\}$ represents a vertex cut of G.
Recall that an adjacent component of a pair of vertices $\{u, v\}$ that represent a vertex cut of G, is a component of the graph $G-\{u, v\}$. There must be at least two adjacent components of $\{u, v\}$, since the removal of both u and v results in $G - \{u, v\}$, which only has at least two components if $\{u, v\}$ is a vertex cut.
Let G_1, G_2, ..., G_n be all the adjacent components of $\{u, v\}$. Suppose that every component get assigned a '0' if an even amount of Whitney switches has been applied and a '1' if an odd amount of Whitney switches has been applied. This implies that the number of every adjacent component is a 0 for the graph G, since applying a Whitney switch an even number of times on any adjacent component results in the exact same graph as the graph that had no Whitney switches applied to that adjacent component. This claim is also supported by Theorem 3.6, which implies that the only equivalent graphs of G achievable by Whitney switches would have all 0's or 1's.
Suppose G_i is assigned a 1 and all other adjacent components are assigned a 0, which is equivalent to applying one Whitney switch on G_i and applying no other Whitney switches. Let the resulting graph be called G'. Then by Theorem 3.6 and the definition of the assigned number, G' is equivalent to graphs where G_i is assigned a 1 and all other adjacent

components are assigned a 0 and is equivalent to graphs where G_i is assigned a 0 and all other adjacent components are assigned a 1. Furthermore, G' is not equivalent to any other graph achievable by Whitney switches on $\{u, v\}$.
This implies that for any arbitrary assignation of 0 or 1 to every adjacent component, there is exactly one other equivalent graph achievable by Whitney switches at $\{u, v\}$ and the graph is not equivalent to any other graph achievable by Whitney switches at $\{u, v\}$. Since there are n adjacent components of $\{u, v\}$, the number of different graphs achievable by applying any amount Whitney switches on G is given by 2^{n-1}. □

4 SPQR-tree

Whitney's theorem provides a method to count the number of 2-isomorphic graphs of any 2-connected graph. To solve the problem, one needs to consider every Whitney switch of the given graph.
As seen in Section 3.1, a Whitney switch is applied to a pair of vertices that represent a vertex cut. To consider every Whitney switch, one needs to find every pair of these vertices in the graph.
No vertex of any of these pairs of vertices can be part of a 3-connected component of the graph, since these parts of the graph require at least three vertices to be removed to disconnect the graph. Furthermore, all other parts of the graph are 2-connected and require at least two vertices to be disconnected. This property ensures that no vertex of a 2-connected component of the graph is part of a 3-connected component of the graph. Thus no vertex of these pairs of vertices can be contained in a 3-connected component.
This results in the need to differentiate 2-connected and 3-connected component of the graph, which is solved by using a decomposition of a graph called the SPQR-tree. The *SPQR-tree* is a data structure that is similar to a graph. It can contain nodes of four different types and edges between these nodes. The SPQR-tree is always a tree, which implies that it is connected and does not contain any cycles. The creation of the SPQR-tree is described below to understand what the SPQR-tree is and how it is defined.

Step 1: 3-connected components. The first step in making an SPQR-tree of a graph is identifying all 3-connected components of the graph, where a 3-connected component of a graph is a maximal subgraph based on a set of vertices that have at least three internally disjoint paths between them in the graph. Furthermore, the 3-connected component should not be contained in any other maximal 3-connected subgraph. The reader should also note that a graph can contain multiple different 3-connected components. These 3-connected components are important due to containing no pairs of vertices that represent a vertex cut. Another important aspect of these components is that Whitney's theorem which implies that there are no other 2-isomorphic graphs to a 3-connected graph than the 3-connected graph itself, since no Whitney switches can be applied in a 3-connected graph by the definition of the Whitney switch requiring to be applied to a pair of vertices that represent a vertex cut.

These 3-connected components results in two different node types of the SPQR-tree: The P-node and R-node. However, some other definitions need to be given before properly being able to give the definition of these node types.
The nodes of the SPQR-trees represent a maximal subgraph of the graph, which is referred to as the *skeleton* of the node. The skeleton of a node can contain *virtual edges*, which are edges that have been added to the skeleton of the node that are used to resemble at least

one path between two vertices, where the path is not included in the skeleton of that node. Every edge of a skeleton of a node in the SPQR-tree is referred to as a *real edge*.
This is visualized in Figure 2, where an example is given of a 3-connected component of a graph being connected to some other component $G2$ by a pair of vertices that are a vertex cut. There is a path between the red vertex and the blue vertex in $G2$, since the graph has to be 2-connected and otherwise the red and blue vertices would both be vertex cuts by themselves. The black vertices on the left and the colored pair of vertices are 3-connected with each other, since there are at least three internally disjoint paths between any of these pairs of vertices. One of the paths between the red and blue vertices is completely in $G2$, which is represented in the skeleton of the SPQR-tree node by a virtual edge (the dotted line between the vertices). The reader should also note that vertices can be part of skeletons of different SPQR-tree nodes, but edges of the graph cannot be part of skeletons of multiple nodes.

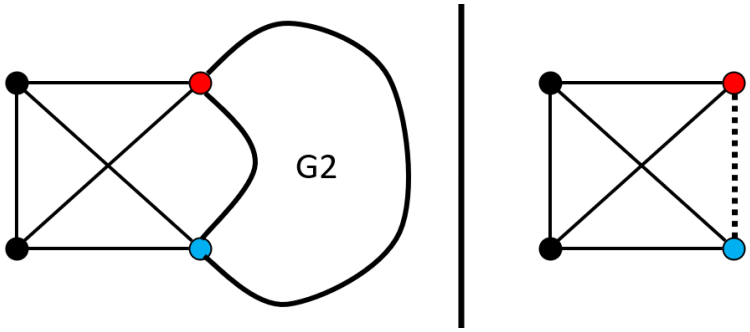

FIGURE 2: A 3-connected component (left) and the skeleton of the SPQR-tree node (right). The colored vertices are a vertex cut.

Now we return to the P-node and R-node. The P in P-node refers to 'parallel' case, since the skeleton of a P-node contains exactly two vertices and at least three (virtual) edges between the two vertices.
The R-node represents the 'rigid' case. The skeleton of the R-node contains at least three vertices and is 3-connected. The skeleton in Figure 2 is an example of an R-node.
There is one important notion when dealing with the skeleton of an R-node. Whenever the graph is designed as Figure 2, i.e. there is a vertex cut pair, one could argue that if there is an edge is added between the vertex cut pair, then that edge should be part of the skeleton of the R-node. However, this edge between the red and blue vertex would be part of a new P-node. The skeleton of this P-node would consist of the added edge, a virtual edge representing the path from the red vertex to the blue vertex in $G2$ and a virtual edge for the paths by traversing the black vertices. In short, the P-node takes priority if there is an edge between a pair of vertices that represents a vertex cut.
More reading material is available on this subject in for example [5], which is a paper of Hopcroft and Tarjan that talks about this particular step of creating the SPQR-tree. Furthermore, [4] is a paper about a linear time implementation of making an SPQR-tree of a graph, which includes some corrections on the paper of Hopcroft and Tarjan.

Step 2: All other components of the graph The second step of making a SPQR-tree of a graph is to look at the other components of the graph that are not part of a 3-connected component.
The first case is whenever the graph has exactly 2 vertices and 2 edges between these

vertices. Then the graph contains no 3-connected components and consists of two Q-nodes. The skeleton of the Q-node is an edge between two vertices. The Q-node is only used in this trivial case.
The second case involves all edges that are not part of any 3-connected component. These edges are part of (possibly multiple different) S-nodes. This node type is used in case of a 'series'. The skeleton of an S-node is a cycle of at least length 3 that consists of edges and possibly virtual edges. The skeleton of an S-node is not 3-connected, since there are exactly 2 internally disjoint paths between any pair of vertices in that cycle.

The structure of the SPQR-tree should be discussed before going further into depth about these S-nodes and how the skeletons are determined from a given graph. The SPQR-tree consists of nodes and edges between nodes. An edge of the SPQR-tree represents a pair of vertices that are a vertex cut. These vertices connect two different parts of the graph.
An example is given in Figure 3, where the given graph G consists of two 3-connected components that share the two colored vertices. These components are the skeletons of the $R1$ and $R2$ nodes. There are no other components in the SPQR-tree, thus the SPQR-tree can be made as shown at the bottom of the figure. One could think of the skeleton of the SPQR tree as skeletons of multiple nodes where the virtual edges that are connected represent the adjacent nodes in the SPQR-tree.

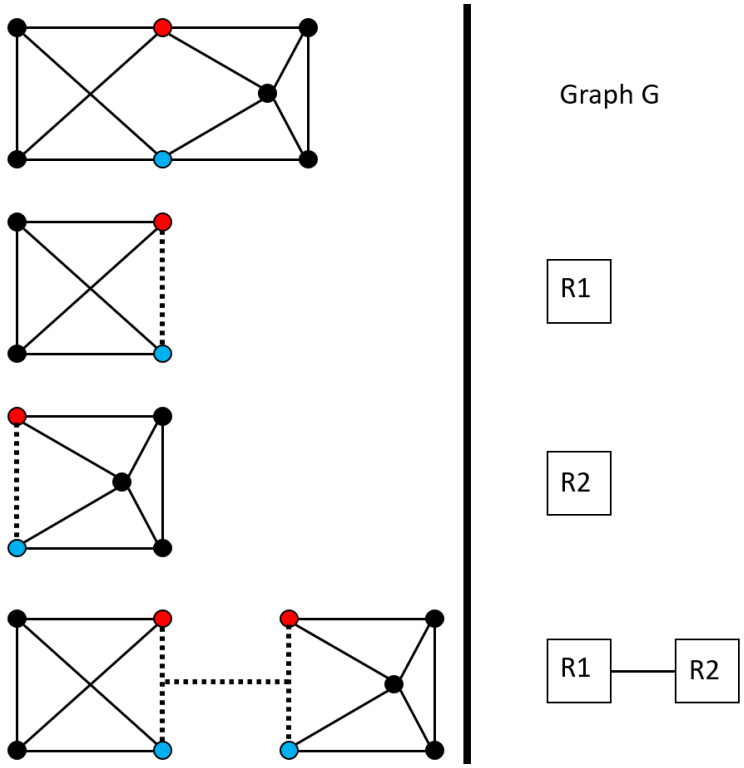

FIGURE 3: Making the SPQR-tree of a graph made of two 3-connected components.

An additional example would also show how an S-node is found. For simplicity, the new graph should look similar to the graph in Figure 3. This new example is shown in Figure 4 and shows a graph and the corresponding SPQR-tree. This is the SPQR-tree of the graph, since the 3-connected components have been identified and separated, which lead to a single edge that was not part of any 3-connected component, but creating a cycle with the virtual edges of the 3-connected components. This skeleton would result in a

SPQR-tree that looks like two different R-nodes connected to a single S-node. The reader should note that it is also possible for P-nodes to be adjacent to S-nodes and that it is handled similarly to the example sketched above.

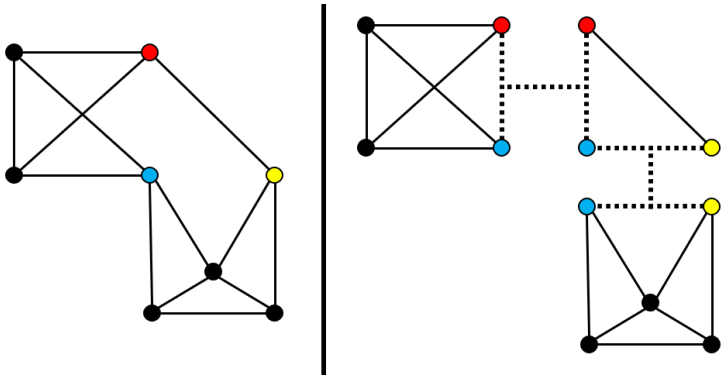

FIGURE 4: Cycle in skeleton of SPQR-tree.

Lastly, the minimal SPQR-tree does not allow two adjacent S-nodes or two adjacent P-nodes. If two P-nodes are adjacent, then there is a opportunity to merge the two P-nodes in a single P-node that contain the edges of both P-nodes, since two P-nodes can only be connected if the vertices of the skeletons are the same.
S-nodes are not allowed to be adjacent, which is best explained by a graph that is a cycle of length 4. One could suppose that two adjacent edges and their respective vertices represent an S-node and that the other two edges in the cycle represent the skeleton of another S-node. Then the skeletons share exactly two vertices, which would have a virtual edge between them in both skeletons. However, the two S-nodes can be combined into a single S-node by letting the skeleton of that node be a cycle of length 4.

Step 3: Nodes with only virtual edges. This step is very similar to the previous step. This step assigns S-nodes and P-nodes to cases similar to Figure 4, except where the edge between the red and yellow vertex is replaced with another 3-connected component. In that specific case, the skeleton of the S-node would only consist of virtual edges.
Another example is shown in Figure 5, where a P-node is necessary to connect the SPQR-tree and where the P-node contains only virtual edges.

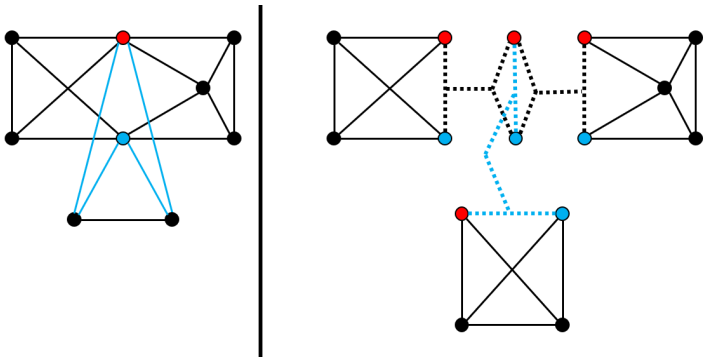

FIGURE 5: SPQR-tree with skeleton of P-node. Blue edges are regular edges used to avoid confusion due to overlap.

Again, there should not be adjacent S-nodes and P-nodes as explained in step 2. SPQR-

trees that do not allow adjacent S-nodes and P-nodes are also called minimal SPQR-trees.

Thus these steps should enable the reader to create an SPQR-tree from a given 2-connected graph. There are properties of the minimal SPQR-tree that are not proven in this research which can be found in other papers such as [2] and [4], which talk about creating the SPQR-tree of a graph in linear time. These properties include that the SPQR-tree is a tree and that the minimal SPQR-tree of a 2-connected graph is unique.

5 Counting 2-isomorphisms with SPQR-trees

In Section 3.2 a method is shown to derive the number of different graphs achievable by applying Whitney switches to a pair of vertices that represent a vertex cut. In Section 4, these pairs of vertices are represented by edges in the SPQR-tree. However, the skeleton of the S-node can also contain pairs of vertices that represent a vertex cut. This leads to Theorem 5.1.

Theorem 5.1. *Consider an SPQR-tree T of a 2-connected graph G with labeled edges, where T consists of one S-node and no other nodes. Let n be the length of the cycle in the skeleton of the S-node. The number of different 2-isomorphic graphs of G is given by: $\frac{(n-1)!}{2}$.*

Proof. The skeleton of an S-node has length $n \geq 3$ by definition, which implies that the graph has n vertices and n edges.
Suppose that the cycle is defined as the ordered list of labels of edges that form the cycle and preserves the adjacency of edges with respect to other edges, where an edge is adjacent to another edge if both edges have at least one vertex in common. Suppose that the labels of the edges are chosen such that the cycle of length n such that the resulting ordered list is given by (1).

$$(1, 2, 3, ..., n-1, n) \tag{1}$$

The ordered list of (1) implies that the edge with label 1 is adjacent to the edges with labels 2 and n, the edge with label i is adjacent to the edges with labels $i-1$ and $i+1$ for $i \in \{2, 3, ..., n-1\}$, and the edge with label n is adjacent with the edges with labels $n-1$ and 1. There are a total of $n!$ possible ordered lists that represent 2-isomorphic graphs to the cycle.

One of the equivalent graphs is given by $(2, 3, ..., n-1, n, 1)$, since the set of adjacent labeled edges of every labeled edge is exactly the same as in (1). To prevent this type of equivalent graphs, one could 'fix' an edge of the cycle, which would imply that the edge is always in the same place of the ordered list.
Suppose the edge 1 of (1) is fixed in the first spot of the ordered list. To then find the number of different graphs one only has to look at all possibilities for the ordered list $(2, 3, ..., n-1, n)$. This reduces the number of possibilities to $(n-1)!$ Furthermore, this restriction of fixing an edge does not reduce the number of different possible cycles, since for any ordered list where the fixed edge is not in the correct location (for example $(2, 3, ..., n-1, n, 1)$ where 1 should be fixed to the first spot of the ordered list), there is an equivalent graph where the labeled edge is in the correct spot ($(1, 2, 3, ..., n-1, n)$ in case of the example).

Suppose edge 1 is again fixed for (1), then there is an equivalent cycle given by $(1, n, n-1, ..., 3, 2)$, which is equivalent since the set of adjacent labeled edges of every

labeled edge is exactly the same as in (1). Every possible ordered list with a fixed edge has exactly one of these types of equivalent graphs, which is given by applying the permutation $[1, n, n-1, ..., 3, 2]$ to the ordered list (assuming that the fixed edge is in position 1 of the ordered list). This reduces the number of possibilities to $\frac{(n-1)!}{2}$

Suppose that (1) has another equivalent graph. This equivalent graph would have an edge fixed (for simplicity choose edge 1). Then the list can not be $(1, n, n-1, ..., 3, 2)$, since that case is accounted for. However, the adjacent edges of 1 should be the same as (1), which implies that the list must end with n and start with 1 and a 2. Repeating this process for the edges 2 and n and then for edges 3 and $n-1$ and etc. will result in the same list as (1). Thus the only equivalent cycle is the cycle itself. This implies that there are a total of $\frac{(n-1)!}{2}$ possible ordered lists that represent 2-isomorphic graphs to the cycle, since the cycle is obviously preserved in every different graph, which implies that the graph is 2-isomorphic. □

All possible vertex cuts of size two are represented in the skeletons of S-nodes and edges of the SPQR-tree, since all other nodes of the SPQR-tree are 3-connected, which implies that the minimal vertex cut size is 3. This results in the number of different graphs that are achievable by Whitney switches. Then applying Whitney's theorem results in the number of different 2-isomorphic graphs. This is used in the proof of the following theorem to derive the number of different 2-isomorphic graphs of a given 2-connected graph G.

Theorem 5.2. *Suppose G is a 2-connected graph with labeled edges. Let T be the minimal SPQR-tree of G. The number of different 2-isomorphic graphs of G is equal to*

$$\max\left\{1, (2^{n_R-1}) \cdot \prod_{i=1}^{n_S} (|C_i| - 1)!\right\} \tag{2}$$

Where n_R is the number of R-nodes in T, n_S is the number of S-nodes in T and $|C_i|$ refers to the length of the cycle in the skeleton of an S-node in T.

Proof. If the SPQR-tree of G consists of only two Q-nodes with an edge between these to nodes, then G would consist of two vertices and two edges between these two vertices. There is no pair of vertices that represent a vertex cut, since removing the two vertices in the graph does not result in a graph with at least two components. This implies that no Whitney switches can be applied to this graph and thus by Whitney's theorem, the number of different 2-isomorphic graphs is given by 1. The Q-node is not used in any other SPQR-tree.

All other cases can be proven by induction. Start with a trivial SPQR-tree and keep adding a node and an edge to the SPQR-tree such that the SPQR-tree remains a tree, i.e. the SPQR-tree should be connected and contain no cycles. The base cases are given by the following trivial SPQR-trees:

1. If the SPQR-tree contains one R-node and no other nodes, then the graph cannot contain a pair of vertices that represent a vertex cut, since the graph is 3-connected, which directly implies that at least 3 vertices are required to disconnect the graph. Thus no Whitney switches can be applied, which implies that the number of different 2-isomorphic graphs of the skeleton of an R-node is 1 by Whitney's theorem.

2. If the SPQR-tree contains one P-node and no other nodes, then the graph cannot contain a pair of vertices that represent a vertex cut by similar reasoning of the Q-node case. Thus no Whitney switches can be applied, which implies that the number of different 2-isomorphic graphs of the skeleton of a P-node is 1 by Whitney's theorem.

3. If the SPQR-tree contains one S-node and no other nodes, then the number of different 2-isomorphic graphs of the skeleton of an S-node is $\frac{(|C|-1)!}{2}$ by Theorem 5.1.

All the above given cases of an SPQR-tree with one node, agree on (2) for the number of different 2-isomorphic graphs.

Consider the minimal SPQR-tree T of the 2-connected graph G with labeled edges, where T has $n > 1$ nodes. Since T is a non-trivial tree, it is known that there are at least two leaf nodes of T, where a leaf node is a node with one incident edge in T. Let L be a leaf node and let L be incident with an edge e of the T. This edge e represents a pair of vertices u and v. These vertices are contained in the skeleton of L and are contained in the skeleton of the node W, which is a node of the SPQR-tree that is adjacent to L. Let the removal of a leaf node of T result in T', where the virtual edge representing the leaf node in the skeleton W is replaced with a real edge for T'.
By the induction hypothesis, it can be assumed that the number of different 2-isomorphic graphs of the graph corresponding to T' is given by (2). Add L to T' by replacing the real edge $\{u, v\}$ with a virtual edge that represents the leaf node to recreate T. This results in cases where L can be a P-type, R-type or S-type node and cases where W is any node-type.

If the leaf node L is an R-node, then the number of different 2-isomorphic graphs of the graph corresponding to G is 2 times the number of different 2-isomorphic graphs of the graph of T'. L can be attached to the following types of node:

1. If L is attached to an R-node R_1 of T', then by the earlier removal of L from T to create T', a virtual edge in the skeleton of R_1 was replaced by a real edge. This real edge is removed and a virtual edge representing L is added to the skeleton of R_1. The vertices of this virtual edge now represent a vertex cut, which implies that the number of different 2-isomorphic graphs based on this vertex cut is 2.
   However, this pair of vertices were not a vertex cut in the skeleton of T', since these vertices were only connected by one edge and were part of the skeleton of an R-node, which implies that removing these two vertices could not disconnect the graph.
   This implies that the number of different 2-isomorphic graphs of G is two times the number of different 2-isomorphic graphs of the graph of T', which implies that (2) is correct in this case.

2. If L is attached to a P-node of T', then the pair of vertices in the skeleton of the P-node in the created T gains one more adjacent component, since an edge between the pair of vertices is replaced by a skeleton of L. This implies that the number of different 2-isomorphic graphs of G is 2 times the number of different 2-isomorphic graphs of the graph of T' by Corollary 3.7. However, if the T' only contains one P-node and no other nodes, then the number of adjacent components of the vertices of the P-node is equal to 0. When adding L to create T, the number of adjacent components of the pair of vertices becomes 1, which leads to only 1 different 2-isomorphic graph. However this case still agrees on (2), since the outcome of the equation for an SPQR-tree with one P-node and one R-node is 1. Thus the number of different 2-isomorphic graphs of G is given by (2).

3. If L is attached to an S-node of T', then a real edge in the skeleton of the cycle is replaced again. The vertices incident with this real edge are not a vertex cut in the skeleton of T', since there is only one real edge between these two vertices, which implies that there are no vertices adjacent to both of these vertices.
   The graph is 2-connected and there are no vertices connected to both of the removed vertices, which implies that there is at least one path between every pair of vertices in the graph that does not use either of the removed vertices. Thus the removal of these two vertices in the skeleton of T' results in a graph that is connected and thus the removed vertices is not a vertex cut.
   In T, there is an edge in the SPQR-tree that represents these two vertices, which directly implies that these vertices are a vertex cut. This vertex cut in T with two adjacent components that is not in T' implies that the number of different 2-isomorphic graphs of G is 2 times the number of 2-isomorphic graphs of the graph of T', which implies that (2) is correct in this case.

Suppose L is a P-node. The vertices of a P-node only represent a vertex cut if there are at least two adjacent nodes in the SPQR-tree, since these 2 adjacent nodes ensure that there are 2 adjacent components to the pair of vertices in the skeleton of the P-node, which are required for the vertex pair to be a vertex cut. However, the P-node is a leaf of the SPQR-tree, which implies that there is only one adjacent node in the SPQR-tree T. Thus the number of different 2-isomorphic graphs of G is the same as the graph of T', since the added P-node is always a leaf node of T.

If L is an S-node, then there are only two possible nodes in T' where the S-node can be adjacent to: a P-node or an R-node. L can not be adjacent to another S-node, since that would contradict T being a minimal SPQR-tree.

1. If L is attached to an R-node of T', then by similar reasoning to the case where the leaf node is an R-node and is attached to an R-node of T', the number of different 2-isomorphic graphs created by the newly added vertex cut is 2 by Corollary 3.7. However, the skeleton C of L also has $\frac{(|C|-1)!}{2}$ different possible 2-isomorphic graphs. This implies that the number of different 2-isomorphic graphs is $(|C|-1)!$ times the number of different 2-isomorphic graphs of the graph of T', which implies that (2) is correct in this case.

2. If L is attached to a P-node of T', then by similar reasoning to the case where the leaf node is an R-node and is attached to a P-node of T', the number of adjacent components of the vertex pair in the skeleton of the P-node is increased by 1. Thus the number of different graphs as implied by that vertex cut is multiplied by 2 according to Corollary 3.7. However, if T' is only one P-node and no other nodes, then the number of different graphs is not multiplied by two. Furthermore, the skeleton of the S-node also has $\frac{(|C|-1)!}{2}$ different possible 2-isomorphic graphs, where $|C|$ is the length of the cycle in the skeleton of the S-node. This implies that the number of different 2-isomorphic graphs is $\frac{(|C|-1)!}{2}$ times the number of different 2-isomorphic graphs of the graph of T' whenever T' consists of only a P-node, and $(|C|-1)!$ times the number of different 2-isomorphic graphs of the graph of T' whenever T' does not consist of only a P-node. This implies that (2) is correct in this case.

The number of different possibilities of the cycle in the skeleton of the S-node can be multiplied by the number of possibilities implied by the P-node, since any different possible cycle results in a different graph for any possible combination of Whitney switches at the

vertex cut.
Thus the number of 2-isomorphic graphs of T is given by (2). What remains to be proven is that any minimal SPQR-tree can be reduced to the trivial SPQR-tree by repeatedly removing leaf nodes. This would prove the statement by induction. Suppose T is a non-trivial minimal SPQR-tree, then it is known that T is a tree. Removing a leaf node of T results in another tree, since T is obviously connected if a leaf node is removed and since T contains no cycles. Then for any non-trivial tree it is known that there are at least two leaf nodes [1], which implies that removing a leaf node of T results in a smaller SPQR-tree that has at least two leaf nodes. This removal of leaf nodes can be repeated until the SPQR-tree is trivial.
Thus by induction: the number of different 2-isomorphic graphs for the minimal SPQR-tree T of the 2-connected graph G with labeled edges is given by (2). □

6 Enumerating all 2-isomorphic graphs efficiently

In the previous section, the total number of different 2-isomorphic graphs of a given 2-connected graph with labeled edges is calculated based on the minimal SPQR-tree of the graph. This result helps us achieve the first research goal of determining the total number of different 2-isomorphic graphs. However, the second research goal is about enumerating all of these different 2-isomorphic graphs efficiently.

To enumerate all these graphs, one needs to first look at how the total number of different 2-isomorphic graphs is achieved based on the minimal SPQR-tree. The number of 2-isomorphic graphs depends on every S-node in the SPQR-tree and on every vertex cut of size two that is represented by the edges of the minimal SPQR-tree.

The skeleton C of every S-node is a cycle and has $\frac{(|C|-1)!}{2}$ different 2-isomorphic graphs by Theorem 5.1, where $|C|$ is the length of the cycle.

The vertex cuts represented by the edges in the minimal SPQR-trees have 2^{n-1} different 2-isomorphic graphs by Corollary 3.7 and Whitney's theorem, where n is the number of components in the graph with the two vertices of the vertex cut removed (adjacent components).
If $n \geq 3$, then the minimal SPQR-tree contains a P-node that contains the two vertices of the vertex cut. This P-node has n neighbouring nodes in the minimal SPQR-tree, since every adjacent component must be represented by a node in the SPQR-tree. Furthermore, all these neighbouring nodes are either R-type or S-type nodes, since there are no adjacent P-nodes in the minimal SPQR-tree and since the Q-node is only used in case of the graph being two vertices and two edges, which is not the case whenever there is a P-node in the SPQR-tree. Thus the P-node has n neighbours, which implies that there are n edges incident with the P-node in the minimal SPQR-tree.
If $n = 2$, then there is only a P-node whenever there is at least one edge between the two vertices that represent the vertex cut. This case would result in 2 neighbouring nodes to the P-node and thus 2 edges incident with the P-node in the SPQR-tree. If there is no edge between the two vertices, then this vertex cut is represented by a single edge between either two R-nodes or an R-node and an S-node of the minimal SPQR-tree, since no two S-nodes can be adjacent in a minimal SPQR-tree. There is 1 edge that represents this vertex cut and it is not incident with any P-node.
If there is a P-node that is a leaf of the minimal SPQR-tree, then there is only one adjacent component of the graph, thus there is 1 edge incident with that P-node, which results in

no additional different 2-isomorphic graphs.
From all these cases of the number of adjacent components of pairs of vertices that represent a vertex cut, that the number of edges that represent the vertices of the vertex cut in the minimal SPQR-tree is related to the number of adjacent components of that vertex cut. This results in $n-1$ being equal to the amount of edges that represent the vertex cut minus the amount of P-nodes that use both vertices of the vertex cut, i.e.

$$2^{n-1} = 2^{\epsilon - p}$$

Where 'ϵ' is the number of edges in the minimal SPQR-tree and 'p' is the number of P-nodes in the minimal SPQR-tree.

An algorithm that enumerates all different 2-isomorphic graphs of a given graph G starts by making the minimal SPQR-tree and then finding all P-nodes.
Then the algorithm decides to either apply a Whitney switch to an adjacent node of this P-node or choose not to apply a Whitney switch to an adjacent node. The algorithm makes the decision based on Theorem 3.6 to not create equivalent graphs by applying a Whitney switch to every adjacent node. To prevent this issue, the algorithm chooses one of the adjacent nodes and decides to never apply a Whitney switch to that node. This still results in the same number of different 2-isomorphic graphs, as shown in the proof of Corollary 3.7.
For every edge that is not incident with any P-node in the SPQR-tree, there are two possible different 2-isomorphic graphs achievable by applying Whitney switches according to Corollary 3.7. It does not matter on which adjacent component the Whitney switch is applied, since the resulting graphs turn out to be equivalent by Theorem 3.6. Thus the algorithm either decides to apply a Whitney switch or not.

When dealing with the S-nodes, the algorithm considers the (virtual) edges of the skeleton of the S-node as an ordered list of the labels of edges that form the cycle, where every position in the list refers to an edge in the skeleton of the cycle. So a list of length k could be given by $\{1, 2, 3, ..., k-1, k\}$, where the numbers in this list are labels of edges. Then according to the proof of Theorem 5.1, a label should be fixed to an edge should be fixed to prevent equivalent graphs. Suppose that the label for the edge represented by the first position of $\{1, 2, 3, ..., k-1, k\}$ is fixed, then the only other situation of an equivalent graph that should be avoided, is given by $\{1, k, k-1, ..., 3, 2\}$. This is also shown in the proof of Theorem 5.1.

Thus to efficiently enumerate every different 2-isomorphic graph of a given 2-connected graph G with labeled edges, one needs to:

1. Create the minimal SPQR-tree of the given graph.
2. Find all adjacent components of pairs of vertices that are represented by an edge of the SPQR-tree. If there is a P-node for that pair of vertices, then choose one incident edge to never apply a Whitney switch to the adjacent components with respect to that specific pair of vertices. This adjacent component is fixed with respect to that pair of vertices.
3. For every non-fixed adjacent component of a pair of vertices that represent a vertex, choose to either apply a Whitney switch or to not apply a Whitney switch.

4. For every S-node, choose one of the $\frac{(|C|-1)!}{2}$ different 2-isomorphic graphs, where $|C|$ is the length of the cycle. This can be done by representing the graph as an ordered list of labels $\{1, 2, 3, ..., |C|-1, |C|\}$, where the position of the label in the ordered list represents an edge of the skeleton of the cycle. Choose one label to be fixed in the ordered list, suppose label 1 in the first position of the ordered list. Then all possibilities of the other parts of the list, i.e. $\{2, 3, ..., |C|-1, |C|\}$, need to be considered. For every chosen ordered list of labels $\{L_1, L_2, ..., L_{|C|-2}, L_{|C|-1}\}$, the cycle becomes $\{1, L_1, L_2, ..., L_{|C|-2}, L_{|C|-1}\}$. Furthermore, the algorithm does not choose $\{L_{|C|-1}, L_{|C|-2}, ..., L_2, L_1\}$ in any other iteration from now on, since that would create equivalent graphs.

5. Enumerate every combination of either applying or not applying Whitney switches of step 3 and every possible skeleton of every S-node that does not result in an equivalent graph as described in step 4. The output of the algorithm is a graph, which implies that the decision of either applying or not applying a Whitney switch and the decision for every S-node needs to be executed by the algorithm.

6.1 Efficiency

The algorithm to enumerate all different 2-isomorphic graphs has been designed. This leaves the question on how efficient this algorithm is.

Algorithm analysis The first step of the algorithm is creating the minimal SPQR-tree, which can be done in linear time based on the number of vertices and the number of edges, as shown in [4].

The second step involves finding every pair of vertices that are represented by at least one edge in the SPQR-tree and are represented by a P-node. This can be achieved by searching for every P-node in the graph. Then for every P-node the algorithm should choose one adjacent component where a Whitney switch can not be applied. This is equivalent to choosing a neighbouring node where a Whitney switch can not be applied with respect to the pair of vertices.
For every other edge in the SPQR-tree, there should be freedom of choosing whether or not a Whitney switch can be applied or not. Applying a Whitney switch on an edge of the SPQR-tree means that a Whitney switch is applied to an R-node or S-node that is incident with the edge.
This part only has to be done once at the start of the algorithm. This step should be able to be done in $O(|V|)$ time, where $|V|$ is the number of nodes of the SPQR-tree. Choosing whether or not to apply a Whitney switch to an edge should not be computationally difficult.

The third step involves a method of choosing the position of the labeled edges in the skeleton of every S-node. This firstly requires the S-nodes of the SPQR-tree, which can be found while searching for the P-nodes during the second step. Choosing the labels to assign to the edges of the cycle should not be a computationally difficult task.
However, suppose that the skeleton of the S-node is length 4 or more and contains a virtual edge between vertices v_1 and v_2. To change the position of this virtual edge in the skeleton of S to be an edge between v_3 and v_4 requires all edges that are incident with either v_1 or v_2 in the other node(s) represented by the virtual edge have to be correctly changed to v_3 or v_4. This shows that the choice of the edges in the skeleton of the S-node is not computationally difficult, but altering the edges in the graph might turn out to be computationally difficult.

The fourth step is to try all different possibilities of either applying or not applying Whitney switches and all different possibilities for the skeleton of the S-nodes of the SPQR-tree. This requires choosing whether or not to apply a Whitney switch for every edge and assigning labels to the edges of every cycle, which needs to be done $2^{\epsilon - n_P} \cdot \prod_{i=1}^{n_S} \frac{(|C_i|-1)!}{2}$ times, where n_P is the number of P-nodes, n_S is the number of S-nodes, and $|C_i|$ is the length of the cycle in the skeleton of one of the S-nodes in the SPQR-tree.
The algorithm returns the graph after choosing and applying the Whitney switches and choosing and changing the cycles of the S-nodes.

Time complexity. To find the time complexity of the algorithm, an analysis of the time between outputting 2 different graphs is required. Let the algorithm choose a cycle for every P-node, after which the algorithm applies every combination of Whitney switches. Then the algorithm changes one of the cycles to another different graph. After which every possibility with Whitney switches is applied again. This process is repeated until every possible combination of different graphs in the S-nodes have been seen.
Choosing a new combination of Whitney switches can be done by choosing to apply a Whitney switch to one component: Let the list of 0's and 1's represent whether a Whitney switch is applied to that node with respect to some vertex cut. Then in the original graph this list is $[0, 0, 0, ..., 0]$. We want to find every possible list of 0's and 1's, since all of these components can be Whitney switched to create a different graph, as discussed earlier in this section. A way to find all these possibilities by applying only one Whitney switch at a time is given by the following reasoning:
Suppose the list is size two: $[0, 0]$, then the following sequence will cycle:

$$[0, 0] \to [0, 1] \to [1, 1] \to [1, 0]$$

. IF the list is size three, then the sequence is similar (note that one of the numbers is 0 for the first half of the sequence and 1 for the second half of the sequence):

$$[0, 0, 0] \to [0, 0, 1] \to [0, 1, 1] \to [0, 1, 0][1, 1, 0] \to [1, 1, 1] \to [1, 0, 1] \to [1, 0, 0]$$

. This sequence can be extended in a similar manner for any size list. This type of problem is characterized by the K-cube graph [1].

Thus the algorithm can find all possible combinations of Whitney switches while only applying one Whitney switch at a time.

Thus the algorithm either changes a cycle or applies one Whitney switch. The time complexity of both of these operations is $O(|E|)$, since every edge only needs to be changed once in these operations. This implies that enumerating every different 2-isomorphic graph has a time complexity of $O(N|E|)$, where N is the number of different 2-isomorphic graphs. This does not include the creation of the SPQR-tree, so the time complexity of enumerating every 2-isomorphic graph is estimated to be $O(|V| + |E|)$ by [4]. Thus the total time complexity is $O(|V| + (N + 1)|E|)$, which is equivalent to $O(|V| + N|E|)$

7 Conclusion

In this paper, we investigated the number of different 2-isomorphic graphs and proposed an algorithm to enumerate them efficiently. The number of different 2-isomorphic graphs of a given 2-connected graph G is found through the application of Whitney switches and the utilization of the SPQR-tree. The number of different 2-isomorphic graphs of a given 2-connected graph G can be determined by counting the number of R-nodes and considering the skeleton of every S-node in the minimal SPQR-tree of G. Let T be the minimal SPQR-tree of G, let n_R be the number of R-nodes in T, let n_S be the number of S-nodes in T and let $|C_i|$ be the length of the cycle of one of the S-nodes in T. Then the number of different 2-isomorphic graphs is given by

$$\max\left\{1, (2^{n_R-1}) \cdot \prod_{i=1}^{n_S}(|C_i|-1)!\right\}$$

To enumerate every 2-isomorphic graph an algorithm is discussed that achieves a time complexity of $O(|V| + N|E|)$ to enumerate all N different 2-isomorphic graphs. This algorithm has not been formally described by pseudo-code, which can be explored further in a potential follow-up research.

References

[1] J. A. Bondy, U. S. Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[2] G. Di Battista and R. Tamassia. On-Line Maintenance of Triconnected Components with SPQR-Trees. *Algorithmica (New York)*, 15(4), 1996.

[3] F. V. Fomin and P. A. Golovach. Kernelization of whitney switches. *SIAM Journal on Discrete Mathematics*, 35(2), 2021.

[4] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1984, 2001.

[5] J. E. Hopcroft and R. E. Tarjan. Dividing a Graph into Triconnected Components. *SIAM Journal on Computing*, 2(3), 1973.

[6] H. Whitney. 2-Isomorphic Graphs. *American Journal of Mathematics*, 55(1/4), 1933.