

# Structural Properties of Decomposable Digraphs

by

Gabriella Juhl Jensen

supervised by

Prof. Jørgen Bang-Jensen



UNIVERSITY OF SOUTHERN DENMARK

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

# Contents

0.1	Introduction . . . . .	2
<b>I</b>	<b>Introduction to Decomposable digraphs and some solutions to Hamilton cycle problem on those digraphs.</b>	<b>3</b>
<b>1</b>	<b>Intro</b>	<b>4</b>
1.1	Graphs and Digraphs . . . . .	4
1.2	Computational complexity . . . . .	6
1.2.1	Measure time of algorithm (Polynomial, exponential) . . . . .	6
1.2.2	NP problems and classifications . . . . .	7
1.3	Classes of Digraphs . . . . .	7
1.3.1	introduction to some digraph classes . . . . .	7
1.3.2	Semicomplete Digraphs . . . . .	9
1.3.3	muligvis Transitive Digraphs . . . . .	9
1.3.4	Strong Digraphs . . . . .	9
1.3.5	Round Digraphs . . . . .	9
<b>2</b>	<b>Decomposable Digraphs</b>	<b>10</b>
2.1	Genral about Decomposable digraphs . . . . .	10
2.2	Quasi-transitive . . . . .	11
2.3	Locally semicomplete . . . . .	12
<b>3</b>	<b>Path cover, Hamilton cycles and pancyclic digraphs</b>	<b>15</b>
3.1	Why hamilton path and cycle problem is NP-Hard . . . . .	15
3.2	Hamiltonian Locally semicomplete Digraphs . . . . .	16

3.3	Hamiltonian Quasi-transitive Digraphs . . . . .	17
3.4	Pancyclisc Digraphs . . . . .	17
<b>II</b>	<b>Linkage and weak linkage</b>	<b>18</b>
<b>III</b>	<b>Spanning disjoint subdigraphs (Arc decomposition)</b>	<b>19</b>

## **0.1 Introduction**

Why we need graphs.

## **Part I**

**Introduction to Decomposable  
digraphs and some solutions to  
Hamilton cycle problem on those  
digraphs.**

# Chapter 1

## Intro

This chapter is if you do not know what a graph is, there is some notation this thesis may use differently then others are and different articles are doing. this chapter is also going to introduce the notation and names for the classes of digraphs we are looking at. Then in section 1.2 we are going to cover runing time principles and problems that in general are hard to solve, called NP-Hard problems and what these have to do with this thesis. Last section 1.3 is going to cover all the different classes we are going to use and cover in this thesis.

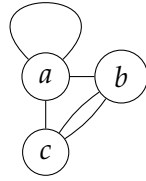
### 1.1 Graphs and Digraphs

Before going deep into structural properties of decomposable digraphs we first need to establish what a graph is. For some graph  $G(V, E)$  where  $V$  and  $E$  are two sets contaning the **vertices** (also commonly called nodes) and **egdes** of the graph respectively. We define the **size** of the graph to be the number of vertices  $|V|$  this is also known as **cardinality** of  $V$ . An **edge**  $e \in E$  where  $e \equiv (a, b)$  and  $\{a, b\} \subseteq V$  then  $e$  is an edge in  $G$ ,  $e$  is said to be **incident** to  $a$  and  $b$ . We call  $a, b \in V$  **adjacent** if there is an edge  $(a, b)$  or  $(b, a)$  (two given vertices connected by an edge is said to be adjacent). If an edge goes from and to the same vertex  $(a, a)$  it is called a **loop**. The set of edges  $e_1, \dots, e_k$  is usally describe whit the letter  $E$  where each edge contains a pair of vertices that are adjacent.

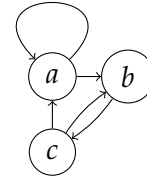
In a graph we have something called a **walk** which is a alternately ordering of vertices an edges in the graph  $G$  where the edge in between the two vertices in the ordering is an edge between the vertices in  $G$  (for  $(a, e_1, b)$  to be a walk the edge  $e_1$  has to be between  $a$  and  $b$ ). We call a walk closed if the first vertex in the walk is the same as the last.

Every vertex  $v \in V$  of  $G(V, E)$  have a **degree** denoted  $d(v)$  which is the number of incident edges to  $v$ . A **path** in a graph is a walk where each vertex in the ordering can only appear one time. A cycle is a closed walk where the only vertex pressent more then one time is the first vertex(for the walk to be closed the first vertex has to appear last to also called a closed path). Let  $X$  be a subset of the vertices  $X \subseteq V$  then we say that  $V \setminus X$  is the set of vertices with out the vertices in  $X$ , i.e.  $V/X \equiv V - X$ . A subgraph  $H$  of  $G$  can contain any of the vertices and the arcs connected to the chossen vertecies in  $H$ . you can not have an edge conecting no vertices in  $H$  but you do not have to choose all the arcs in  $G$  between the chossen vertices in  $H$  for  $H$  being a subgraph.

Before delving more specific into graphs and digraphs we must establish some impor-



(a) graph  $G(V, E)$  is an example of a graphs, the red edge is a loop, and all pair of vertices in this graph is adjacent.



(b) This is an oriantation of the edges in the graph which makes this a digraph

tant prerequisite and properties. A graph is called **simple** if there is no loops and no multiple edges. With multiple edges it means multiple edges between the same pair of vertices like in Figure 1.1a between  $b$  and  $c$ .

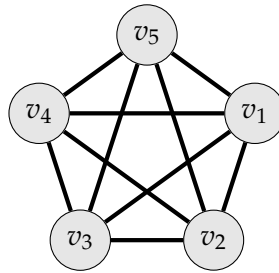


Figure 1.2: Complete graph with 5 vertices.

A graph is **connected** if there exists a path between all pair of vertices in the graph and **disconnected** otherwise. A graph is called **complete** if there for all pair of vertices in the graph is an edge between them see Figure 1.2.

If we instead of edges have **arcs** between the vertices we call it a **digraph**. An arc is describe just like an egde with two adjacent vertices  $(a, b)$  the first vertex mentioned in an arc is the vertex **from** where the arc starts also called the **tail**, the second vertex is where the arc is pointing **to** also called **head**. The set of arcs is normaly denoted  $A$  like the set of edges is denoted  $E$ . So the arc  $(a, b)$  goes from  $a$  to  $b$ , if you wanted it the other way around the arc is  $(b, a)$ . These graph contaning only arcs and no edges is called a digraph  $G(V, A)$  which is what we in this project are focusing on see Figure 1.1b.

For two vertices  $x$  and  $y$  in  $D(V, A)$  then if we have an arc from  $x$  to  $y$  we say that  $x$  **dominates**  $y$  this is denoted like this  $x \rightarrow y$ . If we talk about subgraphs  $A$  and  $B$ , then  $A$  **dominates**  $B$  if for all  $a \in A$  and  $b \in B$ ,  $a \rightarrow b$ . If there is no arcs from  $B$  to  $A$  we denote it  $A \mapsto B$  and if both  $A \rightarrow B$  and  $A \mapsto B$  we say that  $A$  **completely dominates**  $B$  and this is denoted  $A \Rightarrow B$ .

In a digraph we have something called the **underlying graph**. An underlying graph of a digraph is where all arcs are replaced by edges (edge is used every time we talk about undirected edges between vertices, when using directions it is called an arc). Let  $X \subseteq V$  Then we can make the subdigraph  $D \langle X \rangle$  which is the subgraph  $D$  induced by the set  $X$  meaning that all the vertices is from  $X$  and the arcs is from  $A \in G$  but where both head and tail is incedent to the vertices in  $X$ . We will denote the graph  $D \langle V \setminus X \rangle$  for some  $X \subseteq V$  as  $D - X$ . A digraph is **connected** if the underlying graph is connected, (also called weakly connected), a digraph can be **strongly connected** and **semi connected** too. A digraph is called **semi connected** if there for each pair  $u$  and  $v$  exists a path from either  $u$  to  $v$  or  $v$ . It

is said to be **strongly connected** if for each pair of vertices  $u$  and  $v$  there exists a path from both  $u$  to  $v$  and  $v$  to  $u$ . A strongly connected digraph is also called a **strong** digraph. A strong digraph have a subset  $S$  called a **seperator** if  $D - S$  is not strong, we also say that  $S$  **seperates**  $D$ . A seperator  $S$  is called **minimal seperator** of  $D$  if there exists no proper subset  $X \subset S$  that seperates  $D$ . Now we can introduce a  **$k$ -strong** digraph  $D$  which is a strong digraph with  $|V| \geq k$  and a minimal seperator  $S$  on  $|S| = k$ .

In a digraph  $D(V, A)$  we mostly use the **ddegree** as two different degrees namely **out degree**,  $d^+(v)$ , and **in degree**,  $d^-(v)$ , that is the arcs from  $v$  and to  $v$  respectively. In a digraph  $D$  we can talk about the over all **minimum out degree**,  $\delta^+(D) = \min\{d^+(v) | v \in V\}$  and **minimum in degree**,  $\delta^-(D) = \min\{d^-(v) | v \in V\}$  sometime we is going to need the minimum of these to  $\delta(D) = \min\{\delta^+(D), \delta^-(D)\}$  and called the **minimum degree**. For every vertex  $v$  the vertices that is **adjacent** with  $v$  we denote  $N^+(v)$  and  $N^-(v)$  as the set of vertices that is dominated by (out-nieghbours) $v$  and dominates (in-nieghbours) $v$ , respectively. This means that  $d^+(v) = |N^+(v)|$  and  $d^-(v) = |N^-(v)|$ .

For simplicity when mentioning paths and cycles in digraphs it will be **directed** paths and cycles if not anything else is mensioned. By **directed** means that we go from tail to head on every arc on the path or cycle. When mentioning paths in a digraph it sometimes makes more sence specifying the head an tail of the path, so a path from  $s$  to  $t$  is denoted  **$(s, t)$ -path**. In some digraphs there is more than one path between the same two vertices these paths can use the same arcs or same vertices or be totally distinct from eachother, the maximum number of disjoint path between two vertices in a digraph is denoted  $\lambda_D(s, t)$

## 1.2 Computational complexity

In this section we will go over how time is measured for an algorithm and what it means for a problem to be polynomially solveable or polynomially verifyable. Also what it means for a problem to be NP-hard and NP-complete and how we found out if a problem is either of them.

### 1.2.1 Measure time of algorithm (Polynomial, exponential)

The runing time of an algorithm is based on how many steps it is going thourgh which is somtimes based on the input that the algorithm takes we are going to denote an algorithms running time as a function  $f(n)$  over the input  $n$ . This is how different functions can descirbe the running time of an algorithm, if an algorithm has the same number of steps no matter what the input is it has a constat running time where the constant is the number of steps the algorithm uses.

An algorithm can also take the form of an polynomial function or even exponentiel, if this is the case we uses some notation as big- $O$  notation or  $\theta$ . Big-oh is the most used one and is the notation we are going to use in this thises, if the algorithm takes  $f(n) = 4n^3 + 2n^2 - n + 2$  time we denote it in big-oh notation as  $O(n^3)$  as it is the biggest term of  $f(n)$ .

Since the shorter the runningtime is the better the algortihm is. Since the exponentiel runningtime algortihms take forever on large inputs, we would want to improve them, but sometimes you are left with problems where that is not a possibility.

So we are going to classify the problems in groups of how long time it takes to decide or verify the problem's solution. A problem that is decided in polynomial time is in the class called  $P$ . Which means for every given time of input in a problem from  $P$  we can find the solution for the problem in polynomial time.

### 1.2.2 NP problems and classifications

As shortly described above there is something called a **polynomial verifier** for a problem. That means given a problem and then given a solution we can in polynomial time verify if it is a solution to the given problem. This is the class we call NP.

**Definition 1.2.1.** *NP is the class of languages that have polynomial time verifiers.*

Obviously if you can find a solution in polynomial time you can also verify whether a solution is correct in polynomial time. So  $P \subseteq NP$ . There is also a class called  $NP - \text{Hard}$  but before we can explain that we need to explain what it means for a problem to be polynomial reducible to another problem. For a specific problem  $A$  and another problem  $B$  then if there exists an algorithm that can take a solution from  $A$  and make it a solution for  $B$  in polynomial time. When such an algorithm exists it is called a polynomial verifier and we say that  $A$  is **polynomial reducible** to  $B$  or just that  $A$  is **reduced** to  $B$ . **NP-Hard** are the class of problems that every NP problem can be polynomially reduced to. A problem in the class of NP-Hard problems does not necessarily mean that it is NP itself. If a problem is both **NP** and **NP-Hard** we call it **NP-Complete**. The problems we are **mostly** focusing on are in the class of **NP-Complete** problems.

## 1.3 Classes of Digraphs

We can classify specific collections of graphs. The reason for this is that digraphs of smaller collections of digraphs (like tournaments are a smaller collection of semicomplete digraphs) might be easier to solve on general digraphs but are easy/polynomially solvable on specific types of digraphs.

A group of these problems is called NP-complete problems which sometimes sound easy to solve for graphs but only for some specific graphs we know how to solve it in polynomial time. Like finding paths in digraphs or cycles or more specific things, but in general the more we know about a digraph we can use to solve hard problems which in general would be time consuming like the problems that are NP-hard. By some quick fast algorithm you can check whether a digraph belongs to a certain **class** of digraphs. A class of digraphs is a collection of digraphs with certain properties in common like **tournaments**.

### 1.3.1 introduction to some digraph classes

**Tournaments** is a digraph where the underlying graph is complete. So a complete graph of order 5 any orientation of the edges concludes in a tournament. Strong digraphs are also in themselves a classification of digraphs. Classes of digraphs can be overlapping each other or



be fully contain in each other like tournaments is fully contain in the class called semi-complete digraph. A **semicomplete** digraph is where the underlying graph is complete multigraph, there can be some multiple edges in between the same pair of vertices in the underlying graph. Since the class called semicomplete digraphs contains all digraphs where the underlying graph is a complete multigraph it clearly also contains the graph with only one arc between every pair of vertices (Tournaments). A **complete** digraph is where every pair of vertices  $a, b \in V$  the arc  $(a, b)$  and  $(b, a)$  is present in the graph.

If you can split the graph into two sets of vertices  $A$  and  $B$  such that  $A \cup B = V$  and there is no arcs inside these sets, then we classify this as an **bipartite** digraph. This means all arcs in the graph is in the form  $(a, b)$  or  $(b, a)$  for all  $a \in A$  and  $b \in B$ . The sets  $A$  and  $B$  are called the partite sets of  $D(V, A)$ . The underlying graph of a bipartite digraph is also called bipartite since there is no edges inside  $A$  or  $B$ . If there exists more than two of these partite sets we call the digraphs **multipartite**, since there is multiple partite sets in the graph, bipartite sets  $\subset$  multipartite.

A much used type of digraph is an **acyclic** digraph. It is a digraph where for an specific ordering of the vertices  $V = v_1, v_2, \dots, v_n$  the arcs in the digraph is  $(v_i, v_j)$  where  $i < j$  for all  $(v_i, v_j) \in A$ . This ordering is called an **acyclic ordering** and can also be used to order strong components in a non-strong digraph such that the ordering of the component  $C_1, C_2, \dots, C_k$  is an acyclic digraph when contracting the components into  $k$  vertices. When classifying digraphs there is several ways of doing this, like **transitive** digraphs which are digraphs where for all vertices  $a, b, c \in V$  where the arc  $(a, b)$  and  $b, c$  is present in the digraph ( $\in A$ ), the arc  $(a, c)$  has to be a part of  $A$  too. using the same kind of classification there is digraphs which are **Quasi-transitive** which is for all vertices  $a, b, c \in V$  where the arc  $(a, b)$  and  $b, c$  is present in the digraph ( $\in A$ ),  $a$  and  $c$  has to be adjacent by at least one (more arcs in between are also allowed) arc in either direction ( $(a, c)$  or  $(c, a)$ ). These graphs are going to be mentioned a lot in this thesis since the graph is also what we call **decomposable**.

**Decomposable** digraphs is also a classification of graphs which are decomposable, for a graph  $D$  to be decomposable we have  $H_1, H_2, \dots, H_k$  **houses** and  $S$  where  $V(S) = s_1, s_2, \dots, s_k$  which are all digraphs by them self but if each  $s_i$  is replaced by the digraph  $H_i$   $i = 1, 2, \dots, k$  we have the graph  $D$ , where  $H_i \rightarrow H_j \in D$  if  $s_i \rightarrow s_j \in S$  denote this decomposition like  $D = S[H_1, H_2, \dots, H_k]$ . This is the class of digraphs we are focusing on in this thesis. If all the houses are independent sets we call  $D = S[H_1, H_2, \dots, H_k]$  the extension of  $S$ . If  $S$  is a semi-complete digraph we call the extension of these **extended semicomplete** digraph. Like we already mentioned Quasi-transitive digraphs are decomposable but we have several classes that are decomposable, and another class of digraphs that is going to be used a lot in this is **locally semicomplete** digraphs.

First we are going to introduce **in-locally semicomplete** digraphs and **out-locally semicomplete** digraphs which is for every in-neighbor of a vertex  $x \in V$  they have to be adjacent ( $x \cup N^-(x)$  induces a semicomplete digraph) is the in-locally semicomplete digraph if it is true for all  $x \in V$ . Respectively it is called an out-locally semicomplete digraph if  $\forall x \in V$  the out-neighbors,  $N^+(x)$ , has to be adjacent. If a digraph is both in-locally semicomplete and out-locally semicomplete, it is called a **locally semicomplete** digraph. Why both Quasi transitive digraphs and some locally semicomplete digraphs are decomposable will be described in section ??.

The last class of digraph that are important for this thesis is the round digraphs. A digraph is called a **round** digraph if there exists an ordering of the vertices  $v_1, v_2, \dots, v_n$  such that for all  $v_i$ ,  $N^+(v_i) = v_{i+1}, v_{i+2}, \dots, v_{i+d^+(v_i)}$  and  $N^-(v_i) = v_{i-d^-(v_i)}, v_{i-d^-(v_i)-1}, \dots, v_{i-1}$ .

**1.3.2 Semicomplete Digraphs**

**1.3.3 muligvis Transitive Digraphs**

**1.3.4 Strong Digraphs**

**1.3.5 Round Digraphs**

## Chapter 2

# Decomposable Digraphs

Decomposable digraphs is what we in this thises is focusing on. We have introduced short what a decomposable digraph is but there is subclasses to focus on and a lot of other crucial definitions and theroems to cover about these digraphs before delving into the NP-hard problems. First we cover some general things about decomposable digraphs the next section is about quasi-transitive digraphs, and why they are a subclass of decomposable digraphs and  $\phi_1$ -decomposable digraphs. At the end of the section we proof that these decompositions can be found in polynomial time. Which is going to be crucial for solving some NP-hard problems for this class of digraphs. Then we are going to look at a very general class of digraphs locally semicomplete digraphs, where this class can be split up to 3 different subclasses where 2 of those are decomposable. This is covered in section 2.3 and is going to be used in later chapters.

### 2.1 Genral about Decomposable digraphs

Recall that a decomposable digraph  $D$  can be decomposed into a main graph  $S$  where  $|S| = k$  and  $k$  houses  $H_1, H_2, \dots, H_k$ , where each vertex in  $S = \{v_1, v_2, \dots, v_k\}$  is replaced by the house  $H_i$  replace  $v_i$  and the arcs between the houses is as follows  $H_i \rightarrow H_j$  in  $D$  if  $v_i \rightarrow v_j$  in  $S$  remember that for a set  $X$  to dominate an other set  $Y$  (meaning every vertex in the dominating set dominates every vertex in the dominated set) we denoted it  $X \rightarrow Y$ . If no arc between  $v_a$  and  $v_b$  in  $S$  then there is no arc between the sets  $H_a$  and  $H_b$  in  $D$ . The thing about decomposable digraphs is that if there is an arc between  $H_i$  and  $H_j$  either one of the houses totally dominates the other (ex.  $H_i \Rightarrow H_j$ ) or they dominate each other (ex.  $H_i \rightarrow H_j$  and  $H_j \rightarrow H_i$ ).

Decomposable digraphs can be classed by a set of digraphs  $\phi$ , when  $D = S[H_1, H_2, \dots, H_k]$  it is  $\phi$ -**decomposable** if  $D \in \phi$  or if  $S \in \phi$ . The chioces of  $H_i$  for  $i = 1, 2, \dots, k$  does not determine anything about the digraph being  $\phi$ -decomposable but the class of **totally  $\phi$ -decomposable** digraphs is where  $D$  is  $\phi$ -decomposable and each  $H_i$  is totally  $\phi$ -decomposable. We are going to make two shuch sets of digraphs  $\phi_1$  which is the union of semicomplete digraph and acyclic digraph both classes deskribed in section 1.3 and  $\phi_2$  which is the union of semicomplete and round digraphs also deskribed in section 1.3.

$$\phi_1 = \text{Semicomplete digraphs} \cup \text{Acyclic digraphs} \quad (2.1)$$

$$\phi_2 = \text{Semicomplete digraphs} \cup \text{Round Digraphs} \quad (2.2)$$

## 2.2 Quasi-transitive

First we need to recall what a quasi transitive digraph is. For every triplet  $x, y, z$  in a quasi-transitive digraph if  $x \rightarrow y$  ( $x$  dominates  $y$ ) and  $y \rightarrow z$  ( $y$  dominates  $z$ ), then there has to be at least one arc in either direction between  $x$  and  $z$ . When working with quasi-transitive digraphs there are many things you can depend on, things that the structure has already decided for us.

**Lemma 2.2.1.** [1] Suppose that  $A$  and  $B$  are distinct strong components of a quasi-transitive digraph  $D$  with at least one arc from  $A$  to  $B$ . Then  $A \rightarrow B$ .

Recall that this means that every vertex in  $A$  has an arc to every vertex in  $B$ . Like non-strong quasi-transitive digraph we can also say something about strong quasitransitive digraphs.

**Lemma 2.2.2.** [1, 2] Let  $D$  be a strong quasi-transitive digraph on at least two vertices. Then the following hold:

- (a)  $\overline{UG(D)}$  is disconnected;
- (b) If  $S$  and  $S'$  are two subdigraphs of  $D$  such that  $\overline{UG(S)}$  and  $\overline{UG(S')}$  are distinct connected components of  $\overline{UG(D)}$ , then either  $S \rightarrow S'$  or  $S' \rightarrow S$  or both  $S \rightarrow S'$  and  $S' \rightarrow S$  in which case  $|V(S)| = |V(S')| = 1$ .

These two lemmas are also a part of proving the one theorem which states that quasi-transitive digraphs can be decomposed no matter if there are strong or nonstrong digraphs.

**Theorem 2.2.3.** [3] Let  $D$  be a quasi-transitive digraph.

1. If  $D$  is not strong, then there exists a transitive acyclic digraph  $T$  on  $t$  vertices and strong quasitransitive digraphs  $H_1, \dots, H_t$  such that  $D = T[H_1, \dots, H_t]$ .
2. If  $D$  is strong, then there exists a strong semicomplete digraph  $S$  on  $s$  vertices and quasitransitive digraphs  $Q_1, \dots, Q_s$  such that each  $Q_i$  is either a single vertex or is nonstrong and  $D = S[Q_1, \dots, Q_s]$ .

This theorem is also what we are going to use more than once, to prove several of the problem solving theorems throughout this thesis.

*Proof.* Since we can decompose both strong quasi-transitive digraphs and non-strong quasi-transitive digraph we are going to prove if  $D$  is not strong first and then after if  $D$  is strong. So suppose  $D$  is not strong, then we know we can enumerate the strong components in an acyclic order let these be  $H_1, \dots, H_t$ . Recall that an acyclic ordering of the strong components does not mean that there are no arcs going back in the ordering, but we will prove that now. Now from 2.2.1 we know that if there is an arc between two of the strong components, one of them dominates the other. Let without loss of generality these set be  $H_i$  and  $H_j$  and let  $H_i \rightarrow H_j$ . Then since  $D$  is not-strong  $H_j \nrightarrow H_i$  now let say that  $H_j \rightarrow H_k$ , then since  $D$  is quasi-transitive then either  $H_k \rightarrow H_i$  or  $H_i \rightarrow H_k$ . But since  $H_i \cup H_j \cup H_k$  is not strong  $H_k \nrightarrow H_i$  meaning contracting each  $H_i$  for  $i = 1 \dots, t$  we will have a transitive digraph  $T$  and we have also shown that there are no backwards going arcs in the ordering meaning that  $T$  is not only transitive but acyclic. blablabla  $\square$

From this theorem we can see that quasi-transitive digraphs is totally  $\phi_1$ -decomposable. Since the transitive digraph for the nonstrong quasi-transitive digraphs is acyclic  $T \in \phi_1$  and each  $Q_i$  is in itself strong quasi transitive digraphs and you can therefore use item 2.2.3 again. For the strong quasi-transitive digraphs  $D$ ,  $S$  is semicomplete so  $S \in \phi_1$  and each  $Q_i \in \phi_1$  because it is either one vertex which is a digraph that is both acyclic and semicomplete or it is non-strong and must be quasi-transitive and therefore item 2.2.3 can be used again. So every nonstrong and strong quasi-transitive digraphs is totally  $\phi_1$ -decomposable.

**Theorem 2.2.4.** *quasi decomposition can be found in poly time*

## 2.3 Locally semicomplete

blablabla

**Theorem 2.3.1.** *round decompose locally semicomplete digraph*

Every locally semicomplete digraph can be classified into some other groups of digraphs namely semicomplete digraphs and round decomposable digraphs and the last one which is neither of the two is call evil. Round decomposable digraph  $D = R[D_1, \dots, D_r]$  is where  $R$  is a round digraph of the strong componentents  $D_i$  and  $|R| = r$ .

**Theorem 2.3.2.** [3] *Let  $D$  be a locally semicomplete digraph. Then exactly one of the following possibilities holds. Furthermore, there is a polynomial algorithm that decides which of the properties hold and gives a certificate for this.*

- (a)  $D$  is round decomposable with a unique round decomposition  $R[D_1, \dots, D_r]$ , where  $R$  is a round local tournament on  $r \geq 2$  vertices and  $D_i$  is strong semicomplete digraph for  $i = 1, 2, \dots, r$ .
- (b)  $D$  is evil
- (c)  $D$  is a semicomplete digraph that is not round decomposable.

If the locally semicomplete digraph is nonstrong it turns out that it is decomposable this is called a semicomplete decomposition.

**Theorem 2.3.3.** [3, 1, 4] *Let  $D$  be a nonstrong locally semicomplete digraph and let  $D_1, D_2, \dots, D_p$  be the acyclic order of the strong components of  $D$ . Then  $D$  can be decomposed into  $r \geq 2$  disjoint subdigraphs  $D'_1, D'_2, \dots, D'_r$  as follows:*

$$D'_1 = D_p, \lambda_1 = p,$$

$$\lambda_{i+1} = \min\{j | N^+(D_j) \cap V(D'_i) \neq \emptyset\},$$

and

$$D'_{i+1} = D \langle V(D_{\lambda_{i+1}}) \cup V(D_{\lambda_{i+1}+1}) \cup \dots \cup V(D_{\lambda_{i+1}-1}) \rangle$$

The subdigraphs  $D'_1, D'_2, \dots, D'_r$  satisfy the properties below:

- (a)  $D'_i$  consists of some strong components that are consecutive in the acyclic ordering of the strong components of  $D$  and is semicomplete for  $i = 1, 2, \dots, r$ ;

(a)

Figure 2.1: (a)(b) and (c)

- (b)  $D'_{i+1}$  dominates the initial component of  $D'_i$  and there exists no arc from  $D'_i$  to  $D'_{i+1}$  for  $i = 1, 2, \dots, r-1$ ;
- (c) if  $r \geq 3$  then there exists no arc between  $D'_i$  and  $D'_j$  for  $i, j$  satisfying  $|j - i| \geq 2$

For simplification of 2.3.3 the properties is drawn out in Figure 2.1 Now we focus more on the structure of the evil locally semicomplete digraph which we have not covered yet, there is a fine understanding of the structure of round decomposable and the semicomplete digraphs, even the semicomplete decomposition which is a part of the evil structure too. First we have to recall what a minimal separator from section 1.1, then use this to construct what we call a **good** separator.

**Lemma 2.3.4.** [3] *Let  $S$  be a minimal separator of the locally semicomplete digraph  $D$ . Then either  $D \setminus S$  is semicomplete or  $D \setminus (V - S)$  is semicomplete.*

Then a **good** separator of a locally semicomplete digraph is minimal and  $D \setminus (V - S)$  is not semicomplete. When finding a good separator in a evil locally semicomplete digraph, then the part that is left  $D - S$  a semicomplete decomposition can be found it turns out that there is a lot to say about this decomposition.

**Theorem 2.3.5.** [3, 4] *Let  $D$  be an evil locally semicomplete digraph then  $D$  is strong and satisfies the following properties.*

- (a) *There is a good separator  $S$  such that the semicomplete decomposition of  $D - S$  has exactly three components  $D'_1, D'_2, D'_3$  (and  $D \setminus S$  is semicomplete by 2.3.4);*
- (b) *Furthermore, for each such  $S$ , there are integers  $\alpha, \beta, \mu, \nu$  with  $\lambda_2 \leq \alpha \leq \beta \leq p-1$  and  $p+1 \leq \mu \leq \nu \leq p+q$  such that*

$$N^-(D_\alpha) \cap V(D_\mu) \neq \emptyset \text{ and } N^+(D_\alpha) \cap V(D_\nu) \neq \emptyset, \quad (2.3)$$

$$\text{or } N^-(D_\mu) \cap V(D_\alpha) \neq \emptyset \text{ and } N^+(D_\mu) \cap V(D_\beta) \neq \emptyset, \quad (2.4)$$

where  $D_1, D_2, \dots, D_p$  and  $D_{p+1}, \dots, D_{p+q}$  are the strong decomposition of  $D - S$  and  $D \setminus S$ , respectively, and  $D_{\lambda_2}$  is the initial component of  $D'_2$

Even though this is a structure we can work with, we can actually go deeper into the structure of this evil locally semicomplete digraph. Namely trying to group the components inside the semicomplete decomposition  $D'_1, D'_2, D'_3$  and the good separator  $S$ . This structure is mentioned in [3] but also in [5]. First we can establish this lemma which is a big part of the structure of evil locally semicomplete digraphs.

**Lemma 2.3.6.** [5] *Let  $D$  be an evil locally semicomplete digraph and let  $S$  be a good separator of  $D$ . Then the following holds:*

- (i)  $D_p \Rightarrow S \Rightarrow D_1$ .

(ii) If  $sv$  is an arc from  $S$  to  $D'_2$  with  $s \in V(D_i)$  and  $v \in V(D_j)$ , then

$$D_i \cup D_{i+1} \cup \dots D_{p+q} \Rightarrow D_1 \cup \dots \cup D_{\lambda_2-1} \Rightarrow D_{\lambda_2} \cup \dots \cup D_j$$

.

(iii)  $D_{p+q} \Rightarrow D'_3$  and  $D_f \Rightarrow D_{f+1}$  for  $f \in [p+q]$ , where  $p+q+1=1$ .

(iv) If there is any arc from  $D_i$  to  $D_j$  with  $i \in [\lambda_2-1]$  and  $j \in [\lambda_2, p-1]$ , then  $D_a \Rightarrow D_b$  for all  $a \in [i, \lambda_2-1]$  and  $b \in [\lambda_2, j]$ .

(v) If there is any arc from  $D_k$  to  $D_l$  with  $k \in [p+1, p+q]$  and  $l \in [\lambda_2-1]$ , then  $D_a \Rightarrow D_b$  for all  $a \in [k, p+q]$  and  $b \in [l]$ .

## Chapter 3

# Path cover, Hamilton cycles and pancyclic digraphs

In this chapter the focus is the hamilton cycle problem, where we know that if we can solve the path covering problem then we can solve the hamilton cycle problem, and in the end of this chapter we are short going to cover the pancyclic digraphs.

The hamilton cycle problem and path covering are closely related since if you find a path covering all vertices you can find the hamilton cycle in polynomial time. all this is what we in the first section is going to cover, how to get from a path cover, also called hamilton path, to a hamilton cycle in polynomial time. Then why both problems is NP-hard problems and then shortly why, finding out wheter a graph is pancyclic is NP-hard problem too.

The next section is about path-mergeable digraphs and that locally semicomplete digraphs are a subclass of these and how this helps in the path covering problem and there by the hamilton cycle problem. Then state some theroems that says knowing special things about the graph we know when it contain a hamilton cycle.

The following section is covering quasi-transitive digraphs and when we know there exists a hamilton cycle in those, here the decomposition of the quasi-transitive digraphs is going to be a crucial part of proving this.

Last section for this chapter is going to cover when a quasi-transitive digraphs is pancyclic.

### 3.1 Why hamilton path and cycle problem is NP-Hard

Finding a hamilton cycle in a digraph is a well know problem, but here is a shortly explanation of what that is. When we define what a hamiltonian digraph is we first have to explain what a hamilton cycle is. A hamilton cycle is a directed cycle  $C_H$  in a digraph that contains(pass by) every vertex in the digraph  $\forall v \in V(D), v$  is in  $C_H$ .

**Definition 3.1.1.** *A Hamiltonian digraph is a graph containing a hamilton cycle*

We can also define digraphs called traceable

**Definition 3.1.2.** *A traceable digraph is a digraph containing a hamilton path*

A hamilton path is a path containing all vertices of the digraph.

The problems that is considered **NP-Hard** is finding out whether an arbitrary digraph is



traceable or hamiltonian. We are going to show that hamilton cycle problem is **NP-Hard** by reducing it to a problem we know is. Then we are going to show that if we know that a digraph is traceable it takes polynomial time to figure out wheter it is hamiltonian too, making the traceable problem **NP-Hard** too. Because if the you in polynomial time could figure out wheter a arbitrary digraph is traceable you know that if it is not, it is defenatly not hamiltonian. And if it is you can in polynomial time figure out if it is hamiltonian, making the hamton cycle problem a polynomial time solution problem (not **NP-Hard**).

## 3.2 Hamiltonian Locally semicomplete Digraphs

Recall that a locally semicomplete digraph is both in-locally semicomplete and out-locally semicomplete. Before this gets relevant we are going to introduce a class of digraphs called path-mergeable they are not introduced under section section 1.3 since we are only going to use it in this section. A short explanaiton of a path mergeable digraph is that it is the class of digraphs where given two paths with the start- and endpoint incommen you can merge the two paths into one using all vertices in the two paths. A more formal definition of path mergeable digraphs is if there exists a pair of distinct vertices  $x, y \in V(D)$  and any two disjoint  $(x, y)$ -paths there exists a new path from  $x$  to  $y$  where it is a union of the vertices used in the two vertex-disjoint paths (ending up with a "merge" path of the two given path). These digraphs are easy to regonize with the following corolary we can do it in polynomial time too and the following theroem gives us a nice propertie of path-mergeable digraphs.

**Corollary 3.2.0.1.** [1] *Path-mergeable digraphs can be regonized in polynomial time*

**Theorem 3.2.1.** [1] *A digraph  $D$  is path mergeable if and only if for every pair of distict vertices  $x, y \in V(D)$  and every pair  $P = xx_1 \dots x_r y$ ,  $P' = xy_1 \dots y_s y$ ,  $r, s \geq 1$  of internally disjoint  $(x, y)$ -paths in  $D$ , either there exists an  $i \in \{1, \dots, r\}$ , such that  $x_i \rightarrow y_1$ , or there exists a  $j \in \{1, \dots, s\}$  such that  $y_j \rightarrow x_1$ .*

to explain this 3.2.1 it tells us that for every path mergeable digraph in every two disjoint  $(x, y)$ -path there has to be from one of the path a vertex that dominates the first vertex after  $x$  in the other path. This has to hold for every distict pair of vertices  $x$  and  $y$ . It turns out that in these digraph we can easily determine whether it is a hamiltonian digraph too.

**Theorem 3.2.2.** *A path-mergeable digraph  $D$  of order  $n \geq 2$  is hamiltonian if and only if  $D$  is strong and  $UG(D)$  is 2-connected.*

**Corollary 3.2.2.1.** *There is an  $O(nm)$ -algorithm to decide whether a given strong path-mergeable digraph has a hamiltonian cycle and find one if it exists.*

So it turns out that for path-mergeable digraphs this problem is polynomial solveable, and a subclass of these path-mergeable digraph is namely the locally semicomplete digraphs. If we can prove this we only know that we can solve the hamilton cycle in polynomial time and since the locally semicomplete digraphs is a subclass of in-locally semicomplete digraphs we have an even better time for these.

**Propersition 3.2.3.** *Every locally in-semicomplete (out-semicomplete) digraph is path-mergeable.*

*Proof.* proof this ... ..

□

Then it turns out that 3.2.2 and 3.2.2.1 can be improved if we are only looking at the in-locally semicomplete digraph, since the locally semicomplete digraph is a subclass of these, and it is the ones we are interested in, in this thesis. It turns out that every strong in-locally semicomplete digraph has a 2-connected underlying graph, which means the only thing we need to check is whether it is a strong digraph.

**Theorem 3.2.4.** *A locally in-semicomplete digraph  $D$  of order  $n \geq 2$  is hamiltonian if and only if  $D$  is strong.*

It turns out that when looking at the strong locally in-semicomplete digraphs out of the path-mergeable digraph finding the hamiltonian cycle can be done in polynomial time by theorem discovered by ... ..

**Theorem 3.2.5.** *There is an  $O(m + n \log n)$ -algorithm for finding a hamiltonian cycle in a strong locally in-semicomplete digraph.*

### 3.3 Hamiltonian Quasi-transitive Digraphs

blabla

### 3.4 Pancyclisc Digraphs

blabla

## **Part II**

# **Linkage and weak linkage**

## **Part III**

# **Spanning disjoint subdigraphs (Arc decomposition)**

# Bibliography

- [1] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.
- [2] Jørgen Bang-Jensen and Jing Huang. Quasi-transitive digraphs. *Journal of Graph Theory*, 20(2):141–161, 1995.
- [3] Jørgen Bang-Jensen, Tilde My Christiansen, and Alessandro Maddaloni. Disjoint paths in decomposable digraphs. *Journal of Graph Theory*, 85(2):545–567, 2017.
- [4] Jørgen Bang-Jensen and Jing Huang. Decomposing locally semicomplete digraphs into strong spanning subdigraphs. *Journal of Combinatorial Theory, Series B*, 102(3):701–714, 2012.
- [5] Tilde My Christiansen. *Algorithmic and structural problems in digraphs*. PhD thesis, 2018.