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

I Introduction to Decomposable digraphs and some solutions to Hamilton cycle problem on those digraphs. 3

1 Intro 4

1.1 Graphs and Digraphs 4

1.2 Computational complexity 7

1.2.1 Measure time of algorithm (Polynomial, exponential) 7

1.2.2 NP problems and classifications 7

1.3 Classes of Digraphs 8

1.3.1 introduction to some digraph classes 8

2 Decomposable Digraphs 10

2.1 Genral about Decomposable digraphs 10

2.2 Quasi-transitive 11

2.3 Locally semicomplete 13

3 Path cover and hamilton cycles 16

3.1 Why hamilton path and cycle problem is NP-Hard 16

3.2 Hamiltonian Locally semicomplete Digraphs 17

3.3 Hamiltonian Quasi-transitive Digraphs 19

II Linkage and weak linkage 21

4 Disjoint path in decomposable digraphs 22

4.1 The Linkage Problem 22
4.2 Solving the Linkage Problem in ϕ-decomposable Digraphs 23
4.2.1 linkage for qausi-transitive digraph among other 24
4.3 Solving Linkgae Problem in Locally Semicomplete Digraphs 26

5 Arc-disjoint path in decomposable digraphs 28
5.1 The Weak-Linkage Problem 28
5.2 Solving Weak-Linkage in Quasi-transitive Digraphs 29
5.2.1 k-linkage problem for quasi-transitive digraphs 34
5.3 Solving Weak-Linkgae in Locally Semicomplete Digraphs 34

III Spanning disjoint subdigraphs (Arc decomposition) 37

0.1 Introduction

Why we need graphs.

Part I

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

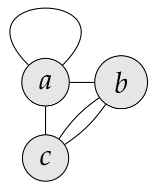
Chapter 1

Intro

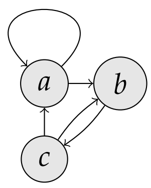
This chapther is if you do not know what a graph is, there is some notation this thesis may use differently then others and different articles are doing. Then in section 1.2 we are going to cover runing time principles and problems that in general are hard to solve, called NP-Complete problems and what these have to do with this thesis. Last section 1.3 is going to introducing the names of all the 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 respectivlely. We define the **size** of the graph to be the number of vertices $|V|$ this is also known as **cardinality** of V. An **edge** $e \in E$ is where $e \equiv (a, b)$ and $\{a, b\} \subseteq V$ we then say e is an edge in G, e is in this case called **incedent** to a and b. We call $a, b \in V$ **adjecent** if there is an edge (a, b) or (b, a) (two given vertices connected by an edge is said to be adjecent). If an edge goes from and to the same vertex (a, a) it is called a **loop**. The set of edges $e_1, \ldots, e_k$ is usally describe whit the letter E where each edge contains a pair of vertices that are adjecent. The letter V is to denoted the set of vertices in the given graph.
In a graph we have something called a **walk** which is a repeting ordering of vertices and 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 an edge (a, b)). by repeting it means that a vertex can apear twice in a walk. 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 apear one time. A **cycle** is a closed walk where the only vertex pressent more then one time is the first vertex(also called a closed path). Let X be a subset of the vertices $X \subseteq V$ then we say that $V \backslash X$ is the set of vertices with out the vertices in X, i.e. $V \backslash X \equiv V - X$. 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 to be a subgraph.
As we can look at subsets we sometimes need to look at sub-paths, for a path $P = x_1 \ldots x_k$ a **sub-path** is a path $P' = x_i \ldots x_j$ of P where $1 \leq i < j \leq k$.



(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 adjecent.



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

Before delving more specific into graphs and digraphs we must establish some important 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.

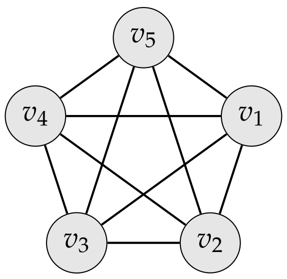


Figure 1.2: Complete graph with 5 vertices.

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.
Somtimes when looking at specifiks set of vertices we are actually interested in something called **independents set** which is a set of vertices of G where there is no edge between the vertices in the set. A maximal independet set of G is a independent set where you can not add any new vertex in the set that is not adjencent to any vertex in the set(adding a vertex makes the set no longer independent). A maximum independent set is the maximal independent set with greatest cardinality. Let $I \subset V$ be a maximum independet set then $|I|$ is called the **independence number**.

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 adjecent 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 $D(V, A)$ which is what we in this project are focusing on see Figure 1.1b(as G denote a **G**raph, D denote a **D**igraph).
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 $A \Rightarrow B$.

Sometimes when working with a digraph or solving a problem we have a subset of vertices $X \subseteq V(D)$ that want to work with as one vertex. Then we **contract** the vertecies X into one vertex x where $N^+(X)\backslash X = N^+(x)$ and $N^-(X)\backslash X = N^-(x)$ (so we only keep the ingoing and outgoing arcs of X and delete all vertices of X and the arcs inside). When we contract X of D we will try useing the notation D/X. There is also another kind of contraction where you also delete possible multiple arcs, if this is the case it will be explained in the section.

In a digraph we have something called the **underlying graph** denoted $UG(D)$. 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 the subdigraphcontains all the vertices in X and all arcs in $A \in G$ where both head and tail is incedent to the vertices in X. We will denote the graph $D\langle V\backslash X\rangle$ 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 to u. 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 where $|V| > k$ and a minimal seperator S is where $|S| = k$. In the same way we can define k**-arc-strong** digraph is where you need to delete at least k arcs for the digraph to no longer be strong.

In a digraph $D(V, A)$ we mostly use the **dregree** 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**, $\delta^-(D) = \min\{d^-(v)|v \in V\}$ somtime we are going to need the minimum of these to $\delta(D) = \min\{\delta^+(D), \delta^-(D)\}$ called the **minimum degree**. For every vertex v the vertices that is **adjacent** with v is called **nieghbours** of v. We denote $N^+(v)$ and $N^-(v)$ as the set of vertices that is dominated by (**out-nieghbours** of) v and dominates (**in-nieghbours** of) v, respectivly. 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 specifing the head an tail of the path, so a path from s to t is denoted as an **(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 $\lambda_D(s, t)$

Theorem 1.1.1. *mengers thm*

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 θ. 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 gruops of how long time it take to decide or verify the problems solution. a problem that is decided i polynomiel 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 polynomiel 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.*

Obiously 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 $NP - Hard$ but before we can explain that we need to explain what it means for a problem to be polynomial reduceable 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 a algorithm exists it is called a polynomial verifier and we say that A is **polynomial reducable** to B or just that A is **reduced** to B. **NP-Hard** are the class of problems that every NP problem can be polynomiel reduced to. A problems in the class of NP-Hard problems does not nessesarily mean that it is NP it-self. If a problem is both **NP** and **NP-Hard** we call it **NP-Complete**. The problems we are mostly focusing on is in the class of **NP-Complete** problems.

1.3 Classes of Digraphs

We can classefy specific collection of graphs the reason for this is that digraphs of smaller collections of digraphs (like tournaments is a smaller collection of semicomplete digraphs) might be because of problems that is hard to solve on general digraph but is easy/polynomial solvable on specific types of digraphs.

A group of these problems is called NP-complete problems which sometimes sound easy solvable 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 des 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 checks wheter a digraph belongs to a certion **class** of digraphs. A class of digraph is a collection of digraph with certain properties incommen 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 is also in it self a classification of digraphs. Classes of digraphs can be overlapping each other or be fully containt in each other like tournaments is fully containt 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 vertecies (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 partites 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 then 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 $V = v_1, v_2, \ldots, v_n$ 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 an non-strong digraph such that the ordering of the componentent $C_1, C_2, \ldots C_k$ 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 $a, d, c \in V$ 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 forall vertices $a, d, c \in V$ where the arc (a, b) and b, c is present in the digraph ($\in A$), a and c has to be adecent 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 thises since the graph is also what we call **decomposable**.
Decomposable digraphs is also a clasification of graphs which are decomposable, for a graph D to be decomposable we have $H_1, H_2, \ldots, H_k$ **houses** and S where $V(S) = s_1, s_2, \ldots, s_k$ which are all digraphs by them self but if each s_i is replaced by the digraph H_i $i = 1, 2, \ldots, k$ 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 $D = S[H_1, H_2, \ldots H_k]$. This is the class of digraphs we are focusing on in this thises. If all the houses are independent sets we call $D = S[H_1, H_2, \ldots, H_k]$ the extension of S. If S is a semicomplete digraph we call the extensin 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 giong 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-nieghboor of a vertex $x \in V$ they have to be adjecent ($x \cup N^-(x)$ 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-nieghboors, $N^+(x)$, has to be adjecent. 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 decomposeble will be described in section **??**.
The last class of digraph that are important for this thises is the round digrphs. A digraph is called a **round** digraph if there exists an ordering of the vertices $v_1, v_2, \ldots, v_n$ such that for all v_i, $N^+(v_i) = v_{i+1}, v_{i+2}, \ldots, v_{i+d^+(v_i)}$ and $N^-(v_i) = v_{i-d^-(v_i)}, v_{i-(d^-(v_i)-1)}, \ldots, v_{i-1}$.

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 ϕ_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 chapthers.

2.1 Genral about Decomposable digraphs

Recall that a decomposable digraph D can be decomposed into a main graph S (also sometimes called **quotient** graph) 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 replaces v_i). The arcs between the houses is as follows $H_i \to H_j$ in D if $v_i \to v_j$ 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 $X \to Y$. 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. $H_i \to H_j$ and $H_j \to H_i$).

Decomposable digraphs can be classed by a set of digraphs ϕ, when
$D = S[H_1, H_2, \dots, H_k]$ it is **ϕ-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 ϕ-decomposable but the class of **totally ϕ-decomposable** digraphs is where D is ϕ-decomposable and each H_i is totally ϕ-decomposable. We are going to make two shuch sets of digraphs ϕ_1 which is the union of semicomplete digraph and acyclic digraph both classes deskribed in section 1.3 and ϕ_2 which is the union of semicomplete and round digraphs also deskribed in section 1.3.

$$\phi_1 = \text{Semicomplete digraphs} \cup \text{Acyclic digraphs} \tag{2.1}$$

$$\phi_2 = \text{Semicomplete digraphs} \cup \text{Round Digraphs} \tag{2.2}$$

Take these sets ϕ_1 and ϕ_2 then for every induced subdigraph of a digraph D where either $D \in \phi_1$ or $D \in \phi_2$ then the induced digraph is agian in the same set (so if $D \in \phi_1$ the induced subdigraph is in ϕ_1, same goes for ϕ_2). When this is true for a set ϕ the set is called **hereditary**. So both ϕ_1 and ϕ_2 is hereditary.

Lemma 2.1.1. *Let ϕ be a hereditary set of digraphs. If a given digraph D is totally ϕ-decomposable, then every induced subdigraph D' of D is totally ϕ-decomposable.*

It also turns out that for ϕ_1 and ϕ_2 there exists an algorithm that checks for a digraph D is totally ϕ_i-decomposable ($i = 1, 2$).

Theorem 2.1.2. *ţtebanggutin There exists an $O(n^2m + n^3)$-algorithm for chekking if a digraph with n vertices and m arcs is totally ϕ_i-decomposable for $i = 1, 2$.*

and $O(n^2m + n^3)$ is clearly polynomial algortihm.

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 $x \to y$ (x dominates y) and $y \to z$ (y domitaes z), then there has to be at least one arc in either dirction between x and z. When working with quasi-transitive digraphs there are many things you can depend on, things that the structure has already diceded 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 $A \to B$.*

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 $S \to S'$ or $S' \to S$ or both $S \to S'$ and $S' \to S$ in which case $|V(S)| = |V(S')| = 1$.

These to lemmas is 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 $H_1, \ldots, H_t$ such that $D = T[H_1, \ldots, H_t]$.*

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 theroem is also what we are going to use more then ones, to prove several of the problem solving theorems through out this thises.

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 there 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 is no arcs going back in the ordering, but we will prove that now.
Now from Theorem 2.2.1 we know that if there is an arc between two of the strong components, one of them dominates the other. Let with out loss of generality these set be H_i and H_j and let $H_i \to H_j$. Then Since D is not-strong $H_j \not\to H_i$ now let say that $H_j \to H_k$, then since D is quasi-transitive then either $H_k \to H_i$ or $H_i \to H_k$. But since $H_i \cup H_j \cup H_k$ is not strong $H_k \not\to H_i$ 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. This end the proof of the non-strong quasi-transitive digraph leving only the strong ones left.

Now suppose that D is a strong quasi-transitive digraph, we now look at the underlying graph $UG(D)$ after this we find the complement of it, $\overline{UG(D)}$ since D is strong we know from Theorem 2.2.2 that $\overline{UG(D)}$ is disconnected, so we find $Q_1, \dots, Q_s$ where $\overline{UG(Q_i)}$ is connected in $\overline{UG(D)}$ $\forall i \in [s]$.
Since these subdigraphs $\overline{UG(Q_i)}$ of $\overline{UG(D)}$ is connected we know that Q_i is non-strong or a single vertex in D. From the same lemma each Q_i (reprecent S in Theorem 2.2.2) which means when contracting Q_i $\forall i \in [s]$ into a single vertex q_i. Denote D with contracted Q_i's as S. We have that every pair of vertex in S have one arc between in either direction or one in both direction making S semicomplte.
This concludes the proof. □

From this theorem we can see that quasi-transitive digraphs is totally ϕ_1-decomposable. Since the transitive digraph for the nonstrong quasi-transitive digraphs is acyclic $T \in \phi_1$ and each H_i is itself strong quasi-triansitive digraphs and you can therefore use item 2.2.3 agian. 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 agian. So every nonstrong and strong quasi-transitive digraphs is totally ϕ_1-decomposable. could not find a therom lemma or anything else so i made my own corolary.

Corollary 2.2.3.1. *quasi-transitive digraphs D are totally ϕ_1-decomposable and you can find the decomposition in polynomial time.*

The polynomial time comes from Theorem 2.1.2 since it is totally ϕ_i-decomposable where $i = 1$.

2.3 Locally semicomplete

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 $|R| = r$. First we need to recal from section 1.3 what a round digraph is and we use the definition from [4].

Definition 2.3.1. *[4] A digraph on n vertices is round if we can label its vertices $v_1, \dots, v_n$so that for each i, we have $N^+(v_i) = \{v_{i+1}, \dots, v_{i+d^+(i)}$ and $N^-(v_i) = \{v_{i-d^-(i)}, \dots, v_{i-1)}$. We call the labeling $v_1, \dots, v_n$ a round ordering.*

So the class of locally semicomplete digraph is split up in these 3 subclasses and these subclasses are going to be important for proving a lot of the problems we are going to corver in this thises.

Theorem 2.3.1. *[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 thet 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 2.3.2. *[3, 1, 5] 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

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

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 $1 = 1, 2, \dots, r$;

(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 $|j - i| \geq 2$

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

For simplification of Theorem 2.3.2 the properties is drawn out in Figure 2.1 If D is a locally semicomplete digraph that is not semicomplete, it have a round-decomposition $D = R[D_1, D_2, \dots, D_r]$ where R is a round digraph but it can also be either of these, then we as above call it evil. If it is a strong digraph we can have a seperator S, we have left a non-strong locally semicomplete digraph and therefore we can make a semicomplete decomposition out of $D - S$.

Theorem 2.3.3. *[1] If a strong locally semicomplete digraph D is not semicomplete then there exists a minimal seperating set $S \subseteq V$ such that $D - S$ is not semicomplete. Furthermore, if $D_1, D_2, \dots, D_p$ is the acyclic ordering of the strong componenst of $D - S$ and $D'_1, D'_2, \dots, D'_r$ is the semicomplete decomposition of $D - S$, then $r \geq 3$, $D\langle S \rangle$ is semicomplete and we have $D_p \mapsto S \mapsto D_1$.*

Some of then are round-decomposable that will as we will later see the once where $r > 3$.

Propersition 2.3.4. *Let $R[H_1, H_2, \dots, H_\alpha]$ be round decomposition of a strong locally semicomplete digraph D. Then for every minimal seperating set S, there are two integers i and $k \geq 0$ such that $S = V(H_i) \cup \cdots \cup V(H_{i+k})$.*

It also turns out that this round-decomposistion is unique.

Corollary 2.3.4.1. *If a locally semicomplete digraph D is round decomposable, then it has a unique round decomposition $D = R[D_1, D_2, \dots, D_\alpha]$.*

Proof. mabey □

From Theorem 2.3.1 we know that for a round-decomposable digraph the quotient graph is round and the houses are semicomplete making them totally ϕ_2-decomposable then by Theorem 2.1.2 we know that we can find the decomposition in polynomial time.

Propersition 2.3.5. *There exists a polynomial algorithm to decide whether a given locally semicomplete digraph D has a round decomposition and to find this decomposition if it exists.*

Like we shortly meansioned above the locally semicomplete digraph that are not semicomplete and not round have a semicomplete decomposistions on where $r = 3$ also those we call evil. The evil locally semicomplete digraphs are the once we are focusing on for the rest of this section.

Lemma 2.3.6. *Let D be a strong locally semicomplete digraph which is not semicomplete. Either D is round decomposable, or D has a minimal seperating set S such that the semicomplete decomposistion of $D - S$ has exactly three components D'_1, D'_2, D'_3.*

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. We are now going to constructthen use this to construct what we call a **good** seperator.

Lemma 2.3.7. *[3] Let S ba a minimal seperator of the locally semicomplete digraph D. Then either $D\langle S\rangle$ is semicomplete or $D\langle V-S\rangle$ is semicomplete.*

Then a **good** seperator of a locally semicomplete digraph is minimal and $D\langle S\rangle$ is semicomplete. When finding a good seperator in a evil locally semicomplete digraph, then the part that is left $D-S$ is a non-strong locally seimcomplete digraph and we can therefore use Theorem 2.3.2 to find the semicomplete decomposition of $D\langle V-S\rangle$ it turns out that there is a lot to say about this decomposition. With this decomposistion we can classify the quotient graph but we can try to deskribe more deebly how it looks.

Theorem 2.3.8. *[3, 5] Let D be an evil locally semicomplete digraph then D is strong and satisfies the following properties.*

(a) There is a good seperator S such that the semicomplete decomposition of $D-S$ has exactly three components D'_1, D'_2, D'_3 (and $D\langle S\rangle$ is semicomplete by Theorem 2.3.7);

(b) Furthermore, for each such S, there are integers α, β, μ, ν with $\lambda_2 \le \alpha \le \beta \le p-1$ and $p+1 \le \mu \le \nu \le p+q$ such that

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

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

where $D_1, D_2, \ldots, D_p$ and $D_{p+1}, \ldots, D_{p+q}$ are the strong decomposition of $D-S$ and $D\langle S\rangle$, 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. Namly trying to group the components inside the semicomplete decomposition D'_1, D'_2, D'_3 and the good seperator S. This structer is menation in [3] but also in [6]. First we can establish this lemma which is a big part of the structure of evil locally semicomplete digraphs.

Lemma 2.3.9. *[6] Let D be an evil locally semicomplete digraph and let S be a good seperator od 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

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

.

(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 and hamilton cycles

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.
The hamilton cycle problem and path covering are closly related since if you find a path covering all vertecies 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 and then why both problems is NP-hard problems.
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 hamilton cycle problem. 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.

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 explanaition 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 trecable or hamiltonian. Before going into why the problems are NP, we are going to state some obvious conditions for graphs to be traceable or hamiltonian.
For a digraph to be traceable it needs to be semi connected and for a digraph to be hamiltonian it needs to be strong. And since hamilton cycle is a cycle factor a digraph that is hamiltonian ofcourse need to have a cycle factor.

We are going to show (shortly explain) why the hamilton path problem is NP-Hard by reducing a problem we know is NP to the hamilton path problem. 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 we 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).

Theorem 3.1.1. *Finding out wether a digraph is trecable is a NP-Hard Problem*

Proof. sketz blbalbalbalablabala □

Theorem 3.1.2. *Finding out wether a digraph is hamiltonian is a NP-Hard Problem*

Proof. sketz blbalbalbalablabala □

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.
A short explanaition 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.

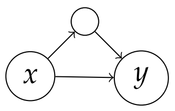
Corollary 3.2.0.1. *[1] Path-mergeable digraphs can be regonized in polynomial time*

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 $j \in \{1, \dots, y_j \rightarrow x_1\}$.*

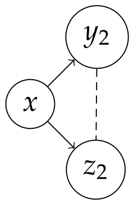
to explain this Theorem 3.2.1 it tells us that for every path mergeable digraph in every two disjoint (x, y)-path there has to be from one of the path a vertex that dominates the first vertex after x in the other path. This has to hold for every distict pair of vertices x and y.
It turns out that in 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.*



(a) An visuel example of two vertex disjoint paths P and Q where $|A(P)| + |A(Q)| = 3$.



(b) Clearly from the definition of out-semicomplete the dashed line need to be an arc in either direction or both directions.

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 do not only know that we can solve the hamilton cycle in polynomial time since the locally semicomplete digraphs is a special subclass of path-mergeable digraph we have an even better time for these.

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

Proof. First we prove its true for out-semicomplete and there after for in-semicomplete digraphs. So lets assume that D is an out-semicomplete digraph, and we take x and y where we say that these two have 2 vertex disjoint (x, y)-paths called P and Q.
Let $P = y_1 y_2 \dots y_k$ and $Q = z_1 z_2 \dots z_s$ where $y_1 = x = z_1$ and $y_k = y = z_s$. We want to show that there exists a path R where $V(R) = V(P) \cup V(Q)$, if $|A(P)| + |A(Q)| = 3$ it is clear from Figure 3.1a that we can just choose the longest of the paths and we have all vertex included from both paths. So we assume that $|A(P)| + |A(Q)| \geq 4$, and ince D is out-semicomplete we know that either $y_2 \to z_2$ or $z_2 \to y_2$ or both has to be true. For conformation see Figure 3.1b. This must be true forevery pair of vertices x and y where there is two distict (x, y)-paths. The rest of this part of the proof is from Theorem 3.2.1 which conlude the proof for out-semicomplete.

Now suppose that D is in-semicomplete then reveersing the arcs will make it out-semicomplete denoted this digraph D-revers. Now for two distict vertices x and y where there exists two distict (x, y)-path P and Q in D, then in D-revers their must exists two distinct (y, x)-paths P-revers Q-revers.
Since D-reverse is out-semicomplete we can find a path R where $V(R) = V(P\text{-revers}) \cup V(Q\text{-revers})$ in D-revers. Then in D we have a (x, y)-path R-revers where $V(R\text{-revers}) = V(P) \cup V(Q)$. Making every in-semicomplete digraphs path-mergeable. □

Then it turns out that Theorem 3.2.2 and Corollary 3.2.2.1 can be imporved 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 thises. It turns out that every strong in-locally semicomplete digraph has a 2-connected underlyning 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 i polynomial time by theorem discorvered by ... ...

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

This ends the section about hamiltonian locally semicomplete digraphs, now we want to know about traceable locally semicomplete digraph.
First we need to know that an **out-branching** for a digraph D is where you have a vertex as the root of this branching and arcs only going out of this for all other vertex they have only one arc going in and arcs going out is ≥ 0. An in-branching is the above explaination where all arcs are reversed.

Theorem 3.2.6. *Every connected locally in-semicomplete digraphs D has an out-branching*

Theorem 3.2.7. *A locally in-semicomplete digraph is traceable if and only if it contains an in-branching*

This also means that reversing the arcs that a locally out-semicomplete digraph is traceable if and only if it contains an out-branching. if this out-branching or in-branching exists for locally out-semicomplete or locally in-semicomplete digraphs respectively we want to find the longest path in these and then we have the wanted hamilton path.

Theorem 3.2.8. *A longest path in a locally in-semicomplete digraph D can be found in time $O(m + n\ logn)$.*

And again this is also true for locally out-semicomplete digraphs. Connecting Theorem 3.2.7 and Theorem 3.2.6 we know that a locally semicomplete digraph is both locally in-semicomplete meaning from Theorem 3.2.6 in contains an out-branching if it is connected and it is locally out-semicomplete. So if it contains an out-branching it is traceable.

Theorem 3.2.9. *A locally semicomplete digraph has a hamiltonian path if and only if it is connected.*

And this path can be found in time $O(m + n\ \log n)$ from Theorem 3.2.8.

3.3 Hamiltonian Quasi-transitive Digraphs

First of all we have to recall item 2.2.3 since it is the key theorem to solve the hamiltonian problem in polynomial time.
Remember that a condition for a digraph to be hamiltonian is that it need to be strong, so for finding a hamilton cycle in a quasi-transitive digraph, we are not interested in the non-strong digrpahs. Leving only the strong quasi-transitive digraphs with decomposition $S[Q_1, \ldots Q_s]$ from item 2.2.3. The given decomposition of strong quasi-transitive digraphs has a semicomplete digraph as the quotient. This is why we need some inside to these before the main solution in this subsection can be proven. another composition of semicomplete digraphs is the extension of these, called extended semicomplete digraph. An extension of a digraph is a composition of the given digraph S where the hauses of the composition is either a single vertex or independence sets.
Before we explain when we can find a hamilton cycle in strong quasi-transitive digraphs we need to recall what a cycle factor is. From section 1.1 we shortly explain that a cycle factor is when we can find $C_1, \ldots C_k$ cycles in D contaning all vertices of D.

Theorem 3.3.1. *An extended semicomplete digraph D is hamiltonian if and only if D is strong and contains a cycle factor. One can check whether D is hamiltonian and construct a Hamilton cycle of D (if one exsists) in time $O(n^{2.5})$.*

Theorem 3.3.2. *A strong quasi-transitive digraph D with a canonical decomposition $D = S[Q_1 \dots, Q_s]$ is hamiltonian if and only if it has a cycle factor $\mathcal{F}$ such that no cycle of $\mathcal{F}$ is a cycle of some Q_i.*

Proof. Since a hamiltonian cycle need to cover all vertices in a digraph, we know that it must cross every Q_i. Moreover the hamilton cycle is a cycle factor not fully containt in any Q_i. So we only need to show that if we have a cycle factor $\mathcal{F}$, where no cycle is in any Q_i, then D is hamiltonian. $\forall i\ V(Q_i) \cap \mathcal{F} =$
$mathcalF_i$, there can not be any circle in this and since every vertex is in $\mathcal{F}$ all vertices in Q_i must be containt in $\mathcal{F}_i$ and there is no cycle containt in $\mathcal{F}_\rangle$ whcich makes it a path factor of Q_i.
Figure here
For all paths in $\mathcal{F}_i$ we make a path contraction. After contraction or before we delete the remaining arcs if this is done before its the arcs going from the end of a path to a begining of an other path. This action will make Q_i an independent set $\forall i \in [s]$. Since S is a semicomplete digraph our new digraph would then because of the independence of each Q_i after the action be an extended semicomplete digraph S'. Since we have only made path contractions along the cycles in the cycle factor of D and not deleted any arcs that are a part of the cycle factor S' contains a cycle factor. Then by Theorem 3.3.1 we know that S' contains a hamilton cycle. Adding the deleted arcs does not change this insert a path instead of a node just makes the cycle longer but it still contains every vertex given a hamilton cycle in D □

A hamilton path does not have the same condition for a digraphs to be strong meaning we are also interested in the non-strong quasi-transitive digraphs $T[H_1, \dots, H_t]$. The next theorem is proven in much the same as **??**.

Theorem 3.3.3. *A quasi-transitive digraph D with at least two vertices and with canonical decomposition $D = R[G_1, G_2, \dots, G_r]$ is traceable if and only if it has a 1-path-cycle factor $\mathcal{F}$ such that no cycle or path of $\mathcal{F}$is completely in some $D < V(G_i) >$.*

We know that the canonical decomposition of a quasi-transitive digraph can be found i polynomial time. We can also find the hamilton cycle in a quasi-transitive digraph in polynomial time, but also verify if it does not exists for the given graph. This result was proved by Gutin. ...

Theorem 3.3.4. *There is an $O(n^4)$ algorithm which, given a quasi-transitive digraph D, either returns a hamiltonian cycle in D or verifies that no such cycle exists.*

Part II

Linkage and weak linkage

Chapter 4

Disjoint path in decomposable digraphs

In this chapter we will cover the Linkage problem given a set a decomposable digraph D and a set of terminals Π to be linked. We will explaining the problem and shortly show why the problem is interresting (NP-complete). After this we will in section 4.2 show that the linkage problem is polynomial solveable in ϕ-decomposable digraph as long as ϕ is a linkage ejector after that we will show that ϕ_1 is a linkeage ejector meaning that the linkage problem is solveable for quasi-transitive digraphs(all of this is only true if the number of pairs needed linking is fixed). Then in section 4.3 we will cover the 3 different subclasses a locally semicomplete digraph can be a part of, and solve the k-linkage problem for those in polynomial time for a fixed k.

4.1 The Linkage Problem

Given a digraph D and two distinct vertices s and t we want to make a path from s to t denoted this P. Recall that in this case s will be the source of P and t the zink. This we sould be able to do easily if one exists, but when adding two extra distict vertices u and v not nessesarily distinct from s and t and we want a path Q between u and v distinct from the path P the problem suddenly become NP-complete. This prolem is what we call the **2-linkage** problem, we can replace 2 with an arbitrary number k and we then call it the **k-linkage** problem or just **the linkage problem**. the vertices s, t, u and v are called **terminals** and (s, t), (u, v) are called **terminal pairs**.

Theorem 4.1.1. *linkage NP-complete*

skezt. blblablbalablablabala □

The notation for this problem in this thesis would be using k as the natruel number of pairs of terminals, and the set of these terminals is denoted $\Pi = \{(s_1, t_1); \ldots ; (s_k, t_k)\}$. As we have done optil now we will still use D as the main digraph we are looking at unless anything else is specified. L is used as a colloction of paths $P_1, \ldots, P_l$ if L is the solution to our linkage problem it means $l = k$ and and the paths P_i links the pair (s_i, t_i) for all $i \in [k]$. If L upholds the above conditions we say that L is a Π-linkgae, or L is the linkage of (D, Π).

Recall that a quasi-transitive digraph is build up by either a transitive acyclic digraph or semicomplete digraph as the quotient of the decomposition. And for these to classes of digraph we kan solve the k-linkage problem in polynomial time for a fixed k. With fixed k there means that an algorithm given a digraph and a naturel number k can solve the k-linkage problem(it is possible that the algorithm needs more information). When k is not fixed then it is already NP-complete for tournaments, since tournaments is a very strict class we will only focus on when k is fixed.

4.2 Solving the Linkage Problem in ϕ-decomposable Digraphs

From item 2.2.3 we know that a quasi-transitive digraph is a composition of acyclic transitive digraphs and semicomplete digraphs. We know that ϕ_1 is the union of acyclic and semicomplete digraphs, which means that every quasi-transitive digraphs are ϕ_1-decomposable as described in chapter 2.

Theorem 4.2.1. *[3] For every fixed k, there exists a polynomial algorithm for the k-linkage problem on acyclic digraphs.*

Theorem 4.2.2. *For every fixed k, there exists a polynomial algorithm for the k-linkage problem on semicomplete digraph.*

Note that this means that there exists polynomial algorithms for a fixed k to solve the k-linkage problem for digraphs in ϕ_1.
For a decomposition $D = S[M_1, \dots M_s]$ and a set of terminal pairs, we can split the set into two different sets of terminals. The set of **internal pairs** Π_i, where internal pair means that both s_i and t_i is in the same hause, and the set of **external pairs** Π_e which is the rest such that $\Pi = \Pi_i \cup \Pi_e$.

Lemma 4.2.3. *[3] Let $D = S[M_1, \dots, M_s]$ be a decomposable digraph and Π a set of pairs of terminals. Then (D, Π) has a linkage if and only if it has a linkage whose external paths do not use any arc of $D\langle M_i \rangle$ for $i \in [s]$.*

Proof. One of the way is trivial since a linkage where external paths uses no arcs inside any house is still a (D, Π)-linkage. So Now we assume that L is a (D, Π)-linkage that uses the smallest amount of vertices possible. We claim that no external path of L uses any arcs inside any house. Now we assume that this is not the case, then there must exist a path $P \in L$ where an arc uv of P is containt in a house $uv \in A(P) \cap A(D\langle M_i \rangle)$ for some $u, v \in V(P)$ and some $i \in [s]$.
Since P is external there is at least one vertex outside the house, ($z \in V(P) - V(M_i)$) either zu or vz is an arc of P. Without loss of generality say vz is the arc then since v and u are in the same hause $uz \in A(D)$and we can make $P' = P - \{uv, vz\} + uz$, Then we can construct a new linkage $L' = L - P + P'$ which indeed is a (D, Π)-Linkage with $V(L') < V(L)$ which is a contradiction since L was suppose to be the linkage with the smallest number of vertices. (for formality say zu was the arc then $P' = P - \{zu, uv\} + zv$ and $L' = L - P + P'$a (D, Π)-linkage where $V(L') < V(L)$). □

Meaning that the external paths do not use arcs inside the hauses only arcs to move from house to house (arcs from the quotient digraph S). Be aweare that internal pairs can be linked by an internal path or an external path going out of the house and later in agian, where ofcourse external pairs has to be linked by external paths.

Before getting into the algorithm for solveing the k-linkage problem for ϕ-decomposable digraphs, we need to set some conditions for the set ϕ. When a set of digraphs ϕ upholds these conditions we are going to say that ϕ is a linkage ejector. But first we need to establish that a set of digraphs can be closed with respect to blow-up. **blow-up** means blowing up a vertex v, with a digraph K(Replacing v with the digraph K). When a set of digraphs ϕ is closed with respect to this operation it means that for a digraph $D \in \phi$ there exists a digraph K such that after K has replaced v the digraph is still a part of the set ϕ. This definition brings this nice lemma.

Lemma 4.2.4. *If a class ϕ is closed with respect to the blowing-up operation $S \in \phi$ and $D = S[M_1, \dots M_s]$, then it is possible to replace the arcs in the digraph M_i with other arcs, so that the resulting digraph is in ϕ.*

This brings us to the definition of a linkage ejector. This definition is a reformulation of the one given in article [3].

Definition 4.2.1. *[3] A class of digraphs ϕ that is closed with respect to blow-up is a linkage ejector if the following conditions is true*

1. *There exists a polynomial algorithm $\mathcal{A}_\phi$ to find a total ϕ-decomposition of every totally ϕ-decomposable digraph.*
2. *There exists a polynomial algorithm $\mathcal{B}_\phi$ for a fixed k, for solving the k-linkage problen on ϕ*
3. *There exists a polynomial algorithm $\mathcal{C}_\phi$ that given a totally-decomposable digraph $D = S[M_1, \dots, M_s]$ constructs a digraph of ϕ by replacing the arcs inside each M_i for $i \in [s]$ as in Theorem 4.2.4.*

Algorithm here

4.2.1 linkage for qausi-transitive digraph among other

To prove that for qausi-transitive digraphs we can solve the linkage problem in polynomial time, we just need to prove that ϕ_1 is a linkage ejector. Since extended semicomplete digraphs and other classes is also a part of the totally ϕ_1-decomposable digraphs, this will then also prove that the linkage problem can be solved in polynomial time for these.

Lemma 4.2.5. *[3] The class ϕ_1 is a linkage ejector*

Proof. First we have to make sure that ϕ is closed with respect to blow-ups. If we blow-up the vertices in with a transitive tournament. Then if $D \in phi_1$ is semicomplete, then since tournaments is semicomplete D after blow-up is still semicomplete. Then if $D \in \phi$ is acyclic and the vertices is blownup by a transitive tournament then it is still acylic since a transitive tournament is acylic Such ϕ_1 is closed to blow-ups if the digraphs that the vertices is blown-up with is a transitive tournament. From Theorem 2.1.2 we have the polynomial algorithm $\mathcal{A}_{\phi_1}$ meaning we only need the function $\mathcal{B}_{\phi_1}$ and $\mathcal{C}_{\phi_1}$.
The algorithm $\mathcal{B}_{\phi_1}$ is a algorithm that determines the k-linkage problem for a fixed k on digraphs in ϕ_1 by Theorem 4.2.1 we have a polynomial algorithm for acyclic digraph and by

Procedure 1 Algorithm $\mathcal{M}$ for k disjoint paths

Input: Digraph D, two natural numbers k and k' where $k' \leq k$, a list of k' terminal pairs Π, A set of arcs $F \subseteq A(D)$ satiesfying:

$$d_F^-(v), d_F^+(v) \leq k - k' \; \forall v \in V(D)$$
$$|F| \leq (k-k')2k$$

Output: Either "NO" or "YES"

```
1: if $\Pi = \emptyset$ then
2:     output YES
3: end if
4: Run $\mathcal{A}_\phi$ to find a total $\phi$-decomposition of $D = S[H_1, \dots, H_s]$.
5: if this decomposition is trivial that is $D \in \phi$ then
6:     run $\mathcal{B}_\phi^-$ solve the problem.
7: end if
8: Let $\Pi^e \subset \Pi$ ($\Pi^i \subset \Pi$) be the list of external (internal) pairs $(s_q, t_q) \in \Pi$.
9: Assume that $M_1, \dots, M_l$ is the houses with the internal pairs.
10: for every partion of $\Pi^i = \Pi_1 \cup \Pi_2$ look for external paths linking the pairs in $\Pi^e \cup \Pi_1$
    and internal pairs in $\Pi_2$ do
11:     if $\Pi^e \cup \Pi_1 = \emptyset$, then for $i = 1, \dots, l$: then
12:         run $\mathcal{M}$ recursively on input $(D\langle M_1\rangle, \Pi_2 \cap (V(M_1) \times V(M_1))), \dots, (D\langle M_l\rangle, \Pi_2 \cap$
            $(V(M_l) \times V(M_l)))$.
13:         if all are linked then
14:             output YES
15:         end if
16:     end if
17:     if $\Pi^e \cup \Pi_1 \neq \emptyset$ then
18:         for each possible choice of $l$ vertex sets $(V_1, \dots V_l)$ and nonnegative numbers
            $n_1, \dots, n_l \leq k$ such that $|V_i| = n_i$ and $V(\Pi^e \cup \Pi_1) \cap V(M_i) \subseteq V_i \subseteq V(M_i) - V(\Pi_2)$
            do
19:             let $S' \in \phi$ be the the result of running the algorithm $\mathcal{C}_\phi$ on
                $S[I_{n_1}, \dots, I_{n_l}, M_{l+1}, \dots, M_s]$, where $I_{n_j}$ is the digraph on $n_j$ vertices with no arcs
                $(V(I_{n_j}) = V_j)$.
20:             Run $B_\phi^-$ on $(S', \Pi^e \cup \Pi_1)$.
21:             if $\Pi^e \cup \Pi_1$ is linked then
22:                 run $\mathcal{M}$ recursively on input $(D\langle V(M_1) - V_1\rangle, \Pi_2 \cap (V(M_1) \times$
                    $V(M_1))), \dots, (D\langle V(M_l) - V_l\rangle, \Pi_2 \cap (V(M_l) \times V(M_l)))$.
23:             end if
24:             if These pairs are linked then
25:                 output YES
26:             end if
27:         end for
28:     end if
29: end for
30: if all choices of $\Pi_1, \Pi_2$ have been examined then
31:     output NO.
32: end if
```

Theorem 4.2.2 we have a polynomial algorithm for semicomplete digraphs such combining these we have an algorithm for solving the k-linkage problem on a digraph $D \in \phi_1$ meaning we have $\mathcal{B}_{\phi_1}$.
For the last algorithm $\mathcal{C}_{\phi_1}$ it takes for every decomposition $D = S[M_1, M_2, \ldots, M_s]$ each M_i for $i = [s]$ and delete and add arcs so each M_i is a transitive tournament. $\square$

Now we have proved that ϕ_1 is a linkage ejector and in section 2.2 that a quasi-transitive digraph is totally ϕ_1-decomposable such for any quasi-transitive digraph we can use 1.

4.3 Solving Linkgae Problem in Locally Semicomplete Digraphs

A locally semicomplete digraph is either Round decomposable, Semicomplete or niether. We have in section section 2.3 called these evil locally semicomplete digraph or just evil. The semicomplete part is solved from **??** but the theorem will also be important in this section. First we will look at the evil semicomplete digraph where we need to recall Equation 2.3.8 (a) where we can see that a evil semicomplete digraph can be partitioned into into maximum 4 semicomplete digraphs S, D'_1, D'_2, D'_3 which leed us to the next theorem.

Theorem 4.3.1. *[3] For every fixed pair of positive integers c,k there exists a polynomial algorithm for the k-linkage problem on digraphs whose vertex set is partinionable into c sets inducing semicomplete digraphs.*

Let $c = 4$ in theorem Theorem 4.3.1 then we know from Equation 2.3.8 that every evil locally semicomplete digraph has a polynomial algoritm for the k-linkage problem when k is fixed.

The remaning class of digraphs inside the class of locally semicomplete digraphs is the round decomposable. Recall the class ϕ_2 = Semicomplete digraphs $\cup$ Round digraphs from section 2.1. As we did in section 4.2 we will in the end prove that ϕ_2 is a linkage ejector and since round decomposable digraphs is totally ϕ_2-decomposable we would have proven that there exists a polynomial algorithm for them. To prove that ϕ_2 is a linkage ejector we know from item 4.2.1 that it need 3 algorithms $\mathcal{A}_{\phi_2}, \mathcal{B}_{\phi_2}$ and $\mathcal{C}_{\phi_2}$. For the algorithm $\mathcal{B}_{\phi_2}$ we only need it for round digraphs.

Theorem 4.3.2. *For every fixed k, there exists a polynomial algorithm to solve the k-linkage problem on round digraphs.*

Proof. maybe blablabalbalablbalab $\square$

The last algortihm will be in the proof of the next theorem which will end the part about round decomposable digraphs.

Theorem 4.3.3. *For every fixed k, there exists a polynomial algorithm to solve the k-linkage problem on round decomposable digraphs.*

Proof. First we know from section 2.3 that round-decomposable digraphs are totally ϕ_2-decomposable. So if ϕ_2 is a linkage ejector then we can use 1 to find the k-linkage of a round decomposable digraph. This means all that are left to prove is that ϕ_2 is a linkage

ejector, for this we need to prove that ϕ_2 is closed with respect to blow-ups. The semicomplete digraphs we know from the proof of Theorem 4.2.5 that we can blow it up with a transitive tournament. A transitive tournament is also a round digraph, which means that we can blow up a vertex in a round digraph and it is still round.
The algorithm $\mathcal{A}_{\phi_2}$ is covered by Theorem 2.1.2. Now for algorithm $\mathcal{B}_{\phi_2}$ we have for the semicomplete digraphs a polynomial algorithm for the k-linkage problem by Theorem 4.2.2 and for round digraphs we have the algorithm Theorem 4.3.2 such combining these theorems we have $\mathcal{B}_{\phi_2}$. The last algorithm $\mathcal{C}_{\phi_2}$ takes for each M_i in a decomposition $R[M_1, M_2, \ldots, M_r]$ delete and add arcs so it becomes a transitive tournament. This makes the ϕ_2 a likage ejector. □

Now we have an algorithm for all locally semicomplete digraphs and therefore to end this section we have the following theorem.

Theorem 4.3.4. *For every fixed k, there exsist a polynomial algorithm to solve the k-linkage problem on locally semicomplete digraphs.*

Chapter 5

Arc-disjoint path in decomposable digraphs

5.1 The Weak-Linkage Problem

This problem is much like the problem we just went through exept instead of linking terminals with vertex disjoint path these path only need to be arc disjoint. Which ofcourse makes the problem apear more likely in digraphs but also harder to control since there is many more checks to go through.
Given two pair of vertices s_1, t_1 and s_2, t_2 finding arc-disjoint paths between each pair is a weak 2-linkage problem. Then the weak k-linkage problem is finding k arc-disjoint paths between k pair of terminals. where a terminal pair is a source and a zink in the paths of the solution of the linkage problem.
The weak linkage problem is also NP-complete and that is because the linkage problem is.

Theorem 5.1.1. *weak NP-complete*

sketz. blablabalablbalab □

The notation in this chapther is much like in the last, D as the digraph we are examine and Π as the set of pairs of terminals, where k is the number of pairs of terminals. Since we are focusing on arcs then F will be the main use of a set of arcs from the digraph D, ($F \subseteq A(D)$).
When talking about linkage problem for decomposable digraph, we can have houses with terminals in and some without any terminals. The houses with no terminals in is called **clean houses**. Then a terminal pair can either be inside the same houses or in different houses. As in linkage the definition of **internal pairs(and paths)** and **external pairs(and paths)**.
F is usely used for arcs that we do not want a part of the linkage we are focusing on mostly because the arcs are already used to link some other pairs, therefore when fousing on a vertex out- and in-degree compared to the set F it is usely bound by the number of pairs already linked. This is important since deleteing arcs could change the class the digraph belongs to. We will by an algorithm maybe end up in the same vertex as another pair that are linked and has to some how control that we do not choose the same arc as we did linking an other pair.

5.2 Solving Weak-Linkage in Quasi-transitive Digraphs

In this section we need to astablish some new properties for the class ϕ. Much like in section 4.2, the class need to have these proporties to be relevant for solving the weak k-linkage problem.
For a integer c, the class denoted $D(c)$ is a digraph D where first there is added as many parallel arcs to arcs that already exists in D then blow-up b vertices where $0 \leq b \leq c$, the digraph that is blown up has to have a size $\leq c$.

Definition 5.2.1. *[3] We say that a class of digraphs ϕ is Bombproof is there exsists a polynomial algorithm $\mathcal{A}_\phi$ to find a totally ϕ-decomposition of every totally ϕ-decomposable digraph and, for every integer c, there exists a polynomial algorithm $\mathcal{B}_\phi$ to decide the weak k-linkage problem for the class*

$$\phi(c) := \bigcup_{D \in \phi} D(c) \tag{5.1}$$

The clean houses (D, Π) actually have an important part, namely the part that we do not need them for linking any of the piars of Π in D.

Lemma 5.2.1. *[3] Let D be a digraph, Π a list of k terminal pairs and $H \subset D$ a clean house with respect to Π. Let D' be the contraction of H into a single vertex h. Then D has a waek Π-linkage if and only if D' has a weak Π-linkage.*

Proof. Maybe prove □

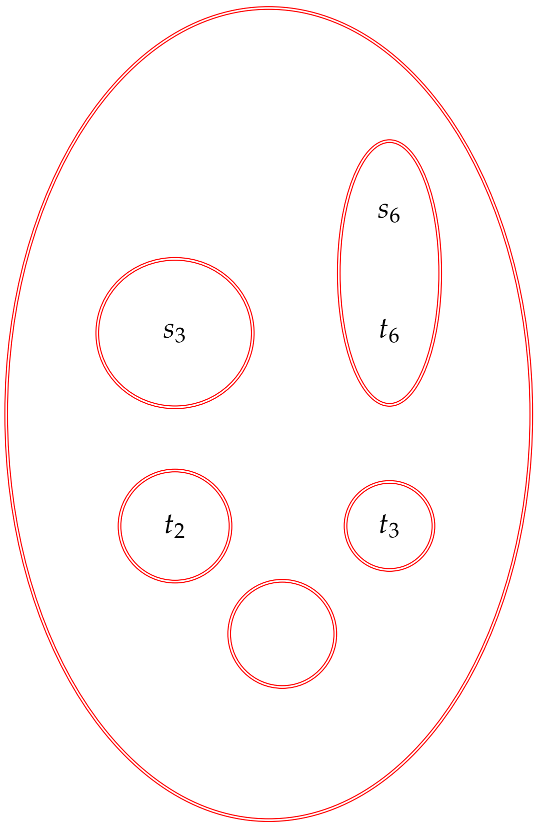
The external pairs do not need the same amount of vertices and arcs inside a house as maybe the internal pairs. It turns out that in [4] we bound the number of vertices for the external paths. The lemma below is a reformulation of the lemma from [4].

Lemma 5.2.2. *Let $D = S[H_1, \dots H_s]$ be a decomposable digraph, let Π' be a list of h terminal pair and let F be a set of arcs in D satisfying that $d_F^-(v), d_F^+(v) \leq r$ for all $v \in V(D)$. If $(D \backslash F, \Pi')$ has a weak linkage, then it has a weak linkage $P_1, \dots, P_h$ such that, we have $|V\left(\bigcup_{i \in \mathcal{E}} P_i \cap H_j\right)| \leq 2h(h+r)$, for every $j \in [1, \dots, s]$, where $\mathcal{E}$ denotes the set of indeces i for wich P_i is external. omformulere*

Proof. blablabalbalabal □

As shortly explain we sometimes have to control the set of arcs already used F but as we remove these arcs from the digraph it may longer belong to the class it did before. we therefore need to make sure the removing some arcs from a digraph do not affect that we have an algorithm for the weak linkage if we had one without removing the arcs.

Lemma 5.2.3. *Let $\mathcal{C}$ be a class of digraphs for which there exists an algorithm $\mathcal{A}$ to decide the weak k-linkage problem, whose running time is bounded by $f(n,k)$. Let $D = (V, A)$ be a digraph, Π a list of k pairs of terminals and $F \subseteq V \times V$ such that $D' := (V, A \cup F)$ is a member of $\mathcal{C}$. There exists an algorithm $\mathcal{A}^-$, whose running time is bounded by $f(n, k + |F|)$, to decide whether D has a weak Π-linkage.*



Proof. Let D be the digraph and $F = \{s'_1t'_1, \ldots, s'_{k'}t'_{k'}$ be the set of arcs missing in D so $D' = D(V, A \cup F)$ is in the class $\mathcal{C}$ let number of arcs in F be denoted by the non-negative number k' then we create a set of terminal based on every arc in F by the arcs tail and head as the pair of terminals in the set $\Pi' = \{(s'_1, t'_1), \ldots, (s'_{k'}, t'_{k'})\}$. We claim that D has a weak Π-linkage if and only if D' has a weak $\pi \cup \Pi'$-linkage which will also prove the theorem. First If D has a weak Π-linkage we just add the arcs from F as the Π'-linkage resulting in a $\Pi \cup \Pi'$-linkage in D'. For the other way we assume that D' has a weak $\pi \cup \Pi'$ deleting the arcs in F we would still have a weak Π-linkage, there are two possibilities either the linkage Π do not use any of the arcs in F and we can delete them without problems. the second possibility is that the weak Π-linkage use an arc of F. If this is the case then lets say its the arc $s'_it'_i$. Since (s'_i, t'_i) is a terminal pair of Π' these has to be linked through some other arcs since the arc $s'_it'_i$ is used already it can't be used, otherwise it is not a solution for the $(D', \Pi \cup \Pi')$ linkage problem. Meaning we substitute the path P'_i which link (s'_i, t'_i) with the arc $s'_it'_i$ in P_j which link s_j, t_j which is still a weak $\Pi \cup \Pi'$-linkage in D', we do this for all arcs that are used by the weak Π-linkage in D' then delete all arcs in F and we have the weak Π-linkage in D. □

Now we are going to state the theorem that is used for the existens of the of our main algorithm in this section. This result is found by ... in .... .

Theorem 5.2.4. *Let ϕ be a bombproof class of digraph. There is a polynomial algorithm $\mathcal{M}$ that takes as input a 5tuple $[D, k, k', \Pi, F]$ where D is a totally ϕ-decomposable digraph, k, k' are natural numbers with $k' \leq k$,Π is a list of k' terminal pairs and $F \subseteq A(D)$ is a set of arcs satiesfying*

$$d^-_F(v), d^+_F(v) \leq k - k' \text{ for alle } v \in V(D) \tag{5.2}$$

$$|F| \leq (k - k')2k$$

and decides wheter $D \backslash F$ contains a weak Π-linkage.

To proof Equation 5.2.4 we first state the algorithm $\mathcal{M}$ then we prove that it works, and last that the time for the algorithm is polynomial. Since the existens of the algorithm lyes in the proof that it works and it is polynomial we will first state it explain it come with an example of the choiches that it makes and then we will prove that is indeed the alforithm mensioned in Equation 5.2.4.

First we deskribe wiht words what the input and output of the algorithm is. The output is already written in words and is very undestandeble.
The input can be elaborated somewhat more, first $\mathcal{M}$ is polynomial but also recursively defined. It decides whether (D, Π) has a weak-linkage on overall k terminals. Since the algorithm is recursive it does not find all the solutions in one go therfore we define k' as the number of terminals that we still need to find a weak linkage for, and F is a part of the solution of at most $k - k'$ already found weak-linkages of D, Π is the set of terminals that we what to find the weak linkage for.
So you can in the begining have $F = \emptyset$ and $k = k'$. This will help on the undestaniding of the algorithm.

Step 1: "$\Pi = \emptyset$" makes sure that if we call the algorithm $\mathcal{M}$ with no pairs then there exists the solution with zero acrs to solve the weak-linkage problem.

Step 2: Recall that the digraph is totally ϕ-decomposable and ϕ is bomproof and from Equation 5.2.1 we know tha the digraph has a algortihm $\mathcal{A}_\phi$ that gives the ϕ-decomposition of the digraph.

Step 3-5: From Equation 5.2.1 we know that $\mathcal{B}_\phi$ decides a weak-linkage for $D \in \phi$, since we cant guarentee that $D \backslash F \in \phi$ we use Theorem 5.2.3 that tells us that $\mathcal{B}_\phi^-$ can decide a weak-linkage in $D \backslash F = (V, A \backslash F)$ if $D' \in phi\ D' = (V, (A \backslash F) \cup F) = (V, A) = D \in \phi$.

Step 6: Here we find all the non-clean houses from $H_1, .., H_s$ and contract all the clean houses w.r.t. Π we make a new nummeration of all the non-clean houses $K_1, \ldots, K_l$ of D w.r.t. Π Since by Theorem 5.2.1 we know that contracting one clean house in D if it has a linkage so does our new digraph, then use this lemma agian and agian until there is no more clean houses. This is our new digraph D' with non-clean houses $K_1, \ldots, K_l$ and if we find a weak linkage w.r.t. Π in D' we know that D has a weak linkage from continueing using Theorem 5.2.1. We also let $F' = F \cap A'$ where A' is the arcset of D'.

Step 7: Recall an internal pair is where both vertices is in the same house and an external pair is where the vertices is two different houses.

Step 8: This for loop is looking for two different cinds of path between internal pairs since the path for an internal pair can be an internal path (fully cept in the house) or an external path going out of and later in the house. For simplyfication look at Figure **??**

Step 9-11: if $\Pi^e \cup \Pi_1 = \emptyset$, either we have already found all external path in this partition or there was none. Either way all terminal pairs left is internal so and $\Pi = \Pi_2$. So we are only interesten in finding the internal path of the internal pairs, which is why we can call

Procedure 2 The main algorithm $\mathcal{M}$

Input: Digraph D, two natural numbers k and k' where $k' \leq k$, a list of k' terminal pairs Π, A set of arcs $F \subseteq A(D)$ satiesfying:

$$d^-_F(v), d^+_F(v) \leq k - k' \; \forall v \in V(D)$$
$$|F| \leq (k-k')2k$$

Output: Either "No weak-linkage exists" or "there exists a weak-linkage in (D, Π) with arc set F."

```
1:  if $\Pi = \emptyset$ then
2:      output that a solution exists and return
3:  end if
4:  Run $\mathcal{A}_\phi$ to find a total $\phi$-decomposition of $D = S[H_1, \dots, H_s]$.
5:  if this decomposition is trivial that is $D = S$ then
6:      $D \in \phi \subset \phi(1)$, so run $\mathcal{B}^-_\phi$ on $(D \backslash F, \Pi)$ to decide the problem and return.
7:  end if
8:  Find among $H_1, \dots, H_s$ those houses $K_1, \dots, K_l$ that contain at least one terminal. Let $D'$
    be obtaint by contracting all the clean houses. Let $F'$ be the set of arcs obtaint from $F$
    after the contraction.
9:  Let $\Pi^e \subset \Pi$ ($\Pi^i \subset \Pi$) be the list of external (internal) pairs $(s_q, t_q) \in \Pi$.
10: for every partion of $\Pi^i = \Pi_1 \cup \Pi_2$ look for external paths linking the pairs in $\Pi^e \cup \Pi_1$
    and internal pairs in $\Pi_2$ do
11:     if $\Pi^e \cup \Pi_1 = \emptyset$, then for $i = 1, \dots, l$: then
12:         run $\mathcal{M}$ recursively on input $[K_i, k, k'_i, \Pi \cap K_i, F \cap A(K_i)]$, where $\Pi \cap K_i$ denotes the
            list of terminal pairs that lie inside $K_i$ and $k'_i$ is the number of those pairs.
13:     end if
14:     if $\Pi^e \cup \Pi_1 \neq \emptyset$ then
15:         let $k'_i$ be the number of pairs in $\Pi_2 \cap K_i$
16:         for each possible choice of $l$ vertex sets $W_i \subseteq V(K_i), i = 1, \dots, l$ of size
            $\min\{|V(K_i)|, 2(k' - k'_i)(k - k')\}$ and arc sets $F_i \subseteq A(K_i\langle W_i \rangle) \backslash F$, $i = 1, \dots, l$ with
            $F_i$ satisfying
```

$$d^-_{F_i \cup (F \cap A(K_i))}(v), d^+_{F_i \cup (F \cap A(K_i))}(v) \leq k' - k'_i. \quad (5.3)$$
$$|F_i| \leq 2(k' - k'_i)(k - k') \quad (5.4)$$

```
            do
17:             for every $K_i$ do
18:                 remove all the vertices of $V(K_i) \backslash W_i$ and then all remaining arcs except those
                    in $F_i$.
19:             end for
20:             Define $D''$ to be the digraph obtaint from $D'$ with this procedure.
21:             Run $B^-_\phi$ on $(D'' \backslash F', \Pi^e \cup \Pi_1)$.
22:             for $i = 1, \dots, l$ do
23:                 run $\mathcal{M}$ recursively on input $[K_i, k, k'_i, \Pi_2 \cap K_i, F_i \cup (F \cap A(K_i))]$.
24:             end for
25:         end for
26:     end if
27:     if the if statement in 11 all intances examined are linked or at the if statement in 14
        there is a choice of $W_i, F_i, i = 1, \dots, l$ such that all instances examined are linked then
28:         output that a weak linkage exists and return.
29:     end if
30: end for
31: if all choices of $\Pi_1, \Pi_2$ have been consideredwithout verifying the existens of any weak
    linkage then
32:     output that no weak linkage exists.
```

$\mathcal{M}$ on each house for itself. Each K_i could be a big graph in itself that is decomposable with at least some house H_i where $|H_i| \geq 2$, if this is not the case the algorithm returns after step 3: and continue with the next. If $\mathcal{M}$ has already found some external paths F might not be empty and may use some arcs inside K_i therefore $F \cap A(K_i)$. $\Pi \cap K_i$ is becouse we are not interested in the terminal pairs that are not a part of the graph we are looking at (pairs inside K_j where $j \neq i$).

Step 12: Looking for external paths in a big graph is a bit more deficualt since we do not know which arcs and vertices not to use.

Step 13-14: First we find all the pairs that is internal pairs, we want to link as internal paths, the number of these is k_i' for each $i = 1, \dots l$. Then we choose a very specifik size of vertex sets W_i and loop over every choiche of these. These vertex set induces a subdigraph, where we make a possible arc set where inside not containg what is inside F we call these F_i we make these set as big as possible linkage need for the rest of the terminal pairs (those we want to find external paths of$\Pi^e \cup \Pi_1$) the number of those is $k' - k_i'$ since every pair mabey has to go through the house we are looking at.

Step 15-19: For each house we remove all vertices not in the vertex set W_i after removing these vertices we remove all remaining arcs except those arcs in F_i. This is defined in the algorithm as D''. We can show that $D'' \in \phi(2k^2)$. First we know that since D is totally decomposable $S \in \phi$ and from Equation 5.2.1 and the definition on $D(c)$ we can take S add as many parralelle arcs as we want we only need to blow up l vertices those houses of D that are not clean we know that there is k' terminal pairs and that $k' \leq k$ meaning $l \leq 2k$ these l vertices needs to be blown up and from lemma Theorem 5.2.2 lets say that we want to find $k'' \leq k'$ external paths in D ($|\Pi^e \cup \Pi_1| = k''$) then we are only looking at k'' terminals meaning in every blow up we need at most $2k''(k'' + (k - k'))$ since $k'' \leq k'$ we have $2k''(k'' + (k - k')) \leq 2k''k$ vertices in W_i and $2kk'' \leq 2kk' \leq 2k^2$ which is the biggest number we will need to blow up the l vertices meaning $c = 2k^2$ so $D'' \in \phi(2k^2)$.

Step 20-24: We need to make sure that the tuple $[K_i, k, k_i', \Pi_2 \cap K_i, F_i \cup (F \cap A(K_i))]$ upholds every condition for every choice of that tuple. Since we are not focusing on loops we know that the max number of arcs is bounded by the max number of vertices $|F_i| \leq 2kk''$ the rest of the terminals is the number of internal pairs which we in the algorithm denote k_i'. we know that $k_i' \leq k' - k''$ meaning $k'' \leq k' - k_i'$. we start calculating the to demands of F in the tuple. Note that $d_{(F \cap A(K_i))}(v) = d_F(v)$, $\forall v \in V(K_i)$ and we also know $d_{F_i}(v) \leq k''$ so

$$d^+_{F \cup F_i}, d^-_{F \cup F_i} \leq k - k' + k'' = k - (k' - k'') \leq k - k_i' \tag{5.5}$$

$$|(F \cap A(K_i)) \cup F_i| \leq |F| + |F_i| \leq 2k(k - k') + 2kk'' \tag{5.6}$$

$$\leq 2k(k - k') + 2k(k' - k_i') = 2k(k - k_i'). \tag{5.7}$$

Clearly the tuple for F holds forall its conditions.

Example of algorithm *mathcalM*

Proof. We have now proved and explained each step in the algorithm, that it does what we think. Now we need to check whether given your favorite digraph, that upholds the conditions, the algorithm gives the right result. If the digraph do not terminate before examine list $\Pi^e \cap \Pi_1$ of k'' terminal pairs if $k'' = 0$ we enter step 9 and $F_i = \emptyset$, for $i = 1, \dots l$, and by the induction hypothesis we can assume that if there exists a weak linkage in each K_i the algorithm would find it. Now if this is not the case and $k'' > 0$ step 12 is then entered and we construct D'' which we have described belong to $\phi(2k^2)$ and then as described before we can use B_ϕ^- which is correct by Equation 5.2.1, so the algorithm will find a weak Π''-linkage if it exists in $D''\backslash F'$. After all this there is made a recursiv call on each K_i finding k_i' weak linkages and by the above proof we know it works. So since B_ϕ^- correctly finds the weak linkage inside $D''\backslash F'$ using only arcs from F_i $\forall i \in [l]$, then each K_i is recursively called from D'' we can easily come back to D' since we find the weak k_i'-linkage inside each K_i which is not using any of the arcs from F_i we know that together these stil form the seperated weak linkages. By Theorem 5.2.3 we know that we can find a weak linkage in $D\backslash F$ if we can find it in $D''\backslash F'$ which just proved we can. given a perfekt weak Π-linkage. Polytime... □

5.2.1 k-linkage problem for quasi-transitive digraphs

We have already establish in section 2.2 that quasi-transitive digraphs are totally ϕ_1-decomposable. It turns out that we just have to prove that ϕ_1 is bombproof, for that we need the two polynomial algorithms $\mathcal{A}_{\phi_1}$ and $\mathcal{B}_{\phi_1}$. Recall that ϕ_1 is bulid up by semicomplete and acyclic digraphs so we need to establish some theorems for the weak k-linkage problem on semicomplete and acyclic digraphs.

Theorem 5.2.5. *The weak k-linkage problem is polynomial solvable for every fixed k when the input ia an acyclic digraph.*

Theorem 5.2.6. *The weak k-linkage problem polynomial for every fixed k, when we consider digraphs that are obtained from a semicomplete digraph by replacing some arcs with multiple copies of those arcs and adding any number of loops.*

Since the bombproof class alows the digraph to no longer be a part of that class we need to consider that an acyclic digraph can get a cycle when blowing up a vertex.

Theorem 5.2.7. *For every natural number p the weak k-linkage problem is polynomial for every fixed k, when we consider digraphs with most p directed cycles.*

Now we can prove that ϕ_1 is bombproof and therefore that quasi-transitive digraphs have a polynomial solution for the weak k-linkage problem, when k is fixed.

Theorem 5.2.8. *The class ϕ_1 is bombproof.*

Proof. For ϕ_1 to be bombproof it has to adhere the properties of Equation 5.2.1, the totally ϕ_1-decomposition can be found in polynomial time for any ϕ_1-decomposable digraph by Theorem 2.1.2. Now we only need the algorithm $\mathcal{B}_{\phi_1}$ for this we need to look at the constructin of $D(c)$ where $D \in \phi_1$. Let $D' \in D(c)$ Either D is semicomplete or D is acylic.
If D is semicomplete D'has at most c blown-up vertices $H_1, \ldots H_c$ of D to size at most c. If these H_i is independent we need for each H_i to be semicomplete not need more then c^2 arcs. So there is at most c^3 arcs missing from D' for it to be semicomplete, then by Theorem 5.2.3 we can find a weak k-linkage for D' if D is semicomplete. Now suppose D is acyclic blowing up c vertices with at a size at most c for this the rest of the proof □

5.3 Solving Weak-Linkgae in Locally Semicomplete Digraphs

Locally semicomplete digraph can be round-decomposable it turns out that we can from the independece number $\alpha(D)$ tell wether a digraph is round-decomposable or not. Recall independence number from section 1.1. The theorem below is from [4] where we omits some part of it since we have it statet elsewhere in the thises.

Theorem 5.3.1. *[4] A locally semicomplete digraph D havinng idependece number $\alpha(D)$ at least 3 is round decomposable with a unique round -decomposition.*

This means when considering all other locally semicomplete digraphs it has an independence number $\alpha(D) \leq 2$ which means for all not round-decomposable locally semicomplete digraphs we can use the algorithm in Theorem 5.3.2 to solve the weak k-linkage problem when k is fixed.

Theorem 5.3.2. *For every natural number α the weak k-linkage problem is polynomial for every fixed k, when we consider digraphs with independence number at most α.*

For solving the weak k-linkage problem in locally semicomplete digraphs we now only need to find a polynomial algorithm for the round-decomposable once. Before going into this we have to introduce something called the cutwitdh. This definition of cutwidth is inspired by the describtion of the cutwitdh in [4].
Given a digraph D and an ordering of the vertices $O = v_1, \dots, v_n$ we say that the ordering O has a **cutwidth** at most θ if $\forall j \in 2, 3, \dots n$ there are at most θ arcs u, v with $u \in \{v_1, \dots, v_{j-1}\}$ and $v \in \{v_j, \dots, v_n\}$ inset figur som viser cutwitdh θ for givet ordering. Say we have another ordering $O'$of the same digraph D, if O' has a cutwitdh at most θ for all possible orderings O' of D, then D is said to have a **cutwidth** at most θ.
The minimum natural number θ such that D has a cutwidth at most θ, we call θ **the cutwidth** of D. When we know the cutwidth of the digraph we can solve the weak k-linkage problem for those i polynomial time.

Theorem 5.3.3. *[4] For every natural number θ the weak k-linkage problem is polynomial for every fixed k, when we consider digraphs with cutwidth at most θ.*

introduction to D_Π(side 102 and 104 in [4]) + introduce Θ

Lemma 5.3.4. *Let B be a digraph of the form $B = R[H_1, \dots H_r]$, where R is round and has cutwidth at least Θ. Let Π be a list of piars of terminals. B has a Π-linkage if and only if B_Π, has a Π-linkage.*

Now we can use all this to prove that ϕ_2 which is defined in section 2.1 is bombproof and recall that round-decomposable digraphs is totally ϕ_2-decomposable.

Lemma 5.3.5. *The class ϕ_2 is bombproof*

Proof. blbalbalbalbalbal □

As mensioned above and proved in section 2.3 round-decomposable digraphs is totally *phi*$_2$-decomposable and we have just proved that ϕ_2 is bombproof so by the algortihm 2 for bombproof classes every round-decomposable digraph now have a polynomial algorithm to solve the weak k-linkage problem.

Theorem 5.3.6. *For every fixed k there exists a polynomial algorithm for the weak k-linkage problem for round-decomposable digraphs.*

This ends the part for round-decomposable digraph and in the begining of this section we proved that all other locally semicomplete digraphs than the round-decomposable once have a polynomial algorithm for the weak k-linkage problem. We can now state this.

Theorem 5.3.7. *For every fixed k there exists a polynomial algorithm for the weak k-linkage problem for locally semicomplete digraphs.*

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 Alessandro Maddaloni. Arc-disjoint paths in decomposable digraphs. *Journal of Graph Theory*, 77(2):89–110, 2014.

[5] 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.

[6] Tilde My Christiansen. *Algorithmic and structural problems in digraphs*. PhD thesis, 2018.