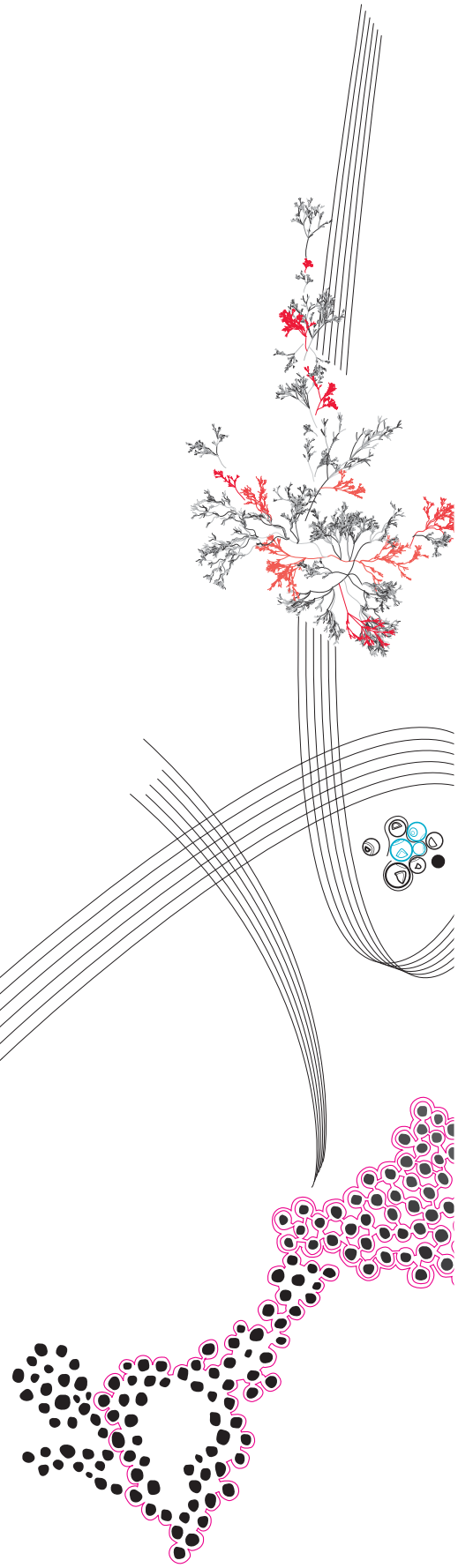


BSc Thesis Applied Mathematics

Computing certificates for the graph realization problem on 3-connected graphs

Gavin Speek

Supervisor: Matthias Walter

June, 2021

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

UNIVERSITY OF TWENTE.

Preface

I would like to thank Dr. Matthias Walter for providing guidance and feedback, and for sharing his enthusiasm for this project.

Computing certificates for the graph realization problem on 3-connected graphs

Gavin Speek*

June, 2021

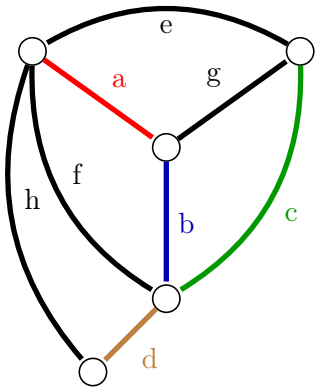
Abstract

We produce an algorithm for minimizing non-graphic matrices, by exploiting structures of 3-connected graphs represented by their submatrices.

Keywords: graph, realization, graphicness, algorithm, 3-connected, matrix, representation

1 Introduction

For the graph realization problem we are given a 0/1-matrix, for which we need to decide if it is a representation matrix or not. In a representation matrix the rows are indexed by the spanning tree edges of a graph and the columns are indexed by the remaining edges. The 1s in a column show which spanning tree edges form a cycle with the indexed edge. See Figure 1 for an example. If there exists a graph such that a 0/1-matrix is the corresponding representation matrix, we say that this matrix is graphic. There already exist almost linear-time algorithms for this problem, which take a 0/1-matrix as input and give a corresponding graph if it is graphic [5].



$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

FIGURE 1: Example of a graph and its representation matrix. The spanning tree edges are colored.

In the case where the input matrix is non-graphic these algorithms simply state this and stop. We want to create an algorithm that exploits certain substructures, in order to

*Email: g.w.speek@student.utwente.nl

compute a minimal non-graphic submatrix. These submatrices can be used as certificates to verify that a given matrix is indeed non-graphic. This can be useful to convince the user of this result and could give an insight as to why a given matrix is non-graphic.

2 Prior knowledge

It is assumed that the reader has some knowledge of graph theory, so definitions that can be found in most books on graph theory will not explicitly be explained. We will now introduce some more specific definitions that we will use.
Two graphs are 2-isomorphic if they produce the same representation matrix.
A set of edges in a graph G is a hypopath if it forms a path in some graph 2-isomorphic to G. Since we only discuss 3-connected graphs, we can use that two 3-connected graphs are 2-isomorphic if and only if they are strictly isomorphic [6]. Thus in our case a set of edges is a hypopath if and only if it is a path. If a set of edges does not form a path, we consider it a non-path.

In Section 1 we briefly mentioned that in a representation matrix the rows are indexed by the spanning tree of some graph and the columns by the rest of the edges. We want to keep a close relation between a graph and its representation matrix, so we explain the effects of the operations on both graphs and their representation matrices.

2.1 Operations

First we will discuss some basic operations. Let $G = (V, E)$ be a finite simple graph and let M be its representation matrix, denoted by $M(G)$. Let $e \subseteq E$ be an edge in G. If we delete e from G, denoted by $G - e$, we obtain the graph induced from G by removing e. In G we can only delete non-spanning tree edges, as deleting a column in M corresponds to deleting an edge in G.

An edge contraction is performed by deleting an edge e and identifying its endpoints. This is denoted by G/e. Opposite to edge deletion, we can only contract spanning tree edges, as deleting a row in M corresponds to contracting an edge in G.

We can also delete vertices. Let $v \subseteq V$ be a vertex in G. Deleting v from G, yields the graph induced from G by removing v and all its incident edges. Vertex deletion is not a clear operation for matrices and will not be used, so we chose to omit this definition for matrices.

The last operation we cover is the pivot. With this operation we can change the spanning tree of a graph or representation matrix. In Figure 2 the pivot is explained for a representation matrix and in Figure 3 for a corresponding graph.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & \textcircled{1} & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ a \\ f \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} e & b & g & h \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

FIGURE 2: Let X and Y be the set of row and column indices respectively. A pivot is performed by first picking a pivot element $M_{x,y}$ for $x \in X$ and $y \in Y$, which must be a 1. Then we replace for every $u \in (X - \{x\})$ and every $w \in (Y - \{y\})$, $M_{u,w}$ by $M'_{u,w} = M_{u,w} + (M_{u,y}M_{x,w})$. Since this is a binary matrix, $1 + 1 = 0$. In this example we chose $M_{2,2}$ as pivot element. We then updated every entry according to the pivot procedure. As one can see, only $M_{1,1}, M_{1,3}$ and $M_{1,4}$ are changed. Afterwards we only need to swap the labels of the row and column of the pivot element.

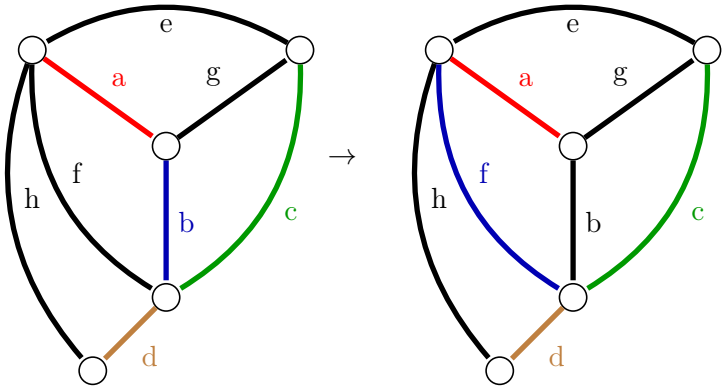


FIGURE 3: Example of a pivot in a graph corresponding to the matrix in Figure 2. By performing a pivot we can change the spanning tree T of a graph G, by swapping an edge $t \in T$ with an edge $e \notin T$. This can only be done if e forms a cycle with some spanning tree edges, including t. In this example we pivot $b \in T$ with $f \notin T$, which is the same pivot as in Figure 2

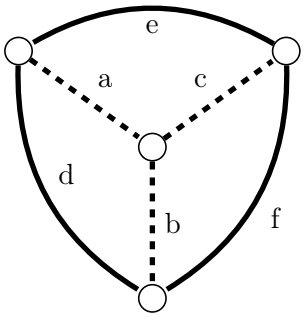
2.2 Graphic or not graphic?

Suppose we have a graphic representation matrix $M' = M(G)$, and a subset P of the rows of M', then by $M := (M'|P)$ we denote the matrix M' augmented by one binary column whose 1-entries are exactly those corresponding to P and let M be non-graphic. Given that M' is graphic, it has been proven that M is graphic if and only if P is a hypopath of G [2, The subrearrangement theorem]. Since we defined M to be non-graphic, P is not a path in G.

As the term non-graphic suggests, such a matrix cannot be represented by a graph. Since we want to exploit the fact that M' is graphic, we somehow want to visualize M. Therefore we will draw the spanning tree edges that are in P as dashed edges. An example can be seen in Figure 4.

Our algorithm heavily exploits the fact that M' is graphic and we thus have a graph G at hand. It would convenient for our algorithm if we could somehow reduce G such that it stays 3-connected and we still have a set of edges P that does not form a path that. For this we distinguish two reasons why a set P could be a non-path. Firstly, if we can identify a vertex that is incident with at least 3 edges of P, we know that it cannot be a path. Secondly, if the subgraph of G edge-induced by P is disconnected, we also know P is not a path. We will discuss these two cases in the next sections.

We will write the algorithm such that it will be performed on a graph. The pivots and



$$M = (M'|P) = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} d & e & f & P \end{array} \\ \left(\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right) \end{array}$$

FIGURE 4: On the left we visualized M, by drawing the graph corresponding to M' and marking the edges that are in P. On the right is the non-graphic matrix M.

contractions can in principle also be performed on the representation matrix of that graph, and will yield the same result if performed correctly. However finding out on which entries we should perform the operations can be quite tricky.

2.3 Minimal submatrix

In Section 1 we mentioned that we would like to obtain minimal non-graphic submatrices. We could define a minimal non-graphic submatrix to be any non-graphic matrix, which becomes graphic after deleting any row or column. There are however instances where such matrices are no longer minimal after performing a pivot, of which an example can be found in Section 3.1. For this reason, we decide to be more specific. Let B be the set of all matrix representations obtained from M by performing any number of pivots. If for every matrix N in B removing any row or column of N will make it graphic, then we say that M is minimally non-graphic.

It has readily been proven that a representation matrix is non-graphic if it contains one of the following submatrices: F_7, F_7*, $\mathcal{M}^*(K_5)$ or $\mathcal{M}^*(K_{3,3})$ [3]. All possible representations of the minors will be discussed in Section 5. This result makes constructing an algorithm much easier, as we will know we are done if we have obtained one of these matrices.

3 Case 1: A vertex incident with at least 3 P-edges

In the previous section we discussed two reasons how P could be not a path. In this section we will discuss the case where we have a vertex V which is incident with at least 3 edges of P.

We want to exploit the fact that G is 3-connected by reducing it to a wheel graph with v as central vertex. We want to achieve this as the $\mathcal{M}^*(K_{3,3})$ and F_7 submatrix can be represented by a matrix corresponding to a wheel graph, with 4 and 3 spokes respectively, augmented by a vector of 1s for all the spokes, see Section 5.

Some new definitions are used in the algorithm. Let $\text{dist}_T(u, v)$ be the length of the unique (u, v)-path in the spanning tree T. Let $G' := G - v$, $T' := T - v$ and let H be the graph induced from G' by contracting T'. Let $V(h)$ be the set of vertices of G that get contracted to h in H. These definitions stay consistent throughout the algorithm.

Algorithm 1

Input: 3-connected graph G with spanning tree T, set of edges P and vertex v incident with at least 3 P-edges.
Output: Sequence of pivots, contractions and deletions to be performed on G.
Note: Every operation performed on G or T should be stored

while $\exists e \in T \backslash P$ incident with v **do**
  Let $V(h)$ be connected to v by e and let f be the link between $V(h)$ and some $V(h')$ for some $h' \in H \backslash \{h\}$.
  $T = T \setminus \{e\} \cup \{f\}$
end while
if H is a forest **then**
  Let h be a leaf node of H and let h' be its unique neighbor in H
  while H is a forest **do**
    Let $e = \{u, w\}$ with $u \in V(h')$ and $w \in V(h'')$ for some $h'' \in H \backslash \{h'\}$ s.t. $\text{dist}_T(u, v)$ is maximal
    Let $f \in T$ be on the unique $\{u, v\}$-path in T s.t. f is incident to u
    Let $T = T \setminus \{f\} \cup \{e\}$
  end while
end if
Let W be the set of vertices in H which are in some cycle of H.
In G, delete all $V(h)$ for which $h \notin W$
In G, contract T' and delete unnecessary edges to form a wheel graph
while $d(v) > 4$ **do**
  In G, pivot an edge $t \in T$ with an adjacent edge $e \notin T$ and let $s \in T$ also be adjacent to e, $s \neq t$
  Contract e
  Pivot s with an adjacent edge $u \notin T$ and contract u
end while
Delete parallel edges
Return sequence of operations performed on G and T and the resulting subgraph

3.1 An example

We will first demonstrate an elaborate example to give a bit more insight into how the algorithm works.

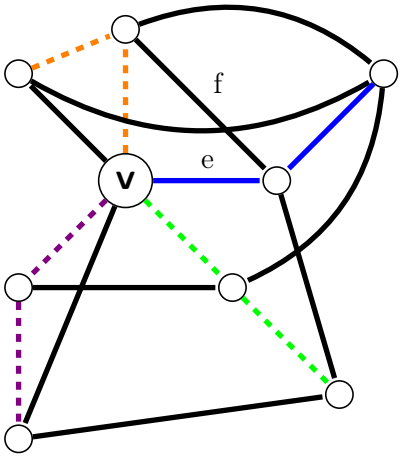


FIGURE 5: With the while loop on line 1-4, we want to make sure that all spanning tree edges incident with v are in P. We do this by pivoting a spanning tree edge e and a non-spanning tree edge f as defined in the algorithm. In this figure we have given the spanning tree edges of each $V(h)$ their own color and all the edges that are in P are dashed. We have also marked e and f for this step of the algorithm.

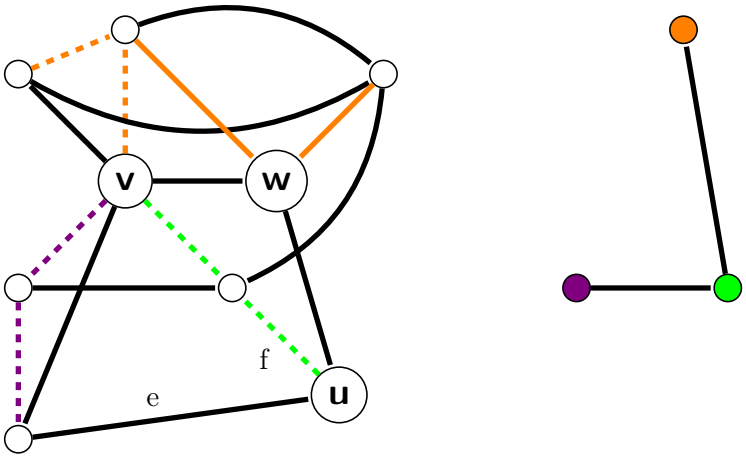


FIGURE 6: On the left is the resulting graph after the pivot of Figure 5. All spanning tree edges incident with v are also in P, so the first while loop terminates. Then, we only perform line 5-12 if H is a tree. On the right H is shown and is clearly a tree. We have given each h the same color as its corresponding $V(h)$ on the left. At this point we have two choices for h, namely the violet and the orange node. We chose the former and thus the green node is h'. Now we choose an edge $e = (u, w)$ such that $w \notin V(h')$ and $u \in V(h')$ is furthest away from v, to make sure that the following pivots do not make h' a leaf since this could cause an endless loop. Now by pivoting e with the given f, $V(h)$ increases in size and $V(h')$ decreases. We can repeat this process until h is no longer a leaf.

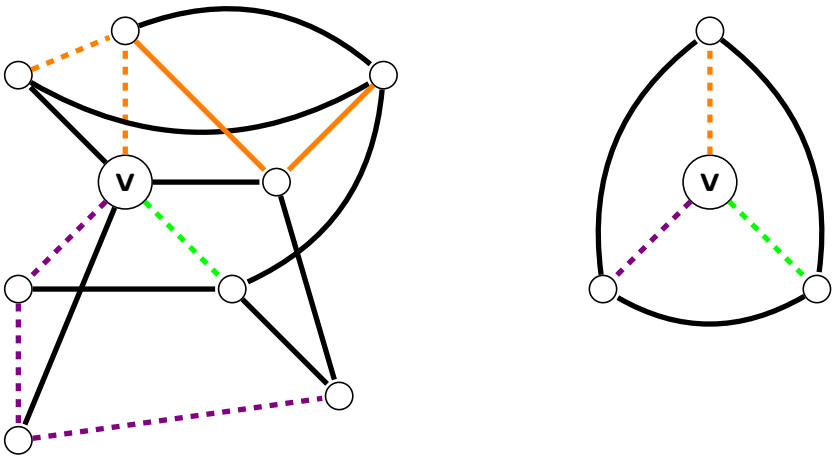


FIGURE 7: At this point H is no longer a tree and the while loop terminates. Next, we only want one cycle in H to continue with, so in line 13 and 14 we pick some cycle and delete all $V(h)$ for which h is not in the cycle we picked in H. In our example we only have one cycle and don't have to delete anything. In line 15, we contract T' in G to create a wheel of which all the spokes are in P. In our example, after deleting parallel edges, we are left with a wheel with 3 spokes which can be represented by the F_7 submatrix and therefore we are done. This can be seen on the right. Figure 8 shows what steps to perform if we are left with a wheel with more than 4 spokes at this point.

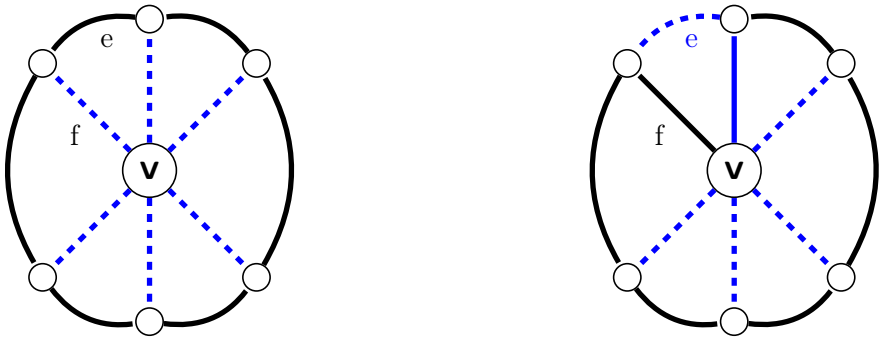


FIGURE 8: In the case we are left with a wheel with more than 4 spokes, we want to reduce it such that it has 3 or 4 spokes. With the while loop on lines 16-20, we pivot and contract one of the spokes. Notice how pivoting a spoke which is in P, causes another spoke to no longer be in P. See Figure 9 to see why that is.

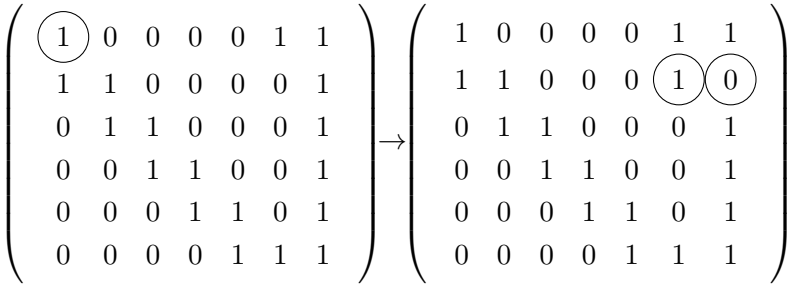


FIGURE 9: Matrix representation of Figure 8, which shows the effect of a pivot on a wheel graph with 6 spokes, which are all in P. The last column is P. In the first matrix the pivot element is highlighted, in the second the elements that changed. Notice how P loses one edge. Since the representation matrix of such a wheel graph, where all the spokes are in P, has 1s on the diagonal and the row below it, it is clear that occurs on a wheel graph of any size.

3.2 Theorem & Proof

This section is dedicated to Theorem 1 and its proof.

Theorem 1. *Given a 3-connected graph G, let $M' = M(G)$, let P be a subset of the rows of M' and let $M := (M'|P)$ be the matrix M' augmented by one binary column whose 1-entries are exactly those corresponding to P and let M be non-graphic. Then M can be reduced to a $\mathcal{M}^*(K_{3,3})$ or F_7 submatrix. Furthermore, Algorithm 1 can reduce G to G' and P to P' such that $(M(G')|P')$ also equals one of these matrices.*

Proof. Part 1: In line 2, f is well defined.

Since G is 3-connected, for every $h \in H$, $V(h)$ must have at least 3 vertex-disjoint paths leading to unique vertices. Since v is the only vertex not in $V(h)$ for any $h \in H$, and there exist at least 3 vertices $h \in H$, such an f must always exist.

Part 2: In line 3, $e \in T$ and $f \notin T$. Furthermore, after the pivot T is still a spanning tree.

It is clear why $e \in T$. The link between $V(h)$ and $V(h')$ for some $h, h' \in H$ cannot be a spanning tree edge, because they would contract to one node in H, thus $f \notin T$. Since f can only be pivoted with e if f closes a cycle of spanning tree edges which includes e, it is clear that $T \cup \{f\}$ contains a cycle, and $T - \{e\} \cup \{f\}$ is a spanning tree.

Part 3: The graph H is connected and h' is well-defined.

G' is at least 2-connected and contractions cannot disconnect a graph, and thus $H := G'/T$ is connected, hence it is a tree. We know that H has a least 3 vertices, so there must exist a leaf with a unique neighbor.

Part 4: In the while loop of line 7 - 11, H is always connected.

In this while-loop, the only operation performed is a pivot, which cannot disconnect G'. Consequently, H will stay connected as well.

Part 5: The while-loop of line 7 - 11 terminates

In H, h is a leaf with h' as unique neighbor. Consequently, $V(h)$ is only incident with $V(h')$ in G'. Since v is incident with at least 3 spanning tree edges in G, there must exist a $h'' \in H$ s.t. h' is incident with h'' in H and thus $V(h')$ is incident with $V(h'')$ in G'. Because G is 3-connected, G' is 2-connected, so there must exist at least 2 vertices in $V(h')$

which are adjacent to one or more vertices of $V(h)$. By the same account, there must be at least 2 vertices in $V(h')$ which are adjacent to one or more vertices some of $V(j)$, $j \in H$, $j \neq h$, $j \neq h'$. With every iteration $V(h')$ becomes smaller, as one or more edges are joined to $V(h'')$. In line 10, by pivoting only one vertex of $V(h')$ which is incident with a vertex of a neighbor in each iteration, we guarantee that $V(h')$ will never disappear and also that h will no longer be a leaf in H. Since G' is finite, at some point there will be no leaves in H and therefore there exists a cycle in H.

Part 6: After line 15, G is a wheel with $k = |W|$ spokes.

At this point in the algorithm, G consists only of v and $V(h)$ for every h that is in the chosen cycle in H. Since $H = G'/T$ is a cycle, G/T is a cycle of which every vertex is adjacent to v, thus we have a wheel with $k = |W|$ spokes.

Part 7: After each iteration of the while loop in lines 16-20, G has 2 spokes less

If the while loop is executed, G has more than 4 spokes and is thus not minimal, but since all the spanning tree edges are the spokes of G, we cannot delete or contract an edge without losing 3-connectedness. In line 17, by pivoting t with e, e can be contracted and G will still be 3-connected and P will still be a non-path. One consequence of this pivot is that s is no longer in T, see Figure 8. Therefore s can be pivoted with u and u can be contracted without changing P.

Part 8: M is reduced to a $\mathcal{M}^*(K_{3,3})$ or F_7 matrix

By the previous steps, it is clear that Algorithm 1 reduces G to be a wheel with 3 or 4 spokes, making sure that only the spokes are spanning tree edges and that P contains all spanning tree edges. Therefore $(M(G)|P)$ becomes either a $\mathcal{M}^*(K_{3,3})$ or F_7 matrix, and $M = (M(G)|P)$ thus also equals one of these matrices.

□

4 Case 2: Disconnected edges

In this section we discuss the case where P is not a path, due to the fact that its edges are not connected. By Tutte's wheel theorem, we know that any 3-connected graph can be obtained from the complete graph K_4 by performing a number of subdivisions [4]. Since contractions are the opposite of subdivisions, the converse is also true. Any 3-connected graph can be reduced to K_4. Edges that can be contracted such that the resulting graph is still 3-connected we define as contractible edges. It has been proven that any 3-connected graph with $p \geq 5$ vertices has at least $\lceil p/2 \rceil$ contractible edges [1].

For constructing an algorithm we also need to make sure that P remains a non-path. It could happen that all contractible edges either remove one of the two remaining P edges, or make P a path. An example can be seen in figure 10. This example shows that we cannot always reduce to an F_7 matrix.

An issue with constructing an algorithm for this case is identifying which edges are contractible. Most of the theory that exists only mentions in which cases contractible edges exist, but not how to find them. One possibility is to contract an edge and then running an algorithm to check for 3-connectivity, which is not an elegant solution. Since we would like to write an algorithm that makes use of certain structural properties, further research is required.

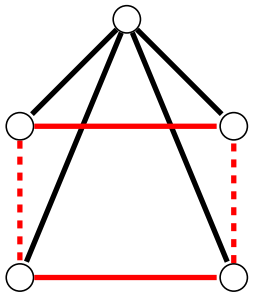


FIGURE 10: Graph represented by $\mathcal{M}^*(K_{3,3})$ with P edges dashed. The contractible edges are marked in red. Notice how contracting any edge will either cause P to be a non-path, or will cause the graph to no longer be 3-connected.

5 Pivots

In an earlier section we mentioned that a representation matrix is non-graphic if it contains one of the following matrices: F_7, F_7*, $\mathcal{M}^*(K_5)$ or $\mathcal{M}^*(K_{3,3})$. However, depending on the spanning tree chosen, some of these matrices can look quite different. The goal of this chapter is to provide all possible representations of these submatrices. To change the spanning tree in a representation matrix, one can perform a pivot as we described in Section 2.1 to swap an edge from the spanning tree with a non-spanning tree edge.

Theorem 2. *Of the F_7, F_7*, $\mathcal{M}^*(K_5)$ and $\mathcal{M}^*(K_{3,3})$ matrices, the only representations up to row and column permutations are shown in Figures 11-14.*

Proof. By checking every possible pivot on these matrices, it can be verified that these are the only possible representations up to row and column permutations. We have given every pivot element a color corresponding to the matrix which is obtained after performing that pivot. □

$$\begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} d & e & f & g \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

FIGURE 11: The only representation matrix of F_7

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccc} e & f & g \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

FIGURE 12: The only representation matrix of F_7*

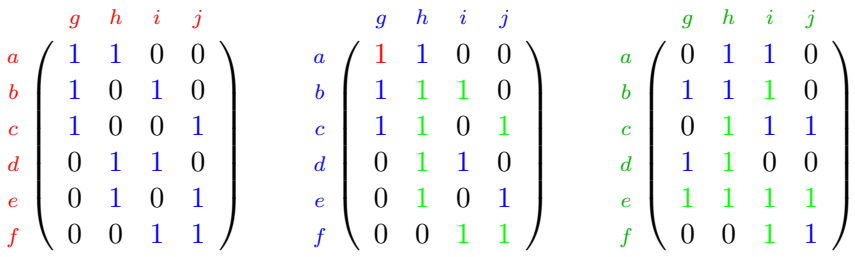


FIGURE 13: The $\mathcal{M}^*(K_5)$ matrix has 3 different representations. Each matrix is given a color and each 1 is given a color corresponding to which matrix is produced if we pivot that element. Red corresponds to the first matrix, blue to the second and green to the third.

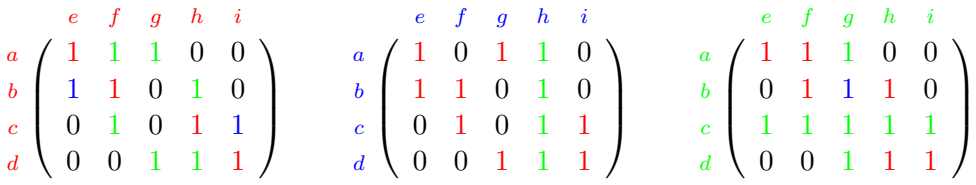


FIGURE 14: The $\mathcal{M}^*(K_{3,3})$ matrix has 3 different representations as well. The pivot elements are again given colors corresponding to the matrix which is obtained after performing that pivot.

6 Discussion

In this section we discuss some comments about previous sections.

In line 8 of Algorithm 1, the attentive reader might have noticed that $h = h''$ is a possibility. This is to prevent h' from becoming a leaf in H, which could cause an endless loop in some examples.

In line 13 of Algorithm 1, if we would choose the shortest possible cycle with the use of some BFS algorithm, we would not have to delete unnecessary edges in line 15.

As mentioned in Section 4, further research is required for constructing an algorithm that exploits certain structural properties. Due to the limited time available for a bachelor assignment and both cases being more complex than initially expected, constructing a proper algorithm for the second case was not possible. We could of course have constructed an algorithm where we perform some reduction and then check if the matrix is still non-graphic using an existing algorithm. Such an algorithm could probably run in polynomial time and would thus still be a feasible solution for this problem. However, this would not be a very interesting result. We have tried to find a way to convert graphs of the second case to graphs of the first case, but we haven't found a way to do so yet.

Overall, the results we obtained show that it is feasible to compute certificates for non-graphic matrices and also give some insight into the structure of the underlying graphs. The next logical step would be to finish the second case we mentioned, and after that combine the two cases to have one algorithm that can do the computations for both cases. Even though we mostly discussed properties of 3-connected graphs, the results might serve as a stepping stone for the case with 2-connected graphs.

7 Conclusions

In this paper we developed an algorithm to reduce non-graphic matrices with the use of 3-connected graphs represented by some submatrix. With the use of hypopaths we were able to split this large problem into two smaller cases, which both gave different structural insights. The output of the algortihm can be used as a certificate to verify non-graphicness and allows the user to have more insight as to why a given matrix is non-graphic. Future research would include writing one algorithm that covers the entire 3-connected case.

References

[1] Kiyoshi Ando, Hikoe Enomoto, and Akira Saito. Contractible edges in 3-connected graphs. *Journal of Combinatorial Theory, Series B*, 42(1):87–93, 1987.

[2] L. Lofgren. Irredundant and redundant boolean branch-networks. *IRE Transactions on Circuit Theory*, 6(5):158–175, 1959.

[3] W.T. Tutte. Matroids and graphs. 1957.

[4] W.T. Tutte. A theory of 3-connected graphs. 1961.

[5] Robert E. Bixby & Donald K. Wagner. *An Almost Linear-Time Algorithm for Graph Realization. MATHEMATICS OF OPERATIONS RESEARCH*, 13, 1988.

[6] Hassler Whitney. 2-isomorphic graphs. *American Journal of Mathematics*, 55(1):245–254, 1933.