UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

WOODALL'S CONJECTURE AND THE LUCCHESI-YOUNGER THEOREM

Autor: Oriol Moles Gené

Director: Dr. Kolja Knauer
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 24 de gener de 2022

Abstract

This project is about directed cuts, directed joins and their packings.

We study the open problem of Woodall's Conjecture, a problem studied by many authors to such an extent that it even has a $5000 dollars prize for his demonstration set by G. Cornuéjols. We will also cover the Lucchesi-Younger Theorem proof which can be seen as a dual result of the conjecture and the counterexample for the Edmonds-Giles Conjecture, the weighted version of Woodall's Conjecture.

Besides studying the theory we have set ourselves the goal of proposing a program that given a graph, checks if all of its orientations validate the Woodall's Conjecture. This program should be able to prove the conjecture, up to a certain number of vertices, by testing all the different combinations.

2020 Mathematics Subject Classification. 05C20, 05C70

Acknowledgments

First of all, I would like to thank my mentor in this project, doctor Kolja Knauer, for giving me the opportunity to work on this topic and for all the help he has given me throughout the semester.

I also would like to thank my family and friends for keeping up with me not only this last semester but throughout the whole degree, for giving me support when I have needed it, and to encourage me in the worst parts.

Contents

List of Figures

Chapter 1

Introduction to Graph Theory

In mathematics and more specifically in graph theory, a **graph** is a structure equivalent to a set of objects in which some pairs of them are in some sense "related". The objects correspond to mathematical abstractions called **vertices** (also called nodes or points) and the path between two related nodes is called an **edge** (also called link or line).

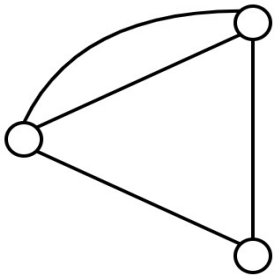


Figure 1.1: Graph with three nodes and four edges.

The following sections gather some basic results in graph theory, which can be found for more information in [23].

1.1 Basic Introduction

Definition 1.1. A **graph** G is mathematically defined as the tuple (V, E) where $V = V(G)$ is a set of elements called vertices, and $E = E(G)$ is a set of edges joining two vertices (not necessarily distinct) called its **endpoints**. A graph is **finite** if its vertex set and edge set are finite.

Definition 1.2. A **loop** is an edge whose endpoints are equal. **Multiple edges** are edges having the same pair of endpoints. A **simple graph** is a graph having no

loops or multiple edges. When two nodes are the endpoints of an edge, they are **adjacent** and **neighbors**.

For our work, we will restrict ourselves to graphs without loops since their existence will not affect our work.

Definition 1.3. A **subgraph** $S = (V', E')$ from the graph G consists of a set of vertices $V' \subseteq V$ and a set of edges $E' \subseteq E$ such that all edges in E' have both endpoints in V'.

A **path** is a simple graph whose vertices can be ordered so that two vertices are neighbors if and only if they are consecutive in the list. Mathematically we define a path as:

Definition 1.4. A **path** in a graph $G = (V, E)$ is a finite sequence of edges joining two vertices. Let n be an integer, we define a path P as $P = (v_1, e_1, v_2, e_2, ..., e_{n-1}, v_n)$ where $v_i \in V$ such that $e_i = (v_i, v_{i+1})$, $e_i \in E$, $i \in \mathbb{N}$ and all v_i are distinct.

Definition 1.5. A **cycle** is a path with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are neighbors if and only if they appear consecutively along the circle. In a cycle, the only nodes that are equal are the first and the last one. If a graph does not have cycles, we call them **acyclic graphs**.

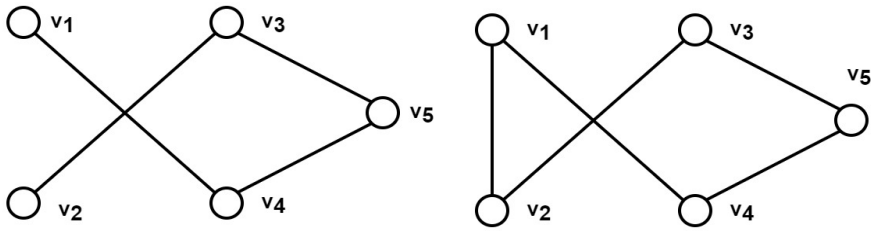


Figure 1.2: Path and cycle examples.

Definition 1.6. In graph theory, a graph G is said to be **connected** if G contains a path from every node to every other node. A graph that is not connected, is said to be **disconnected**.

Definition 1.7. A **connected component** in graph G is a subgraph S in which any two vertices are connected by a path and are not connected to any additional vertices from $G - S$.

For our work, we will restrict ourselves to graphs with one connected component. Having multiple connected components is solved by resolving each component individually.

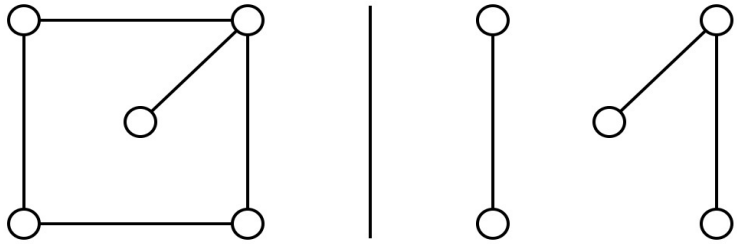


Figure 1.3: Connected graph with one connected component and disconnected graph with two connected components.

1.2 Directed Graphs

A **directed graph** or **digraph** is a graph in which edges have orientations, all the edges are directed from one vertex to another. We will define a digraph as:

Definition 1.8. A **directed graph** D is defined as the tuple $D = (V, A)$ where V is a set of vertices and A is a set of directed edges. We will refer to the edges of a directed graph as **arcs**.

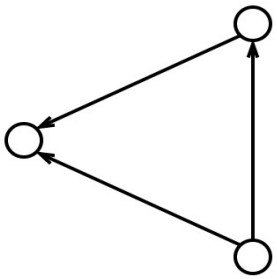


Figure 1.4: Directed graph with three vertices and three arcs.

Definition 1.9. A **weighted digraph** D^w is defined as the tuple (D, w) where $D = (V, A)$ is a digraph and $w : A \longrightarrow \mathbb{Z}^+$ is the weight function that assigns every arc $a \in A$ a weight that is denoted as $w(a)$.

Remark 1.10. Every digraph can be seen as a weighted digraph where $\forall a \in A$, $w(a) = 1$.

Definition 1.11. Given a digraph $D = (V, A)$. We define a **path** as we have defined it for undirected graphs, see definition 1.4, with the added property that the

edges are directed. Every arc in a path is in one of two directions. We differentiate between these two directions by arbitrarily calling one direction the **forward direction** and the other the **backward direction**.

Definition 1.12. Given a digraph $D = (V, A)$. We define a **directed path** as a path, see definition 1.11, with the added restriction that the arcs of the path have to all follow the same direction. We define a **directed cycle** as a cycle, see definition 1.5, with the same added restriction.

Definition 1.13. Given a digraph $D = (V, A)$ and $u, v \in V$. We say v is **reachable** from u if there exists a directed path going from node u to node v.

Definition 1.14. Given a digraph $D = (V, A)$. We call G the **underlying graph** of D, the graph resulting from replacing each arc with an undirected edge.

Definition 1.15. Given a graph $G = (V, E)$ we say that an **orientation** of G is a digraph $D = (V, A)$ such that $E = \{\{u, v\} \mid (u, v) \in A \text{ or } (v, u) \in A\}$

Definition 1.16. A **directed acyclic graph** or **DAG** is a directed graph with no directed cycles, such that it can be ordered by arranging the vertices as a linear sequence that is consistent with all the arcs directions.

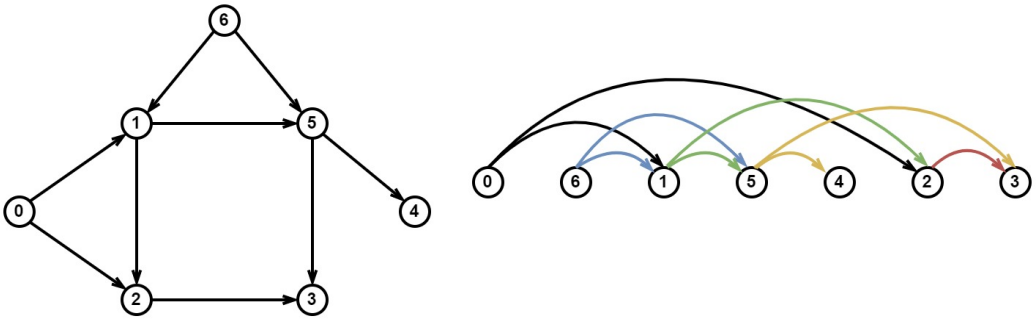


Figure 1.5: Directed acyclic graph with one of his linear sequences.

Definition 1.17. We say a digraph D is **connected** if the underlying graph G is connected, see definition 1.6.

Definition 1.18. A **strongly connected graph** is a directed graph in which there is a directed path from every vertex to every other vertex.

1.3 Dicuts, Dijoins and Packings

The first three definitions in this section also work for directed graphs since in definition 1.17, we have seen that the connectivity of a graph works equally in undirected and directed graphs.

Definition 1.19. An **edge-cut** in a connected graph $G = (V, E)$ is a subset $E' \subset E$ such that $G - E'$ is not connected.

Definition 1.20. A **trivial edge-cut** is a cut that separates a node from the rest of the graph.

Definition 1.21. E' is a **minimal edge-cut** if $\forall F \subsetneq E'$, $G - F$ is connected and $G - E'$ is not connected.

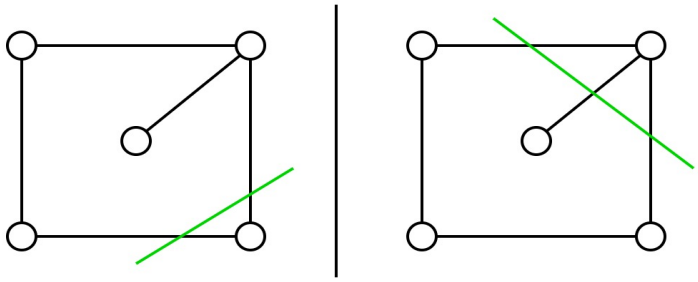


Figure 1.6: Minimal edge-cut and not minimal edge-cut. The edge-cuts contain the edges intersected by the green line.

Lemma 1.22. *E' is a minimal edge-cut $\Leftrightarrow$ $E - E'$ has exactly two connected components C_1, C_2, and all edges of E' have one vertex in C_1 and one in C_2.*

Proof. ($\Rightarrow$) Suppose that E' is a minimal edge cut such that $G - E'$ has more than two connected components.

Call G' the graph obtained after removing E' from G. Put back any edge $e \in E'$ in G'. The two endpoints of e belong to at most two of the components, so adding e can only reduce the number of components of G' by one. In other words, $G' + e$ is still disconnected, which implies that E' was not minimal as supposed so if E' is a minimal edge-cut, $G - E'$ has two connected components C_1, C_2.

Now we have to prove that all edges of E' have one vertex in C_1 and the other in C_2. Suppose there is an edge e that has both endpoints in C_1, then $G - (E' - e)$ is

still disconnected which means E' was not minimal. Contradiction, which means all edges from E' have one vertex in each component.

($\Leftarrow$) Suppose we have the graph G split into two connected components C_1, C_2. All the vertices going from one connected component to the other connected component form an edge-cut E'. Now we want to see if E' is minimal. Pick an arbitrary edge $e \in E'$ and consider $G - (E' - e)$. Since e has a vertex in C_1 and a vertex in C_2 there exists a path from any vertex in C_1 to any other vertex in C_2. This means that $G - (E' - e)$ is connected and E' is a minimal edge-cut. □

It has been proven that if E' is a minimal edge-cut, then we can divide $G - E'$ into two connect components C_1, C_2 where the edges of E' are the ones that connect the nodes from C_1 with the nodes of C_2. With this result, we can introduce, for directed graphs, the following definition:

Definition 1.23. If $D = (V, A)$ is a directed graph, we say a minimal edge-cut A' is **directed** if all the arcs connecting C_1 with C_2 follow the same direction, being C_1 and C_2 the two connected components that result from $A - A'$. From now on we will refer to minimal directed edge-cuts as **dicuts** or directed cuts for abbreviation purposes.

For our work, we will restrict ourselves to minimal directed cuts since it will allow us to be more precise and accurate with the number of sets we have to check.

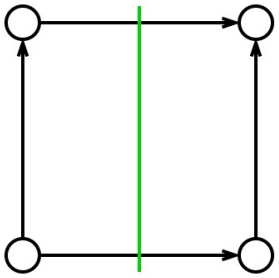


Figure 1.7: Minimal directed edge-cut.

Definition 1.24. A dicut divides a digraph into two disjoint sets called **source** and **sink**. For any subset X of V, we will denote as $\nabla^+(X)$ the set of arcs that leave X and as $\nabla^-(X)$ the set of arcs that enter X.

Definition 1.25. A **source-set** is any proper nonempty subset X of $V(G)$ such that $\nabla^-(X) = \emptyset$ and a **sink-set** is defined as any proper nonempty subset X of $V(G)$ such that $\nabla^+(X) = \emptyset$.

Definition 1.26. A **dijoin** in a digraph is a set of edges that intersect every minimal directed cut.

Definition 1.27. We define a **packing** as a set of mutually disjoint sets.

In the following chapters we will constantly use the definition packing of dijoins or packing of dicuts. From now on we will use the following notation:

- $\nu(D) = \max \{ |B| : B \text{ is a packing of dijoins of the digraph } D \}$
- $\nu^*(D) = \max \{ |C| : B \text{ is a packing of dicuts of the digraph } D \}$
- $\tau(D) = \min \{ |F| : F \text{ is a dicut of the digraph } D \}$
- $\tau^*(D) = \min \{ |G| : G \text{ is a dijoin of the digraph } D \}$

1.4 Bridges and Ears

Definition 1.28. Given a connected graph G we say an edge is a **bridge** if his deletion converts the connected graph into a disconnected one.

Definition 1.29. A connected graph $G = (V, E)$ is **k-edge-connected** if for every edge set $\{e_1, ..., e_{k-1}\} \in E$, $G - \{e_1, ..., e_{k-1}\}$ is connected (i.e. the removal of $k-1$ edges does not disconnect the graph).

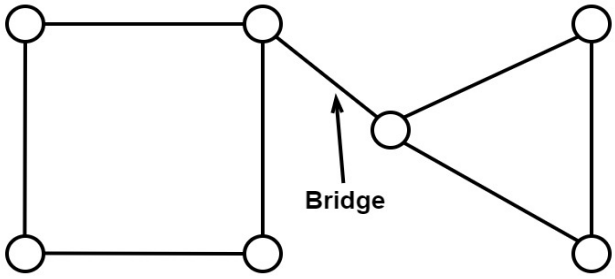


Figure 1.8: Example of a 1-edge-connected graph and a bridge.

Proposition 1.30. *A graph G is 2-edge-connected $\Leftrightarrow$ G has no bridges.*

Proof. ($\Rightarrow$) If the graph is 2-edge-connected, then the removal of an edge does not disconnect the graph which means there are no bridges.
($\Leftarrow$) If the graph does not have bridges it means there are no edges which his removal would disconnect the graph. Then, the graph is 2-edge-connected. □

Definition 1.31. An **ear** of a graph $G = (D, E)$ is a maximal path whose internal vertices have degree 2. An **ear decomposition** of G is a decomposition $P_0, P_1, ..., P_k$ with $k \in \mathbb{N}$ such that P_0 is a cycle and P_i for $i \geq 1$ is an ear of $P_0 \cup ... \cup P_{i-1}$.

Definition 1.32. A **closed-ear** in a graph G is a cycle C such that all vertices of C except one have degree 2 in G. A **closed-ear decomposition** of G is a decomposition $P_0, ..., P_k$ with $k \in \mathbb{N}$ such that P_0 is a cycle and P_i for $i > 1$ is either an (open) ear or a closed-ear in G.

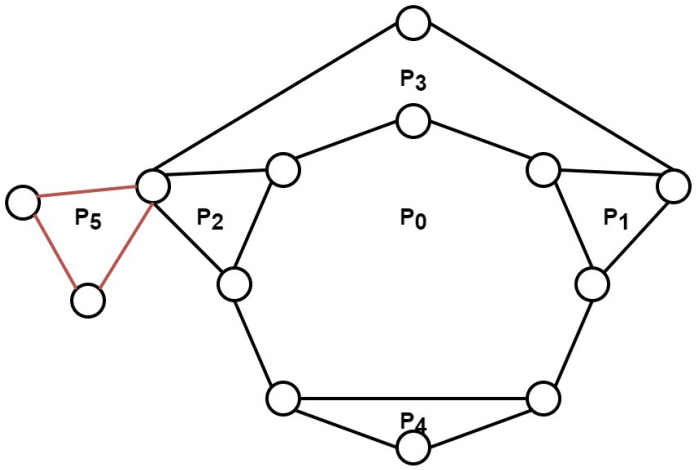


Figure 1.9: Ear decomposition where the black circles are (open) ears and the red circle is a closed-ear decomposition.

Theorem 1.33. *A graph is 2-edge-connected $\Leftrightarrow$ it has a closed-ear decomposition, and every cycle in a 2-edge-connected graph is the initial cycle in some such decomposition.*

Proof. ($\Leftarrow$) Clearly, a bridge does not lie on a cycle, so if every edge lies on a cycle then the graph is 2-edge-connected if and only if every edge lies on a cycle. The initial cycle is 2-edge-connected. When we add a closed-ear, its edges form a cycle. When we add an open ear P to a connected graph G, a path in G connecting the endpoints of P completes a cycle containing all edges of P. In each case, the new graph also is connected. Thus, adding an open or closed-ear preserves the property of 2-edge-connected.

($\Rightarrow$) Given a graph $G = (V, E)$ that is 2-edge-connected, let P_0 be a cycle. Consider a closed-ear decomposition $P_0, ..., P_i$ of a subgraph $G_i = (V_i, E_i)$ of G. When $G_i \neq G$, we find an ear to add. Since G is connected, there is an edge

$uv \in E - E(G_i)$ with $u \in V(G_i)$. Since G is 2-edge-connected, uv lies on a cycle C. Follow C until it returns to $V(G_i)$, forming up to this point a path or cycle P. Adding P to G, yields a larger subgraph G_{i+1} in which P is an open or closed ear. The process ends only by absorbing all of G.

□

1.5 Overview TFG

Once the preliminary part has been completed we can introduce the key statements that we will present in the following chapters.

In chapter two we will talk about the Lucchesi-Younger Theorem (theorem 2.7) which states that the size of the smallest dijoin is equal to the maximum number of disjoint dicuts (i.e. $\tau^* = \nu^*$). First, in section 2.2, we will go through all the preliminary results needed for the proof and then in section 2.3 we will prove the theorem.

After Lucchesi-Younger, in chapter three, we will introduce Woodall's Conjecture (conjecture 3.4) which states that the size of the smallest directed cut is equal to the maximum number of disjoint dijoins (i.e. $\tau = \nu$). Before finishing Woodall's chapter, we will present the proof for the first two cases and mention some partial results in section 3.3.

Once the key statements have been introduced, in chapter four we will propose a pseudo-code to prove, up to a certain number of nodes, the conjecture. In section 4.4, we will present all the key elements gathered through the project that will help us reduce the amount of work needed to do by the program.

Finally, in chapter five we will present the Edmonds-Giles Conjecture which states that the minimum weight of a dicut is equal to the maximum number of dijoins. In section 5.1, we will see a counterexample given by Schrijver that disproves the conjecture.

1.6 First Observations

In this section we will present the first observations we can do about the topics we will target in this project.

Proposition 1.34. *Given a digraph $D = (V, A)$ every arc of the digraph is either in a dicut or in a directed cycle.*

Proof. First, we will prove that dicuts do not have arcs in directed cycles and then we will prove that all arcs not in directed cycles appear on dicuts.

Since in directed cycles there is a path from every node to every other node, if a dicut would have an arc from it, let us say $a \in A$ with endpoints $u, v \in V$ and direction u to v, it would be impossible to separate the graph into two connected components C_u, C_v because the direction of the dicut would be from C_u to C_v but there would exist a path going from v to u which would mean an arc going from C_v to C_u.

Now let us take any arc from the digraph that does not belong to a directed cycle, $b \in D$. Suppose the endpoints of b are x, y and the direction is x to y. Since b does not live in a directed cycle, there is not a path from y to x which lets us separate the graph into two connected components C_y holding all the nodes reachable from node y and C_x with all the rest of the nodes. Finally we have two connected components and an arc-set formed by the arcs connecting C_x to C_y, a directed cut containing b. □

As a result of this proposition, we can extract the following statement:

Remark 1.35. Strongly connected graphs do not have dicuts.

Proof. Since in a strongly connected graph, every node is reachable from every other node, all nodes are inside directed cycles. Using proposition 1.34, it is clear that there are no dicuts. □

Lemma 1.36. *Given $D = (V, A)$ a digraph and $a \in A$, D/a does not have dicuts that were not already in D.*

Proof. Given a digraph $D = (V, A)$ and an arc $a \in D$ with endpoints $u, v \in V$. Suppose we have found all his dicuts and now, contract the arc resulting in D/a, the same graph as D but with all arcs incident to the nodes u, v collapsed into just one node, let us say w. Now suppose we have a new dicut F with at least one arc incident to w since it is the part of the graph that has changed.

Now the goal is to prove that this new dicut in D/a is also a dicut in D. As we have seen in 1.22 a dicut divides a graph into two connected components C_1, C_2 with all arcs in F being the arcs that connect the nodes in C_1 to the nodes in C_2.

Now suppose that $w \in C_2$, when we did the contraction, we did not remove any arc besides the one joining u, v so, by removing the contraction and adding u, v, a to C_2 we still manage to maintain F as a dicut in D. □

Theorem 1.37. *Let $D = (V, A)$ be a digraph and let $D' = (V', A')$ be the acyclic digraph obtained by contracting each strongly connected component to a single vertex, then:*

$$\tau(D) = \tau(D') \quad , \quad \tau^*(D) = \tau^*(D') \quad , \quad \nu(D) = \nu(D') \quad , \quad \nu^*(D) = \nu^*(D').$$

Proof. To prove this theorem we will first start by showing that all dicuts from D are also in D', and then we will prove the reverse statement.

By lemma 1.34 and remark 1.35 every arc contraction, where the arc belongs to a strongly connected component from D is in no dicut. Therefore any dicut A in D is also a dicut in D'.

By lemma 1.36 we have that every dicut A' from D' is a dicut of D in particular, every arc of D' is in a dicut, so D' has no directed cycles, which means D' is acyclic. Hence, both sets are the same, so also the sets of the dijoins are the same. This also means that the possible packings of dicuts or dijoins are equal. Thus,

$$\tau(D) = \tau(D') \quad , \quad \tau^*(D) = \tau^*(D') \quad , \quad \nu(D) = \nu(D') \quad , \quad \nu^*(D) = \nu^*(D').$$

□

With this result, we can say, with no loss of generality, that it will be enough to prove Lucchesi-Younger and Woodall for directed acyclic graphs since transforming our graphs by converging in just one node each strongly connected component, does not affect the metrics we are studying.

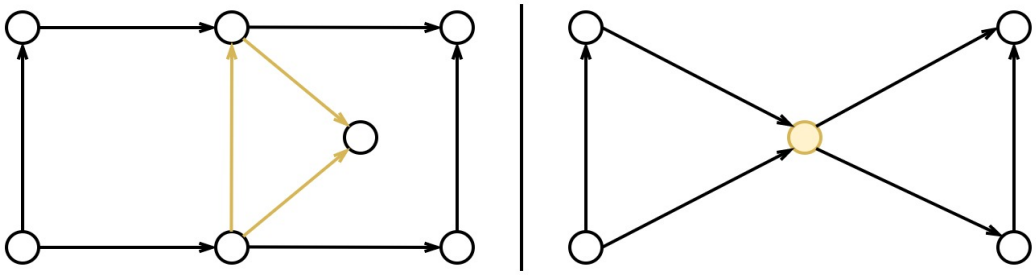
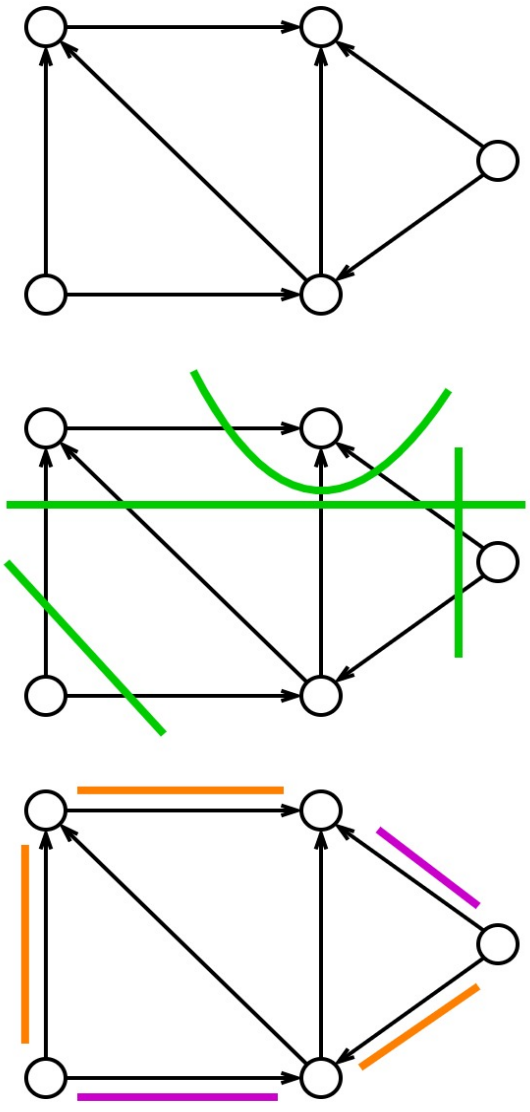


Figure 1.10: Before and after transformation of a digraph.

1.7 Example

Before starting with the main statements, we will do a quick exercise to visualize the main definitions given at the beginning of this chapter. The goal is to find dicuts, dijoins and then extract the metrics needed to check if the digraph fulfills Lucchesi-Younger Theorem and Woodall's Conjecture.

Given the following directed graph, we will find the directed cuts (in color green) and a couple of disjoint dijoins (in color orange and purple):



As a result, we can see that this digraph has:

- The smallest dicut size is two ($\tau(D) = 2$).
- The smallest dijoin size is two ($\tau^*(D) = 2$).
- The maximum number of disjoint dicuts is two ($\nu^*(D) = 2$).
- The maximum number of disjoint dijoins is two ($\nu(D) = 2$).

So, we have seen that this digraph fulfills Lucchesi-Younger Theorem since the number of the smallest dijoin is equal to the maximum number of disjoint dicut ($\tau^* = \nu^*$). It also fulfills Woodall's Conjecture since the number of disjoint dijoins is equal to the size of the smallest dicut ($\tau = \nu$).

Chapter 2

Lucchesi-Younger Theorem

Lucchesi and Younger theorem states that if D is a directed graph with the smallest dijoin of size k then D has k disjoint dicuts, [13].

Before starting this chapter it is important that we understand the authority of the proof and the history behind it.

The paper where Lucchesi and Younger published the proof of the theorem is a revised version of part of the first-named author's Ph.D. thesis [12], done at the University of Waterloo under the second-name author's supervision. As Lucchesi and Younger wrote in their paper, Lovász obtained independently proof of the directed cut minimax equality [11] and in this paper he acknowledges the priority of Lucchesi and Younger original proof and also acknowledges D. H. Younger for his valuable remarks. Lucchesi and Younger also added in their paper that Lovász proof influenced their version.

2.1 Upper Bound

Let us first start showing that the result of Lucchesi-Younger Theorem is best-possible in the following sense:

Theorem 2.1. *If D is a directed graph with the smallest dijoin of size k then D has (at most) k disjoint dicuts.*

Proof. If the minimal size of a dijoin is k, it means that we can cross all the directed cuts with just k edges. This means that there are at most k disjoint dicuts, because if there were $k+1$ disjoint dicuts we would need $k+1$ edges to intersect all the dicuts. Which results in the smallest dijoin formed by $k+1$ edges. □

2.2 Preliminary Results for the Proof

In this section we will present all the preliminary results and information needed for the proof, see [1].

Definition 2.2. Given a digraph $D = (V, A)$, let V be a finite set, and let $\mathcal{S}$ be a family of subsets of V, where some subsets may be equal. We say that two sets, $S_1, S_2 \in \mathcal{S}$ are **crossing** if the four sets $S_1 \cap S_2$, $S_1 - S_2$, $S_2 - S_1$, $V - (S_1 \cup S_2)$ are nonempty. We say that $\mathcal{S}$ is **cross-free** if it has no crossing sets.

Definition 2.3. Let $D = (V, A)$ be a digraph. Take a nonempty subset U of V. We say that the cut $\nabla^+(U)$ is a dicut if $\nabla^-(U) = \emptyset$ (i.e. $\nabla^+(U)$ is a dicut if it has no incoming arc). In this case we will refer to U as an **out-shore** of $\nabla^+(U)$.

Lemma 2.4. (Dicut coloring lemma) *Let $D = (V, A)$ be a digraph, and $\mathcal{F}$ a family of (possibly equal) dicuts whose out-shores form a cross-free family. Take an integer $k \geq 1$. If every arc appears in at most k dicuts of $\mathcal{F}$, then the dicuts of $\mathcal{F}$ can be k-colored so that dicuts of the same color are arc-disjoint.*

Proof. We will not prove this lemma since it requires to have a knowledge of various topics that defer from the general objects we are handling in this work. □

Definition 2.5. Let $D = (V, A)$ be a digraph. An arc is **essential** if it is used in every maximum packing of dicuts.

Proposition 2.6. *Let $D = (V, A)$ be a digraph, then D has an essential arc.*

Proof. We want to prove that given ν^* (the maximum size of a packing of dicuts), there exists at least an edge e that is essential.

Suppose otherwise. Then, for each arc we have a packing of ν^* disjoint dicuts of D excluding the arc. Doing this for every arc of D, we get a family $\mathcal{F}$ such that:

> $(*)$ $\mathcal{F}$ is a family of dicuts of D such that $|\mathcal{F}| = |A| \cdot \nu^*$, and every arc of D is used in at most $|A| - 1$ dicuts of $\mathcal{F}$.

We will recursively update the family $\mathcal{F}$ so that each intermediate family satisfies $(*)$ and, in the end, the outshores form a cross-free family.

If the out-shores of $\mathcal{F}$ form a cross-free family, then we are done.

Otherwise, take dicuts $\nabla^+(U_1)$, $\nabla^+(U_2) \in \mathcal{F}$ where U_1, U_2 are crossing (i.e. $U_1 \cap U_2 \neq \emptyset$). Then $\nabla^+(U_1 \cap U_2)$, $\nabla^+(U_1 \cup U_2)$ are also dicuts such that:

$$\nabla^+(U_1 \cap U_2) \cap \nabla^+(U_1 \cup U_2) \subseteq \nabla^+(U_1) \cap \nabla^+(U_2)$$

$$\nabla^+(U_1 \cap U_2) \cup \nabla^+(U_1 \cup U_2) \subseteq \nabla^+(U_1) \cup \nabla^+(U_2).$$

We update $\mathcal{F}$ by replacing the dicuts $\nabla^+(U_1)$, $\nabla^+(U_2)$ with the dicuts $\nabla^+(U_1 \cap U_2)$, $\nabla^+(U_1 \cup U_2)$. The inclusions above imply that $\mathcal{F}$ still satisfies $(*)$.

Since at each iteration, the potential $\sum_{\nabla^+(U) \in F} |U|^2$ strictly increases:

$$|U|^2 + |V|^2 < (|U| + |V| - |U \cap V|)^2 + |U \cap V|^2$$

$$\text{then, } \tfrac{|U| \cdot |V|}{|U \cap V|} + |U \cap V| > |U| + |V|$$

so, knowing that U, V are crossing, $|U| \geq |V| > |U \cap V|$, up to exchange between U, V, which proves the inequation and then, we can state that we will eventually reach a family $\mathcal{F}$ satisfying $(*)$ whose out-shores form a cross-free family.

Therefore, by the dicut coloring lemma 2.4, we may $(|A| - 1)$-color the dicuts of $\mathcal{F}$ so that each color class is a packing of dicuts. One of the color classes has cardinality at least $\frac{|A| \cdot \nu^*}{|A|-1} > \nu^*$, implying in turn that D has a packing of $\nu^* + 1$ dicuts. Contradiction. Thus, D has an essential arc. □

2.3 Proof of the Theorem

Once we have all the theory needed, we can finally prove the theorem. The proof we present is due to Lovász 1976 [11].

Theorem 2.7. (Lucchesi and Younger, 1978) [13] *If D is a directed graph with the smallest dijoin of size k then D has (at least) k disjoint dicuts.*

Proof. Let $D = (V, A)$ be a digraph. We will prove by induction on $|A| \geq 1$ that the clutter of dicuts packs.

The base case $|A| = 1$ is trivial. One edge means one dicut of size one, which results in one dijoin of size one and only one disjoint dicut.

For the induction step, assume that $|A| \geq 2$. We may assume that the underlying undirected graph of D is connected and that D is not strongly connected, see remark 1.35.

Let ν^* be the maximum size of a packing of dicuts. Let e be an essential arc of D, see proposition 2.6, and let $C_1, \ldots, C_{\nu^*}$ be a maximum packing of dicuts such that $e \in C_{\nu^*}$. To complete the induction step and prove the theorem, it suffices to exhibit a dijoin of cardinality ν^*.

As e is essential, the dicuts $C_1, \ldots, C_{\nu^*-1}$ give a maximum packing of dicuts of D/e since, after the contraction, new dicuts do not appear as seen in lemma 1.36. Thus, by the induction hypothesis, D/e has a dijoin B' of cardinality $\nu^* - 1$. Notice that $B' \cup e$ is a dijoin of D of cardinality $(\nu^* - 1) + 1$, as required since e is in a disjoint dicut of the set $C_1, \ldots, C_{\nu^*-1}$. This completes the induction step. □

Chapter 3

Woodall's Conjecture

Woodall's Conjecture states that if D is a directed graph with the smallest nonempty directed cut of size k, then D has k disjoint dijoins, [25]. Woodall introduced the conjecture in 1978 and to this day, is still an open problem.

3.1 Upper Bound

Let us first start showing a modification of Woodall's Conjecture. It is easy to see that the conjecture is best-possible in the following sense:

Theorem 3.1. *If D is a directed graph with the smallest directed cut of size k, then D has (at most) k disjoint dijoins.*

Proof. If the smallest dicut has size k it means that, concerning that minimal directed cut, we will only be able to construct k disjoint dijoins, the $k+1$ dijoin will have to share an edge with a previous dijoin because there are only k unique options. □

3.2 Small τ

Even though the proof of the conjecture is still an open problem to this date, we can prove the conjecture for $k = 1$ and $k = 2$ but to do so, we need some previous results:

Proposition 3.2. *A 2-edge-connected graph G has at least one cycle C.*

Proof. Let uv be an edge of G. Since the graph is 2-edge-connected we can remove this edge and the graph will remain connected, which means there is still a path

going from node u to node v. Adding to this path the edge uv, results in a cycle C in G. □

Theorem 3.3. (Robbins, 1939) [19] *If G is a 2-edge-connected graph there exists an orientation that creates a strongly connected digraph.*

Proof. Previously we have seen that a 2-edge-connected graph has an ear decomposition (see theorem 1.33) and at least one cycle (see proposition 3.2), so start by taking any cycle C_1 in the graph and orientating it so all edges point in the same direction along the cycle. If C_1 includes every vertex, then we are done.

Otherwise, find some vertex v that is not on C_1 but is adjacent to a vertex u on C_1. We know v exists since the graph is connected. The edge uv must be part of some cycle C_2. Starting at v, we trace along C_2, orienting its edges to all point in the same direction until we reach a vertex on C_1, which could be u or another one. Notice that it is now possible to get from any vertex on $C_1 \cup C_2$ to any other by tracing along C_1 and C_2.

From here we continue this process by finding another vertex w that is not on C_1 or C_2 but is adjacent to a vertex of $C_1 \cup C_2$. It is part of a cycle, C_3 and we trace along C_3, orienting edges consistently in the same direction until we reach a vertex of $C_1 \cup C_2$. Continuing adding vertices following this methodology results in a strongly connected orientation. □

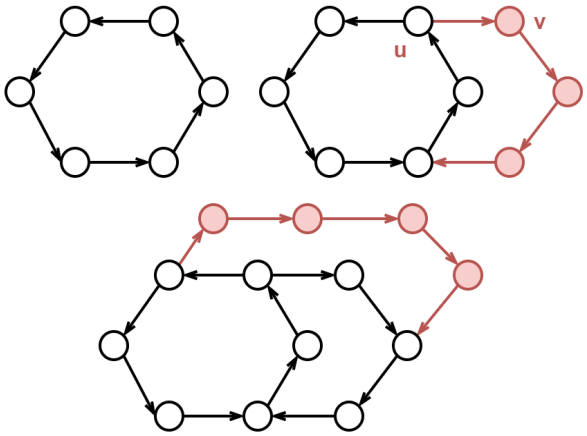


Figure 3.1: Graphic visualization of Theorem 3.3 proof.

Now we can introduce the proof for the first two cases:

Theorem 3.4. (Woodall, 1978) [25] *If D is a directed graph with the smallest nonempty directed cut of size $k \leq 2$, then D has (at least) k disjoint dijoins.*

Proof. If $k \in \mathbb{Z}$ is the size of the smallest nonempty directed cut, then:

($k = 0$) Nothing to show.

($k = 1$) If the smallest directed cut has size $k = 1$, then we just need to create one dijoin. It can be constructed by selecting the edge intersected by the minimal directed cut and the rest of the edges picked randomly one from every directed cut still not intersected by the dijoin.

($k = 2$) To prove the statement for $k = 2$ we will use that the underlying graph G, the undirected graph, is 2-edge-connected. If this did not happen, it would mean that there is a bridge which means that we would be operating in a graph with the smallest non-empty directed cut of size $k = 1$, the directed cut cutting only the bridge.

So, assuming that the graph is 2-edge-connected we have seen that the graph G can be oriented to generate a strongly connected graph H, see theorem 3.3. This graph has zero directed cuts because there is always a path between any pair of nodes, see remark 1.35.

Now partition the edges in two different sets X, Y where X consists of those edges which have the same orientation in both H and D, and Y consists of those edges with different orientations between the two graphs. Immediately, both X and Y meet every directed cut, because all the directed cuts will have at least an edge keeping the orientation and one edge that does not, since in H it exists a path going from one side of the directed cut to the other side and vice-versa, so each is a dijoin.

□

3.3 Other Partial Results

Even though there is not a proof for the conjecture, some authors have proven the conjecture for graphs with specific properties.

Remark 3.5. Woodall's Conjecture holds on all digraphs $\Leftrightarrow$ Woodall's Conjecture holds on all acyclic digraphs.

Proof. See theorem 1.37. □

Definition 3.6. We define **sink-nodes** and **source-nodes** in a digraph as sink-sets and source-sets of one element respectively, see definition 1.25.

Schrijver in 1982 [21] and independently Feofiloff and Younger in 1987 [7] proved the conjecture for digraphs in which every sink-node is reachable from every source-node.

Definition 3.7. We say that a digraph is **series-parallel** if its underlying graph does not contain a subdivision of K_4.

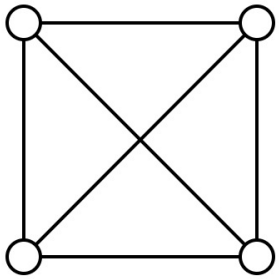


Figure 3.2: Graph K_4 example.

Lee and Wakabayashi in 2001 [10] showed that the conjecture is true for orientations **series-parallel**.

The recognition of series-parallel graphs can be performed with an algorithm that solves the problem in linear time. This algorithm was defined in 1982 by Valdes, Tarjan and Lawer, see [22].

Definition 3.8. A graph **partition** is the reduction of a graph to a smaller graph by partitioning its set of nodes into mutually exclusive groups. Edges of the original graph that cross between the groups will produce edges in the partitioned graph.

Definition 3.9. Let $G = (V, E)$ be an undirected graph. For an integer k, G is said to be **(k,1)-partition-connected** if for every non-trivial partition $\mathcal{P}$ of $V(G)$ there are at least $k(|\mathcal{P}| - 1) + 1$ edges in $E(G)$ that connect different members of $\mathcal{P}$.

Mészáros in 2018 [16] proved, using Olson's theorem [17], that if k is a prime power, then the conjecture holds if the underlying undirected graph is $(k-1, 1)$-partition-connected.

In 1980, a paper was published by András Frank with an algorithm to compute the $(k-1, 1)$-partition-connected property, see [9].

Chapter 4

Program

The goal of this chapter is to propose a program that proves if Woodall's Conjecture holds up to a certain number of nodes, using a computer.

To do so, we will need to check all the combinations of graphs and their orientations for a fixed number of nodes.

4.1 Initial Dataset

One of the first questions we have to ask ourselves is how we can narrow down the starting dataset to make the process as efficient as possible.

Definition 4.1. An **isomorphism** of graphs G and H is a bijection between the vertex sets of G and H, $f\colon V(G) \to V(H)$ such that any two vertices $u, v \in G$ are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. We say G, H are **isomorphic** and we denoted the relation as $G \cong H$.

Lemma 4.2. *Given two digraphs D, D' that are isomorphic ($D \cong D'$). Then, $\tau(D) = \tau(D')$ and $\nu(D) = \nu(D')$.*

Proof. D and D' are isomorphic, which means there exists a bijection, as we have defined in definition 4.1, that sends every dicut from D to a dicut in D' and vice-versa (i.e. $\tau(D) = \tau(D')$). We prove that $\nu(D) = \nu(D')$ analogously. □

So, with this result, we only have to generate all graphs with n vertices up to isomorphism which will be done using software that already exists, in this case, Nauty [14].

As we have seen in chapter one, we will also narrow down the initial dataset by restricting ourselves to minimal directed cuts, graphs with no loops, and graphs with only one connected component. We will also have to restrict our dataset to graphs without multiple edges because it would be impossible to have a finite dataset if multiple edges were allowed (we would always be able to add more edges between a pair of vertices).

4.2 Calculate τ and ν Values

The key part of the program is how we calculate the metrics τ and ν of a digraph.

Given a digraph D, no polynomial algorithm is known to compute $\nu(D)$ but there exists a polynomial algorithm (essentially the max-flow min-cut algorithm) to calculate $\tau(D)$.

Definition 4.3. A **network** is an acyclic weighted digraph $D^w = (D, w)$ with a distinguished source vertex s (source) and sink vertex t (target). In case there are many sources and sinks the network is restricted to the vertices that are reachable from s and reach t.

Definition 4.4. A **flow** is a function $f : A \longrightarrow \mathbb{Z}^+$ that assigns a value to each arc of a network with the constraint that $\forall a \in A$, $w(a) \geq f(a)$. We call **maximum flow** of a network the sum of the flows of the arcs that arrive at the sink node, $\sum_{a_i \in A} f(a_i)$ such that a_i are the arcs of the form ut with $u, t \in V$ where t is the target of the network.

Theorem 4.5. (Max-flow min-cut Ford Fulkerson, 1956) [8] *The maximum flow through any network from a given source to a given sink is exactly the minimum sum of the arc weights that, if removed, would disconnect the source from the sink.*

Flow can apply to anything. It could mean the amount of water that can pass through a network of pipes or it could mean the amount of data that can pass through a computer network.

In figure 4.1, we can see the flow that can pass through an arc divided by the weight of the arcs. Notice that the first and last arcs do not carry the maximum flow because the bottleneck is the part of the graph that fixes the maximum amount of flow the system can deal with.

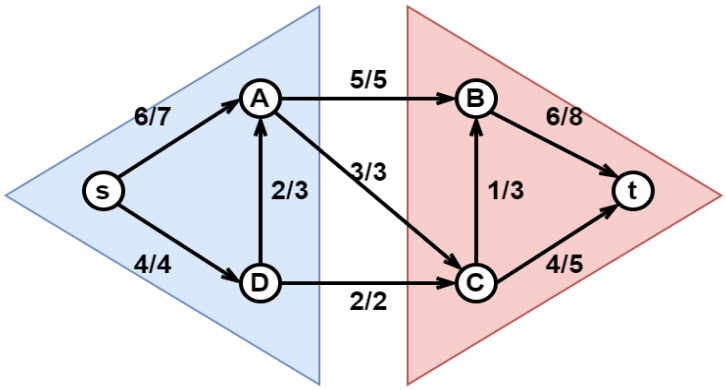


Figure 4.1: Max-flow min-cut example.

Corollary 4.6. *Given an acyclic weighted digraph* $D^w = ((V, A), w)$ *where* $\forall a \in A$, $w(a) = 1$. *The maximum flow is equal to the size of the minimal dicut that separates source* s *from the sink* t *(i.e. the size of* τ*).*

This algorithm can be used to find the size of the minimal dicut, only in acyclic digraphs with a unique source-node and a unique sink-node.

Lemma 4.7. *If* D *is a DAG, every dicut separates some source* s *from some sink* t*.*

Proof. Suppose X is a dicut that separates two connected components, C_1, C_2 with arcs going from C_1 to C_2. In particular, C_1, C_2 are DAG and every DAG has at least one source and one sink.

- Sink t from C_2 is also a sink from D because C_2 only has arcs entering.
- Source s from C_1 is also a source from D because C_1 only has arcs departing.

□

Then, using corollary 4.6 for every pair (s, t) where s is a source-node and t is a sink-node, and minimizing all the sizes of the dicuts obtained, we get τ. This adaptation of the max-flow min-cut algorithm allows us to calculate the value of τ for any DAG. It is important to mention that even with this change, the algorithm runs in polynomial time.

It is interesting to mention that before coming up with the previous idea, we explored other possibilities. The problem was how to deal with multiple sources

and multiple sinks so, trying to find an algorithm we came across an adaptation for graphs with multiple sources and multiple sinks that consisted on the creation of a **super-source node** and a **super-sink node**, that connect with all the sources and sinks respectively with arcs of infinite weight.

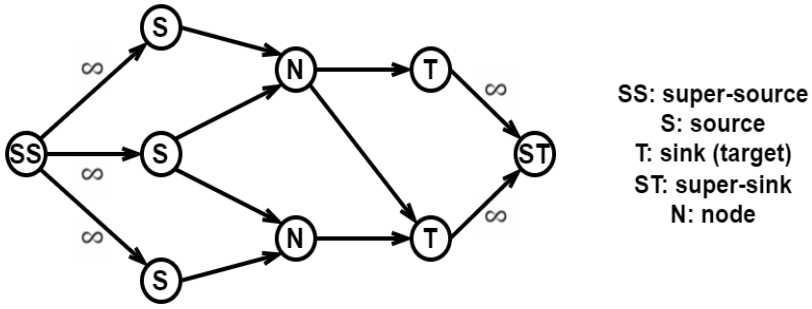


Figure 4.2: Max-flow min-cut for multiple sources and sinks.

The problem with this adaptation was that the algorithm was not targeting all the dicuts. Figure 4.3 shows two of the problems, in color green we can see the dicuts we want and in color blue the dicuts we would be getting. The output of this algorithm could only be seen as an upper bound of τ.

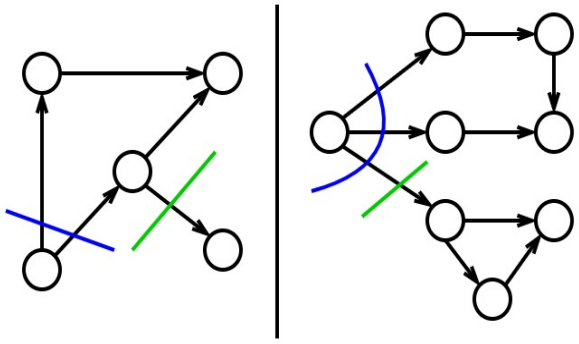


Figure 4.3: Max-flow min-cut dicuts results (in color blue) and the dicuts we are looking for (in color green).

To shorten the distance between what we want and what the algorithm returns we thought of picking, as the upper bound, the minimum between the sizes of the dicuts formed by trivial cuts and the result from the algorithm. This adaptation would create a more accurate upper bound that targeted the problem we have shown in the left part of figure 4.3 when the dicut is generated by a trivial cut.

4.3 Linear Programming Formulation

In 2005, Paulo Feofiloff [6] published a paper in which he presented an algorithm to calculate the τ and ν values, which we will explain below.

Lemma 4.8. *Given a digraph $D = (V, A)$, every set of arcs $A' \in A$ that intersects all dijoins includes a dicut.*

Proof. Suppose we have a set of arcs A' that intersects all dijoins. Now suppose that A' does not include a dicut and we will see that this is a contradiction.

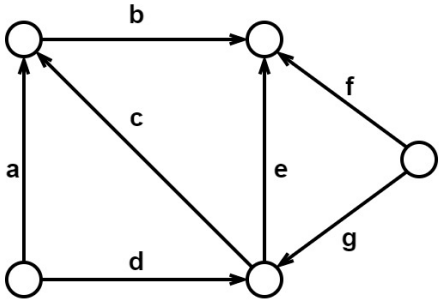
Supposing that A' does not include a dicut is the same as supposing that for every dicut D_i , there is an arc $a_j \in D_i$ such that $a_j \notin A'$ (since there is not a complete dicut inside A'). Hence, it exists a dijoin formed by the arcs a_j that is not intersected by A'. Contradiction. □

Every set of arcs that intersects all the dicuts is a dijoin but also conversely as we have just seen in lemma 4.8. Therefore, given a digraph D, $\nu(D)$ and $\tau(D)$ may be defined by the following pair of integer linear programs:

$$\nu(D) = \{ \text{ maximize } y1 \text{ subject to the constraints } y \in \{0,1\}^J \text{ and } yM \leq 1\}$$
$$\tau(D) = \{ \text{ minimize } 1x \text{ subject to the constraints } x \in \{0,1\}^E \text{ and } Mx \geq 1\}$$

where J is the set of all (inclusion wise minimal) dijoins of G and M is the matrix whose rows are the characteristic vectors of the elements of J.

The optimal value of the first program is $\nu(D)$ and the optimal value of the second program is $\tau(D)$. We can test these programs with the following example:



M =

a	b	c	d	e	f	g
1	1					
1		1		1		
1		1			1	
1			1	1		
1			1		1	
	1	1	1			
			1	1		
			1		1	

Figure 4.4: Matrix M example.

In figure 4.4 it is easy to appreciate that the minimum dicut has size 2 (i.e. $\tau = 2$) and the maximum packing of dijoins has size also equal to 2 (i.e. $\nu = 2$). This means that this graph fulfills Woodall's Conjecture.

The problems for computing this solution are:

- Compute the set of inclusion-wise minimal dijoins to create the matrix M.
- Solve the linear program, which in general is hard, but there are solvers like CPLEX [2].

We have not found a solution to compute the first problem in an efficient way. We do have come up with a brute force algorithm that would let us compute the matrix M but, since the matrix can be very big, it is not ideal.

4.4 Pseudo-code

We will use this section to explain step by step how the code should work gathering all the results we have seen in the previous sections.

The first step is to generate all non-isomorphic undirected graphs according to the restrictions we have set in section 4.1. The program has to do this for a certain number of vertices that, will start on three, and increase in each iteration until the code discovers a graph that does not fulfill Woodall's Conjecture or, it reaches the maximum number of vertices we have decided to study. For this part, we can use from Nauty [14] the command:

```
geng [-cd#] n [file]
```

that generates all non-isomorphic graphs with n vertices, one connected component ($-c$), and with a lower bound for the minimum degree (d#) in our case, equal to two, which will allow us to target graphs that are not 2-edge-connected since their minimal dicut would have a size equal to one, see theorem 3.4.

Once we have all the graphs, the next step is to discard all of those that are series-parallel, using Valdes, Tarjan and Lawer algorithm [22]. This property can already be seen using SageMath [3], which has the built-in function:

```
treewidth()
```

that in case the graph was series-parallel, it would return two.

The last property we have seen that applies to undirected graphs is the $(k-1,1)$-partition-connected using Frank's algorithm [9], see section 3.3. We have not found any implementation for this property in the software's we have checked.

After all the restrictions for the undirected graphs have been checked, the next step is to create all the acyclic orientations of the graphs that have survived. To do so, we will also use from Nauty, the command:

```
directg [-o|-a] [infile [outfile]]
```

which allows us to generate all non-isomorphic directed orientations. The options we have written allows us to:

- $-o$: orient each edge in only one direction, never both.
- $-a$: only make acyclic orientations (implies -o).

It is very important to remark the $-a$ parameter, since it allows us to avoid generating all those directed orientations that once reduced to an acyclic digraph, would be isomorphic.

Once we have all the orientations set, it is the moment to check the property we have seen in section 3.3 that says that the digraphs in which every sink-node is reachable from every source-node fulfill the conjecture. To do so, we would use SageMath to first find the source-nodes and the sink-nodes using the built-in functions:

```
in_degree()
out_degree()
```

For source-nodes, $indegree = 0$ since no arcs arrives at them. For sink-nodes, $outdegree = 0$ since no arcs depart from them. Having the set of source-nodes and sink-nodes, we could use, again from SageMath, any of the following built-in functions:

```
breath_first_search()
depth_first_search()
```

these functions allow us to find all vertices reachable from a specific vertex. So, given a source-node, it would find all vertices reachable from there, and we would only have to make sure all sink-nodes are in that list.

Now, using the max-flow min-cut algorithm adaptation we have seen in section 4.2, we calculate the value of τ that in the case it is equal to two it already verifies the conjecture, see theorem 3.4.

Finally, the last step is to calculate the matrix M and solve the linear program to find ν, following the steps defined in section 4.3. In case that $\tau \neq \nu$, we would print the digraph and stop the program, since this would be the counterexample for Woodall's Conjecture.

The overall pseudo-code of the previous explanation, for graphs with a total number of vertices between 3 and 10, would look like the following:

```
max_number_of_vertices = 10
for (n = 3 ; n < max_number_of_vertices ; n++)
        G = generateAllNonIsomorphicGraphs(num_vertices = n)
        for (graph in G)
                if [(graph is series-parallel) or
                    (graph is (k-1,1)-partition-connected)]
                then
                        break
                end if
                D = generateAllAcyclicOrientations(graph)
                for (digraph in D)
                        if [allSourcesReachAllSinks(digraph)]
                        then
                                break
                        end if
                        τ = maxFlowMinCutAdaptation(digraph)
                        if (τ = 2)
                        then
                                break
                        end if
                        M = calculateMatrixM(digraph)
                        ν = linearProgramNu(M)
                        if (τ ≠ ν)
                        then
                                print(Conjecture does not hold)
                                print(Counterexample:(digraph))
                                exit
                        end if
                end for
        end for
end for
print(The conjecture holds for all digraphs)
print(with a number of vertices between 3 and 10)
```

Chapter 5

Edmonds-Giles Conjecture

In this chapter we will talk about a weighted version of Woodall's conjecture. Edmonds and Giles conjectured in 1977 that the maximum number of directed joins in a packing of dijoins is equal to the minimum weight of a directed cut for any weighted directed graph.

Definition 5.1. In weighted digraphs, a collection of dijoins is a **packing** if each arc a is present in at most $w(a)$ of the dijoins and the weight of a dicut is the sum of the weights of its arcs.

Conjecture 5.2. (Edmonds and Giles, 1977) [5] *For every non-negative integer arc weight function w, the minimum weight of a dicut is equal to the maximum packing of dijoins.*

Remark 5.3. Note that if $w \equiv 1$, then conjecture 5.2 is just Woodall's Conjecture.

5.1 Schrijver's Counterexample

Schrijver [20] exhibited an example showing that the previous statement is not true. However, the conjecture does hold for digraphs where every source is connected to every sink, see Feofiloff and Younger [7] and Schrijver [21].

The counterexample found by Schrijver is the weighted directed graph (D, w), see figure 5.1. In this digraph, the arcs either have weight equal to 0 or 1. By convention, arcs with $w(a) = 0$ are the ones represented by thin lines, and the arcs with $w(a) = 1$ are the ones with thick lines.

Schrijver proved that the weighted digraph $D = (V, A)$ from figure 5.1 has weight two as the minimum weight dicut, but there is not a packing of two dijoins.

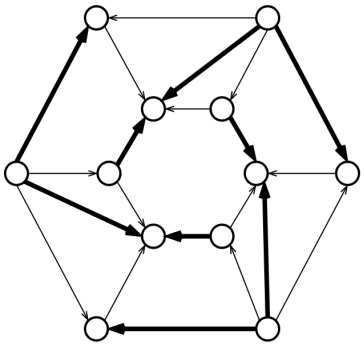


Figure 5.1: Schrijver's counterexample.

To prove that the graph found by Schrijver is a counterexample of the conjecture, I will follow the proof given by Aaron Michael Williams [24]. Previous to the proof there are several ideas that have to be defined:

We will denote the arcs and nodes included in a path P as $arcs(P)$ and $nodes(P)$, respectively.

Lemma 5.4. *Let $D = (V, A)$ be a digraph. Suppose P is a path and $\nabla^+(U)$ is a dicut where U is a subset of V, such that $\mid arcs(P) \cap \nabla^+(U) \mid \geq 2$ then $arcs(P) \cap \nabla^+(U)$ contains at least one forward arc of P and one backward arc of P.*

Proof. A dicut separates a graph into two connected components C_1, C_2, having $\mid arcs(P) \cap \nabla^+(U) \mid \geq 2$ means that at least we have traveled from C_1 to C_2 and from C_2 to C_1 but since all arcs are directed from C_1 to C_2 one will be forward in P and the other will be backward in P. □

The previous figure 5.1 shows that the arcs of weight 1 can be partitioned into three disjoint paths P_1, P_2 and P_3. Furthermore, every node in these paths is either a source or a sink node inducing a trivial cut of weight 2.

Let $\tau = \tau(D, w)$ denote the smallest weight of a directed cut in (D, w). Notice that $\tau = 2$ and this is obtained by trivial dicuts. Let $\nu = \nu(D, w)$ denote the size of the largest packing of dijoins in (D, w).

Suppose that $\nu = 2$, which means it fulfills the conjecture. Therefore, the arcs of weight 1 can be partitioned into two dijoins J_1 and J_2. Due to the trivial cuts of weight 2, J_1 and J_2 neither include nor exclude adjacent arcs along P_1, P_2, or P_3,

see definition 5.1.

In particular, since each path is alternating (arcs in the path alter from backward to forward), J_1 and J_2 must divide the forward and backward arcs of each path between them.

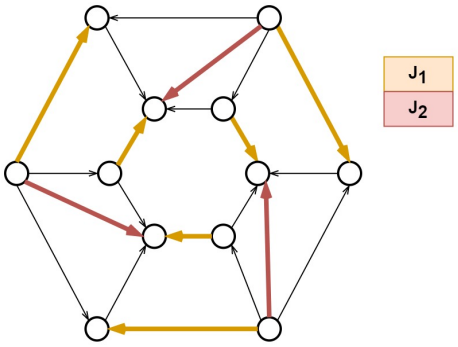


Figure 5.2: One possible distribution of the dijoins.

Up to exchanging J_1 and J_2 there are exactly four such partitions, one of those is shown in figure 5.2. Therefore, using the lemma 5.4, J_1 and J_2 have both a non-empty intersection with each directed cut that intersects P_1, P_2, or P_3 more than once. However, for each of the four possible partitions, there exists a directed cut that intersects each path at most once and is disjoint from either J_1 or J_2. Which makes it impossible for J_1 and J_2 to both be directed joins.

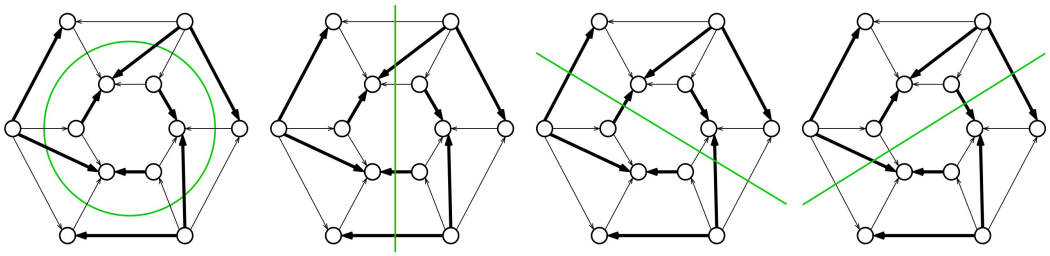


Figure 5.3: Non trivial dicuts (in color green).

In figure 5.2, it is easy to see that dijoin J_1 does not intersect the first dicut from figure 5.3.

Conclusions

In this project, we have gone through the graph theory related to dicuts, dijoins and their packings. We have built up the reader's knowledge of this area from zero to a high understanding of the field.

Once the introduction part was done, we started this project by proving in section 1.6, observations that we thought were important during the phase of documentation. Those observations were not found in any paper but rather stated by us once we realized they were needed or that they would simplify the following chapters.

From the main statements, we have started by seeing and proving the Lucchesi-Younger Theorem, a theorem that can be seen as the dual version of the conjecture. Then, we have written about Woodall's Conjecture, the proof for the first cases and we have seen three specific types of graphs that validate the conjecture. Finally, to close the circle, we have seen the counterexample for the Edmonds-Giles Conjecture, the weighted version of Woodall's Conjecture.

As presented in the introduction we have defined a program that validates, for all possible graphs, if they fulfill Woodall's Conjecture. The initial idea was to go one step further and besides proposing a code, implement it and used it to prove that all graphs, up to a certain number of vertices, fulfill the conjecture. Due to the lack of time and work overload from other areas, the implementation part has not been possible to achieve. Using everything we have gathered, the next step would be the implementation of the program, a process that would not be short and would take many hours since some of the steps we have designed have not been implemented yet by any of the software's we have checked.

Now, on a more personal note, I want to say that even though one of the initial purposes of the work has not been accomplished, I end this project very happy with the outcome we have been able to get. Gathering all the materials related

to Woodall's Conjecture, understanding the Lucchesi-Younger proof, and proving the Edmonds-Giles Counterexample has definitely not been an easy task to do.

I have found it very interesting to study a field still very open to this day. Woodall's Conjecture was stated in 1978 and, since then, many weaker statements have been formulated and still have not been proven. Some of those conjectures are the following:

Conjecture 5.5. (Matthew Devos [4]) *Is there an integer $k \geq 3$ such that, if every directed cut in a digraph has size of at least k, then it contains 3 disjoint dijoins?*

If Woodall's Conjecture was proven to be true, we would have k disjoint dijoins instead of three which would prove the conjecture.

Another weaker statement we want to present is the following:

Definition 5.6. For a positive integer k, a **k-dijoin** is a set of edges containing k arcs from every directed cut.

Conjecture 5.7. (Egres Open [18]) *If every directed cut in a digraph has size of at least $2k$, then there are 2 disjoint k-dijoin.*

This is known to be true for $k = 1$, but not known for larger values of k. Suppose that Woodall's Conjecture is true. If every dicut has size of at least $2k$ it would mean that we have $2k$ disjoint dijoins. Separate those into two groups of k dijoins then, joining all the dijoins in every group would create two disjoint k-dijoins which would prove the conjecture.

I want to end this project by saying that I have found it very curious to work around a field that has only been alive for the past 50 years. All the major statements that I assembled during this project, Woodall's Conjecture, Lucchesi-Younger Theorem, and Edmonds-Giles Conjecture were published in the seventies, which is quite strange comparing it to what I have been seeing in university these past years, since I always have found myself studying topics that were stated centuries ago and, even though they still applied to today's theory, I have found it very refreshing to work around a field that is still today considered very new.

Bibliography

[1] Ahmad Abdi, *47853 Packing and covering: Lecture 9*, (2019).

[2] IBM ILOG Cplex, *V12. 1: User's manual for CPLEX*, International Business Machines Corporation **46** (2009), no. 53, 157.

[3] The Sage Developers, William Stein, David Joyner, David Kohel, John Cremona, and Burcin Erocal, *SageMath, version 9.0*, 2020, url = http://www.sagemath.org.

[4] Matthew Devos, *Open problem garden*, Simon Fraser Univeristy, url = http://www.openproblemgarden.org.

[5] Jack Edmonds and Rick Giles, *A min-max relation for submodular functions on graphs*, Stud. integer Program., Proc. Workshop Bonn 1975, Ann. Discrete Math. 1, 185-204 (1977)., 1977.

[6] Paulo Feofiloff, *Woodall's conjecture on packing dijoins: a survey*, (2005), url = https://www.ime.usp.br.

[7] Paulo Feofiloff and Daniel H Younger, *Directed cut transversal packing for source-sink connected graphs*, Combinatorica **7** (1987), no. 3, 255–263.

[8] Lester R. Ford and Delbert R. Fulkerson, *Maximal flow through a network*, Canadian Journal of Mathematics **8** (1956), 399–404.

[9] András Frank, *On the orientation of graphs*, Journal of Combinatorial Theory. Series B **28** (1980), 251–261 (English), url = https://www.sciencedirect.com/science/article/pii/0095895680900714.

[10] Orlando Lee and Yoshiko Wakabayashi, *Note on a min-max conjecture of Woodall*, J. Graph Theory **38** (2001), no. 1, 36–41 (English).

[11] László Lovász, *On two minimax theorems in graphs*, J. Comb. Theory, Ser. B **21** (1976), 96–103 (English).

[12] Cláudio Leonardo Lucchesi, *A minimax equality for directed graphs*, (1977).

[13] Cláudio Leonardo Lucchesi and Daniel H. Younger, *A minimax theorem for directed graphs*, J. Lond. Math. Soc., II. Ser. **17** (1978), 369–374 (English).

[14] Brendan D. McKay and Adolfo Piperno, *Nauty web page documentation*, (2014), url = https://users.cecs.anu.edu.au/ bdm/nauty/.

[15] Brendan Damien McKay and Adolfo Piperno, *Practical graph isomorphism, II*, Journal of Symbolic Computation **60** (2014), 94–112, url = https://www.sciencedirect.com/science/article/pii/S0747717113001193.

[16] András Mészáros, *A note on disjoint dijoins*, Combinatorica **38** (2018), no. 6, 1485–1488 (English).

[17] J. E. Olson, *A combinatorial problem on finite abelian groups. I*, J. Number Theory **1** (1969), 8–10 (English).

[18] Egres Open, *Woodall's conjecture, Open problem forum of the Egerváry Research group*, 2017, url = http://lemon.cs.elte.hu/egres/open/*MainPage*.

[19] Herbert E. Robbins, *A theorem on graphs, with an application to a problem of traffic control*, Am. Math. Mon. **46** (1939), 281–283 (English).

[20] A. Schrijver, *A counterexample to a conjecture of Edmonds and Giles*, Discrete Math. **32** (1980), 213–214 (English).

[21] Alexander Schrijver, *Min-max relations for directed graphs*, North-Holland Mathematics Studies, vol. 66, Elsevier, 1982, pp. 261–280.

[22] Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler, *The recognition of series parallel digraphs*, SIAM J. Comput. **11** (1982), 298–313 (English).

[23] Douglas B. West, *Introduction to graph theory*, New Delhi: Prentice-Hall of India, 2005 (English).

[24] Aaron Williams, *Packing directed joins*, Master's thesis, University of Waterloo, 2004.

[25] Douglas R. Woodall, *Menger and König systems*, Theor. Appl. Graphs, Proc. Kalamazoo 1976, Lect. Notes Math. 642, 620-635 (1978)., 1978.