List Homomorphism to Irreflexive Oriented Trees

by

Ali PazokiToroudi

B.Sc.,Tehran University, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Approval

Name:	**Ali PazokiToroudi**
Degree:	**Master of Science (Computing Science)**
Title:	***List Homomorphism to Irreflexive Oriented Trees***
Examining Committee:	**Chair:** Ramesh Krishnamurti Professor

Pavol Hell
Senior Supervisor
Professor

Leonid Chindelevitch
Supervisor
Assistant Professor

Arash Rafiey
Supervisor
Adjunct Professor

Binay Bhattacharya
Internal Examiner
Professor
School of Computing Science

Date Defended: 17 October 2017

Abstract

Min ordering of a digraph H plays an important role in deciding the existence of a list homomorphism to H. For reflexive oriented trees T, there exists a concrete forbidden induced subgraph characterization to have a min ordering. For irreflexive oriented trees T, the existence of a min-ordering turned out to be somewhat harder, as there are many types of obstructions to its existence. In this thesis, we first review the existing results for list homomorphism problems $LHOM(H)$ for digraphs and graphs.
Second, for a specific subclass of irreflexive oriented trees, we present a concrete forbidden induced subgraph characterization to have a min ordering and to have an obstruction called invertible pair (*I-pair*) and digraph asteroidal triple (DAT). Moreover, for this subclass of irreflexive oriented trees T, we show that if T contains one of the forbidden obstructions, then the problem $LHOM(T)$ is NP-complete, and is polynomial otherwise.
Third, we discuss general trees, and present some approaches to find the minimal forbidden obstructions in the general case.

Keywords: min ordering; list homomorphism problem; directed asteroid triple (DAT); smooth trees ; irreflexive oriented trees

Dedication

To my little sister Fatemeh jan

Acknowledgements

I am deeply grateful to my supervisor, Professor Pavol Hell. Preparation and submission of this thesis would not have been possible without his patience, support and guidance. Pavol, it has been my honor to being your student. Almost always just one hint from you played a crucial role in solving a problem. I would also like to thank my supervisory committee members, Arash Rafiey and Leonid Chindelevitch. Arash, you were always available for discussion on problems. Thank you so much for your guidances and useful advices. Leonid, thank you for introducing me an interesting problem in computational biology, I really appreciate your support, advice and flexibility with me.
I would like to thank all of my dear friends, for never letting me feel alone and for filling me with joy and happiness with their presence and company, especially Akbar.
Last but not least, I would like to thank my parents whose love, kindness, understanding and advice has supported me, and make me the person I am today. I can not thank you enough for what you have done for me.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we will describe a special case of list constraint satisfaction problems called list homomorphism problems for digraphs H (or $LHOM(H)$). We will present the known facts about this problem and min ordering. Our focus is on Theorem 1.72 which describes the obstructions to both min ordering and polynomial algorithms for the problem $LHOM(H)$.

1.1 Preliminary Definitions

A ***graph*** $G = (V, E)$ consists a set of vertices $V(G)$ and a set of edges $E(G)$ which are unordered pairs of distinct elements of $V(G)$, loops are pairs when the elements are the same. A ***digraph*** $G = (V, E)$ consists a set of vertices $V(G)$ and a set of edges $E(G)$ which are ordered pairs of elements of $V(G)$.
Let $G = (V, E)$ be a digraph. The ***underlying graph*** $U(G) = (V_u, E_u)$ of G is defined as follows: $V_u = V$ and $E_u = \{(a, b)|a \neq b, (a, b) \in E \text{ or } (b, a) \in E\}$.
A ***walk*** $W = w_1, w_2, ..., w_n$ in digraph G is a sequence of consecutively adjacent vertices of G, let $W[w_i, w_j]$ denote the walk $w_i, ..., w_j$ for any $i \leq j$. A ***path*** $P = x_1, x_2, ..., x_n$ is a walk such that all x_i are distinct. Let $V(P)$ denote the set of vertices $\{x_1, x_2, ..., x_n\}$. A ***cycle*** $C = x_1, ..., x_n$ is a walk such that $x_1 = x_n$ and all x_i are distinct for $1 \leq i < n$. A walk $W = w_1, w_2, ..., w_n$ is called a ***closed walk*** if $w_1 = w_n$.
A ***tree*** T is a connected graph such that does not contain any cycle. An ***oriented tree*** T is a digraph whose underlying graph is a tree.

Suppose $W = w_1, w_2, .., w_n$ is a walk in the digraph G. We denote by $V(W)$ the set of vertices $\{w_1, w_2, ..., w_n\}$. If $W = w_1, w_2, .., w_n$ is a walk in G from $a = w_1$ to $b = w_n$, and $W' = w_1', w_2', .., w_n'$ is a walk in G from $b = w_1'$ to $c = w_n'$, we denote by $W + W'$ the walk from a to c which is the concatenation of W and W'. If $W = w_1, w_2, .., w_n$ is a walk in G from $a = w_1$ to $b = w_n$, we denote by $\overline{W}_1 = w_n, w_{n-1}, ..., w_1$ the walk W_1 traversed in the

opposite direction from b to a.
Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two digraphs. H is a ***subgraph*** of G when $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. H is an ***induced subgraph*** of G when $V(H) \subseteq V(G)$ and $E(H) = \{(u,v)|u,v \in V(H),(u,v) \in E(G)\}$. H is a ***proper induced subgraph*** of G when $V(H) \subset V(G)$ and $E(H) = \{(u,v)|u,v \in V(H),(u,v) \in E(G)\}$. Let H be a graph, and let x be a vertex in H. We denote by $H \setminus x$ the subgraph obtained from H by deleting x.

Definition 1.1. *The graph H is called :*

- *a* ***reflexive graph*** *when each vertex has a loop*
- *a* ***irreflexive graph*** *when no vertex has a loop*
- *a* ***general graph*** *when the vertices may or may not have a loop.*

Definition 1.2. *A cycle C in a graph G is called a* ***chordless cycle****, when its length is at least four and it does not have any chord (an edge not lying on the cycle C but connecting two vertices of the cycle C).*

Definition 1.3. *A graph H is called* ***chordal*** *when does not contain any chordless cycle. In other word, every induced cycle in H is a triangle.*

Definition 1.4. *A graph H is called an* ***interval graph*** *when H can be represented by a family of intervals on the real line I_v, $v \in V(H)$, such that $uv \in E(H)$ if and only if I_v and I_u intersect.*

According to the definition of an interval graph H, each interval I_v intersects itself so H is a reflexive graph. Note that this is not the standard definition of interval graphs, usually loops are not included [25]. However, for our purposes the loops are important.

Definition 1.5. *A digraph H is called an* ***interval digraph*** *when H can be represented by a family of pairs of intervals I_v, J_v, $v \in V(H)$, such that $uv \in E(H)$ if and only if I_u intersects J_v.*

Definition 1.6. *An graph H is called a* ***circular arc graph*** *when H can be represented by a family of arcs A_v, $v \in V(H)$, on a fixed circle such that $vu \in E(H)$ if and only if A_v and A_u intersect [10].*

Definition 1.7. *A* ***strongly connected component*** *of a digraph G is a maximal set of vertices $C \subseteq V(G)$ such that for each pair of vertices $u, v \in V(G)$, there exist a directed path from u to v and a directed path from v to u.*

Definition 1.8. *A* ***clique*** *is a subset of vertices of an graph H within which every two vertices are adjacent.*

Definition 1.9. *The **clique covering number** of an graph H is the minimum value k such that the set of vertices $V(H)$ can be partitioned to k cliques.*

Definition 1.10. *A bipartite graph H is **chordal bipartite** if it contains no chordless cycle of length at least 6.*

Definition 1.11. *Suppose G is a digraph. The **k-colouring** problem is defined as follows:*

- ***Given*** *: The digraph G.*
- ***Question*** *: Is there a colouring of the vertices of G such that every two adjacent vertices have different colors ?*

Definition 1.12. *Let H be a digraph, $uv \in E(H)$ is called **forward arc**, in this case vu is also called **backward arc**. If uv is both a forward arc and a backward arc, then it is called **double arc**.*

Definition 1.13. *Two walks $P = x_0, x_1, ..., x_n$ and $Q = y_0, y_1, ..., y_n$ in a digraph H are called **congruent walks** if x_ix_{i+1} is forward arc (backward arc) if and only if y_iy_{i+1} is forward arc (backward arc). In other words, both walks follow the same pattern of forward and backward arcs.*

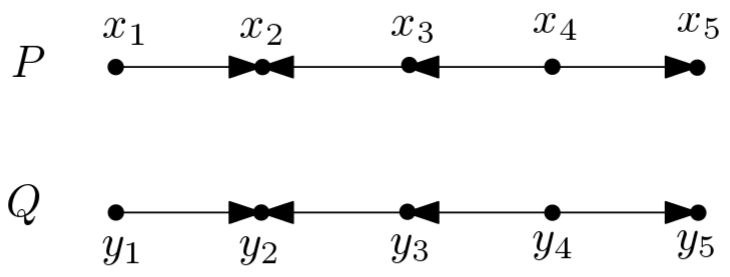


Figure 1.1: The walks P and Q are two congruent walks.

We will briefly discuss the mathematical analysis of algorithms. We require O-notation defined as follows.

Definition 1.14. *Suppose $f : Z^+ \to R$ and $g : Z^+ \to R$ are functions. The function $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that $0 \leq f(n) \leq c.g(n)$ for all $n \geq n_0$.*

When we design an algorithm for solving a problem, the most important question is how much time and space will it consume ? For predicting the time and space needed by an algorithm, we can use mathematical methods. The analysis of an algorithm will delineate how the running time and space consumption of the algorithm behave as a function of the size of the input. For example, we say that the ***running time*** of the algorithm A is a polynomial function p, if the algorithm A terminates in at most $O(p(n))$ steps for every input I with size n . The ***complexity*** of an algorithm is the growth rate of the algorithm's running time.

As a whole, we desire to design algorithms for solving the problems with ***polynomial complexity*** (having growth rate $O(n^c)$ such that c is a positive integer constant). Substantial amount of theory has been developed to figure out which class of problems can be tackled by algorithms with polynomial complexity. Before introducing different complexity classes we need to define decision problems. A problem is called a ***decision problem*** if it requires a yes or no answer. For instance, look at the following example: A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set.

The ***minimum vertex cover problem*** is the optimization problem of finding a smallest vertex cover in a given graph defined below:

The Minimum Vertex Cover Problem:
Given: A Graph G .
Output: Smallest number k such that G has a vertex cover of size k.

The corresponding decision problem is stated as follows:

The Vertex Cover Problem:
Given: Graph G and positive integer k .
Question: Does G have a vertex cover of size at most k ?

Now we are ready to introduce complexity classes. The class of all decision problems for which a polynomial-time algorithm exists is called the class P (polynomial). The class NP (non-deterministic polynomial) refers to the class of all decision problems such that there exists a polynomial-time algorithm which is able to verify the answer of any problem instance. It is not difficult to see that $P \subseteq NP$; on the other hand, it seems likely that $NP \not\subseteq P$, but this has not been proven yet[9]. In point of fact, the vertex cover problem is more likely to be placed in the class $NP \setminus P$. There is additional evidence to support this conjecture [19]. Moreover, no polynomial-time algorithm has been found for solving this problem. Vertex cover problem turns out to be one of the so-called NP-complete problems.

The idea of NP-completeness is based on a method which is called polynomial-time reduction defined as follows.

Definition 1.15. *Let P_1 and P_2 be two decision problems. A* ***polynomial-time reduction*** *from P_1 to P_2 is a polynomial-time algorithm R which constructs an instance $R(I)$ of the problem P_2 from any given instance I of the problem P_1 such that I is a yes-instance of P_1 if and only if $R(I)$ is a yes-instance of P_2.*

The notation $P_1 <_p P_2$ denotes that there is a polynomial time reduction from P_1 to P_2.

Definition 1.16. *A decision problem D is called to be* ***NP-complete*** *if it satisfies the following conditions.*

- $D \in NP$.
- *For every problem $D' \in NP$, there is a polynomial time reduction $D' <_p D$.*

We should notice that while the concept of NP-completeness has some nice properties, the existence of such problems is not actually obvious . To prove that a problem is NP-complete, one of the interesting questions is how it can encode any problem in NP. Cook [3] and Levin [22] independently showed how to do this for every natural problem in NP. They showed that the ***Satisfiability Problem*** (SAT) is NP-complete as the first NP-complete problem.

To introduce constraint satisfaction problem in the next section, we now define the concepts of general relational systems and homomorphisms.

Definition 1.17. *A* ***general relational system*** *$S = (V, R)$ is defined as a finite set $V = V(S)$ (the vertices of S) and a finite set of relations $R = R(S)$ (the relations of S) as follows:*

- $V(S) = \{v_1, v_2, .., v_n\}$.
- *$R = \{R_i(S)|i \in I\}$ such that $R_i(S)$ is a k_i-ary relation on $V(S)$ and I is a finite set.*

The set I and the integers $k_i, i \in I$, form the pattern (or type) of the general relational system S. Binary relational systems are general relational systems when all $k_i = 2$.

Definition 1.18. *Let T and S be two general relational systems with the same pattern. We define a* ***homomorphism*** *$f : S \to T$ to be a mapping $f : V(S) \to V(T)$ such that for every $i \in I$ we have:*

if $(w_1, w_2, ..., w_{k_i}) \in R_i(S)$, then $(f(w_1), f(w_2), ..., f(w_{k_i})) \in R_i(T)$.

Example 1.19. *Consider the following two general relational systems $S = (V, R)$, $T = (V, T)$ of the same pattern ($I = \{1, 2\}$, $k_1 = 2$ and $k_2 = 3$), and the mapping f from $V(S)$ to $V(T)$ which are defined as follows:*

- *The general relational system $S = (V, R)$:*
  - $V(S) = \{a, b, c, d, e\}$,
  - $R_1(S) = \{(a, b), (c, d)\}$, $R_2(S) = \{(a, b, c), (c, d, e)\}$
- *The general relational system $T = (V, R)$:*
  - $V(T) = \{0, 1, 2, 3\}$

  - $R_1(T) = \{(0,1),(1,2)\}$, $R_2(T) = \{(0,1,1),(1,2,3)\}$

- *The mapping* $f : V(S) \to V(T)$
  - $f(a) = 0, f(b) = f(c) = 1, f(d) = 2, f(e) = 3$

It is not hard to see that the mapping f *is a homomorphism between* S *and* T*. Take the pairs* $(a,b) \in R_1(S)$ *and* $(c,d) \in R_1(S)$*, here* $(f(a), f(b)) = (0,1)$ *and* $(f(c), f(d)) = (1,2)$ *are the members of* $R_1(T)$*. Moreover, take the triples* $(a,b,c) \in R_2(S)$ *and* $(c,d,e) \in R_2(S)$*, here also* $(f(a), f(b), f(c)) = (0,1,1)$ *and* $(f(c), f(d), f(e)) = (1,2,3)$ *are the members of* $R_2(T)$*.*

Definition 1.20. *Let* T *and* S *be two general relational systems. Suppose* L *is a family of lists,* $L(v) \subseteq V(T)$ *for every* $v \in V(S)$*. We define a* ***list homomorphism*** $f : S \to T$ *with respect to the lists* L *to be a mapping* $f : V(S) \to V(T)$ *such that:*

- *If* $(w_1, w_2, ..., w_{k_i}) \in R_i(S)$*, then* $(f(w_1), f(w_2), ..., f(w_{k_i})) \in R_i(T)$*.*
- *If* $v \in V(S)$*, then* $f(v) \in L(v) \subseteq V(T)$*.*

1.2 Constraint Satisfaction Problems

Constraint satisfaction problem (or $CSPs$) originated in the the field of artificial intelligence in the 1970s. Many natural problems arising in artificial intelligence and applied computer science can be modeled by the framework of constraint satisfaction problems. For instance, the eight queens puzzle, the Sudoku solving problem and many logic puzzles, the Boolean satisfiability problem, scheduling and assignment problems and different problems in graph theory such as the graph coloring problem can be expressed as constraint satisfaction problems. This problem has been formulated in different ways, here we will define constraint satisfaction problems by using general relational systems.

Definition 1.21. *Suppose* T *is a fixed general relational system with the pattern* P*. The* ***constraint satisfaction problem with template*** T *(or* $CSP(T)$*) asks whether or not an input general relational system* S *with the same pattern* P *admits a homomorphism to* T*.*

- ***Given:*** *A general relational system* S *with the same pattern* P*.*
- ***Question:*** *Is there a homomorphism* $f : S \to T$ *?*

As mentioned earlier in this section, a variety of problems can be formulated by CSP. For instance, here we will illustrate how the graph-coloring problem can be expressed as a constraint satisfaction problem. We construct two general relational systems with a same pattern ($I = \{1\}$, $k_1 = 2$) as follows: in the system T the vertices $V(T)$ are the set of colors $\{1, 2, .., k\}$ and there is just one binary relation $R_1(T) = \{1, 2, 3..., k\}^2 \setminus \{(i, i)|i = 1, ..., k\}$ that k is the number of colors. In the system S, the vertices $V(S)$ are the nodes of the graph $V(G)$ and there is also only one binary relation $R_1(S) = \{(u, v), (v, u)|uv \in E(G)\}$. Assume that there exists a homomorphism f from S to T. Let (a, b) be an arbitrary pair in $R_1(S)$, which is also an arbitrary edge in graph G, here $(f(a), f(b))$ must be in $R_1(T)$ hence we have $(f(a), f(b)) = (i, j)$, $i, j \in \{1, 2, ..., k\}$ and $i \neq j$. It means that no two adjacent vertices a and b in graph G receive the same color, and we can conclude that G is k-colorable. Therefore, we can say that the solution for k-coloring problem of graph G is equivalent to finding a homomorphism of S to T.

We know that NP-complete problems and polynomially solvable problems are subsets of the complexity class NP. Moreover, in 1975 Ladner [20] showed that if $P \neq NP$, then there exist problems in NP which are neither in P nor NP-complete. The class of such decision problems is called NP-intermediate. One of the important open problems in theoretical computer science is the following question asked by T. Feder and M. Vardi known as the Dichotomy Conjecture:

Conjecture 1.22. *[8] Every problem $CSP(T)$ is either in P or NP-complete.*

Recently, three independent papers have claimed a proof of the Dichotomy Conjecture[24, 26, 2].

Definition 1.23. *Suppose T is a fixed general relational system with the pattern P. The* ***list constraint satisfaction problem with template*** *T(or list $CSP(T)$) defined as follows:*

- ***Given**: The general relational system S with the same pattern P and lists $L(v) \subseteq V(T)$ for $v \in V(S)$.*
- ***Question**: Is there a list homomorphism $f : S \to T$?*

One interesting question that comes to mind is, "Is every problem in list $CSP(T)$ either in P or NP-complete". The following dichotomy which is proved by Bulatov answers this question.

Theorem 1.24. *[1] For any general relational system T, the problem list $CSP(T)$ is NP-complete or polynomial time solvable.*

Assume a constraint satisfaction problem with template T such that T has only one relation which is binary (i.e., T is a digraph). In this case, the $CSP(T)$ is exactly a homomorphism problem to digraph H (consider T in place of H).

Definition 1.25. *Let H be a fixed digraph, the* ***homomorphism problem to digraph*** *H (or $HOM(H)$) defined as follows:*

- ***Given:*** *A digraph G.*
- ***Question:*** *Is there a mapping $f : V(G) \to V(H)$ such that $uv \in E(G)$ implies $f(u)f(v) \in E(H)$?*

Hell and Nesetril completely classified the complexity of $HOM(H)$ when H is a graph which is a special digraph that the relation $E(H)$ is symmetric in the following theorem. Note that $HOM(H)$ is trivial when H has a loop because every graph G admits a homomorphism to H by mapping all vertices of G to the vertex v in H which has a loop.

Theorem 1.26. *[12] Let H be an irreflexive graph H. If H is a bipartite graph then $HOM(H)$ is polynomial time solvable, otherwise it is NP-complete.*

Definition 1.27. *Suppose $H = (V, E)$ is a fixed digraph, the* ***list homomorphism problem to*** *H (or $LHOM(H)$) asked the following question:*

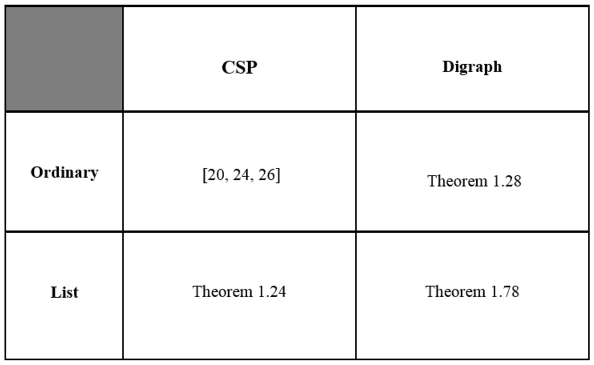
- ***Given**: A digraph $G = (V, E)$ and a family of lists $L(v) \subseteq V(H)$ for $v \in V(G)$.*
- ***Question**: Is there a mapping $f : V(G) \to V(H)$ such that $uv \in E(G)$ implies $f(u)f(v) \in E(H)$ and $f(v) \in L(v)$ for every $v \in V(G)$?*

Moreover, the list homomorphism problem to digraph H is a special case of the list $CSP(T)$ when T has only one relation which is binary. Based on the following theorem proved by T. Feder and M. Vardi, in order to find the solution for the Dichotomy Conjecture we just need to focus on this specific case, when the template T has only one relation which is binary.

Theorem 1.28. *[8] If dichotomy holds for problems $HOM(H)$ for digraphs H , then dichotomy holds for problems $CSP(T)$ for general templates T.*

Theorem 1.28 points out that dichotomy for problems $HOM(H)$ for digraphs is as hard as for all problems $CSP(T)$'s. On the other hand, the theorem below identifies that for the special case of digraphs (symmetric digraphs) dichotomy is known.

Theorem 1.29. *[13] Let H be a reflexive graph. If H admits a min ordering then $LHOM(H)$ is polynomial time solvable, otherwise it is NP-complete.*



	CSP	Digraph
Ordinary	[20, 24, 26]	Theorem 1.28
List	Theorem 1.24	Theorem 1.78

Figure 1.2: The references of dichotomies for CSP problems and homomorphism problems to digraphs.

1.3 Min Ordering

In this section, we shall introduce an ordering over the vertices of digraphs which is called min ordering. We will show that if digraph H admits a min ordering then the problem $LHOM(H)$ can be solved in polynomial time.

Definition 1.30. *Let H be a digraph and k a positive integer. A **polymorphism of H, of arity k** is a mapping $f : V(H)^k \to V(H)$ satisfying the following condition:*

$$\text{if } u^1v^1, u^2v^2, .., u^kv^k \in E(H) \text{ then } f(u^1, u^2, .., u^k)f(v^1, v^2, .., v^k) \in E(H)$$

Definition 1.31. *A polymorphism f is :*

- ***conservative** if always $f(u_1, u_2, ..., u_k) \in \{u_1, u_2, ..., u_k\}$.*
- ***idempotent** if $f(u, u, ..., u) = u$ for all u.*

Definition 1.32. *A polymorphism f of arity two is :*

- ***commutative** if $f(u, v) = f(v, u)$ for all u and v.*
- ***associative** if $f(u, f(v, w)) = f(f(u, v), w)$ for all u, v and w.*

Definition 1.33. *A polymorphism is called a **semi-lattice** when it is idempotent, commutative and associative*

Definition 1.34. *A polymorphism f of arity three is a **majority function** if $f(a, a, b) = f(a, b, a) = f(b, a, a) = a$ for any a and b.*

Definition 1.35. *A polymorphism f of arity three is called **maltsev** if $f(a, a, b) = f(b, a, a) = b$ for any a and b.*

Theorem 1.36. *[1] Let T be a general relation system. If for each pair of vertices $a, b \in V(T)$, there exists a conservative polymorphism f of T that either is*

- *binary and the restriction $f|_{a,b}$ is a semi-lattice*

  or

- *is ternary and the restriction $f|_{a,b}$ is majority or maltsev.*

then list $CSP(T)$ is polynomial time solvable. Otherwise it is NP-complete.

Theorem 1.36 which is proved by Bulatov provides a criterion for drawing a line between tractable and untractable case of list $CSP(T)$. Hell and Rafiey provided a simpler classification when the template T is a digraph as follows:

Theorem 1.37. *[15] Let H be a digraph. If for each pair of vertices $a, b \in V(H)$, there exists a conservative polymorphism f of T that either is*

- *binary and the restriction $f|_{a,b}$ is a semi-lattice*

  or

- *is ternary and the restriction $f|_{a,b}$ is majority,*

then $LHOM(H)$ is polynomial time solvable. Otherwise it is NP-complete.

Definition 1.38. *Let H be a digraph, a linear ordering $<$ of the vertices of H is called a **min ordering** if it satisfies the following condition: if $uv \in E(H)$ and $u^{'}v^{'} \in E(H)$, then $min(u, u^{'})min(v, v^{'}) \in E(H)$.*

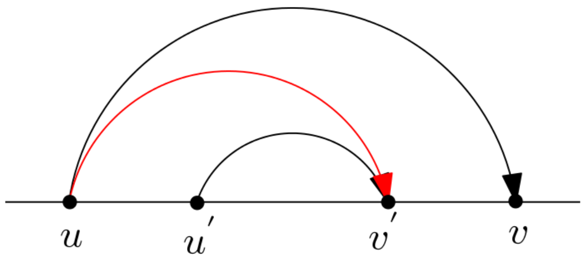


Figure 1.3: Min ordering.

Proposition 1.39. *Let H be a digraph. There exists a conservative semi-lattice polymorphism f on H if and only if H admits a min ordering.*

Proof. First we prove necessity. Suppose that H admits a min ordering $<$. Therefore, if $uv \in E(H)$ and $u^{'}v^{'} \in E(H)$, then $min(u, u^{'})min(v, v^{'}) \in E(H)$. We define $f(u, v) = min(u, v)$ as a mapping from $V(H)^2 \to V(H)$. By the definition of the ordering $<$, it is clear that f is a polymorphism, and it is conservative semi-lattice.
Second we prove sufficiency. Let f be a conservative semi-lattice polymorphism of H. We define an ordering $<$ as follows: $u < v$ whenever $f(u, v) = u$. Now let ab and cd be two arcs

in H, then we have $min(a,c) = f(a,c)$ and $min(b,d) = f(b,d)$. Since f is a conservative semi-lattice polymorphism, we have $f(a,c)f(b,d) \in E(H)$. Therefore, $min(a,c)min(b,d)$ is an arc in H. □

Based on the dichotomy classification for list $CSP(T)$ (see Theorem 1.36), conservative semi-lattice polymorphisms (i.e. min orderings) plays an important role in solving the problem $LHOM(H)$.

Theorem 1.40. *[23] If digraph H admits a min ordering, then $LHOM(H)$ is polynomial time solvable.*

Proof. Here we want to describe a polynomial time algorithm which solves $LHOM(H)$ when H admits a min ordering π. Suppose the digraph G with lists $L(v)$ for every $v \in V(H)$ is given. H is a fixed graph which has a min ordering. We can verify the existence of a list homomorphism from G to H and find it in the following steps:
At each iteration of the algorithm, we take an edge $uv \in E(G)$, and we verify the corresponding lists $l(u)$ and $l(v)$. The edges which are adjacent to u must be verified again when a vertex is deleted from $L(u)$. (See the steps 1 and 2)

- **Step 1:** Choose an unverified edge uv in $E(G)$ (At first all edges are unverified).
  - Find the vertex $x \in L(u)$ which has the minimum value of π among all vertices adjacent to some vertex in $L(v)$.
  - Find the vertex $y \in L(v)$ which has the minimum value of π such that $xy \in E(H)$.
  - If the vertices x or y do not exist, then there is no homomorphism from G to H.
- **Step 2:** Updates the lists $L(v)$ and $L(u)$ as follows:
  - Delete the vertices of $L(u)$ which are before x in π.
  - Delete the vertices of $L(v)$ which are before y in π.
  - If $L(v)$ or $L(u)$ became empty, then there is no homomorphism from G to H.

We have the updated lists $L(v)$ which are not empty for every $v \in V(H)$. Also for every $uv \in E(G)$ there is at least one edge $xy \in E(H)$ such that $x \in L(u)$ and $y \in L(v)$. There is a homomorphism from G to H which is defined as follows: map each vertex $v \in V(G)$ to the vertex $t \in L(v)$ which has minimum value of π. We want to show that the deleted vertices in step 2 can be eliminated for the list homomorphism. Suppose vertices $x^{'}, y^{'}$ such that $x^{'} > x$, $y^{'} < y$ and $x^{'}y^{'} \in E(H)$. Since y is the vertex with the minimum value of π, the vertex $y^{'}$ does not exist. It is easy to see that the min ordering yields the vertex p in $L(v)$ with the minimum value of π is adjacent to the vertex q in $L(u)$ with the minimum value of π.

In this algorithm, the number of the iterations is $O(|E(G)|)$ because each edge can be re-verify at most $O(|V(H)|)$ which is constant (H is fixed). Moreover, the vertex with the minimum value in π can be found in constant time when the min ordering is given. Therefore, the whole time complexity of this algorithm is $O(|E(G) + V(G)|)$. $\square$

Note that in general if H does not admit a min ordering, then the problem $LHOM(H)$ can be NP-complete or polynomial time solvable. In this thesis we identify some special trees T in which the absence of a min ordering does imply that $LHOM(H)$ is NP-complete.

1.4 Min-Max Ordering

Recall that for digraphs G and H, a homomorphism f of G to H is defined as a mapping from the vertices of G to the vertices of H such that it preserves the adjacency of the vertices. If furthermore every vertex $x \in V(G)$ is associated with costs $c_i(x)$, $i \in V(H)$, then the cost of a homomorphism f is $\sum_{x \in V(G)} c_{f(x)}(x)$.

Definition 1.41. *Let H be a fixed digraph. The* ***minimum cost homomorphism problem*** *for H (or MinHOM(H)) is defined as follows:*

- ***Given:*** *A digraph G, together with costs $c_i(x)$, $x \in V(G)$, $i \in V(H)$, and an integer k.*
- ***Question:*** *Is there a homomorphism from G to H of cost not exceeding the integer k ?*

We will define another kind of ordering called min-max ordering. It is similar to min ordering and corresponds to a particular type of lattice polymorphism.

Definition 1.42. *Let H be a digraph. a linear ordering $<$ of the vertices of H is called a* ***min-max ordering*** *if it satisfies the following condition: if $uv \in E(H)$ and $u'v' \in E(H)$, then $min(u,u')min(v,v') \in E(H)$ and $max(u,u')max(v,v') \in E(H)$.*

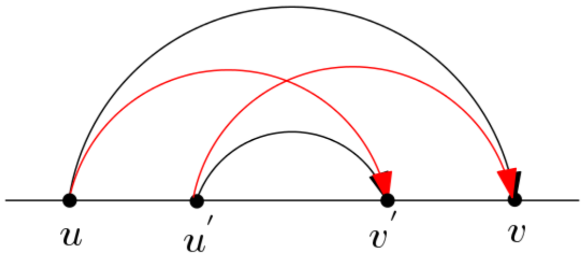


Figure 1.4: Min-max ordering.

In Theorem 1.40, we described that if a digraph H admits a min ordering, then the problem $LHOM(H)$ is polynomial time solvable. It turns out that the min-max ordering of a digraph H is crucial in deciding the existence of a min cost homomorphism to H.

Theorem 1.43. *[11] Let H be a digraph. If H admits a Min-Max ordering, then the problem $MinHOM(H)$ is polynomial time solvable.*

There exists a dichotomy for the problem $MinHOM(H)$ to digraphs. According to that dichotomy, tractable and untraceable cases of the problem $MinHOM(H)$ can be characterized by using an extended min-max ordering[16].

There exists an analogy between the above theorem and Theorem 1.40. By Theorem 1.40, digraphs H which admit a min ordering are easy cases for the problem $LHOM(H)$. On the other hand, by Theorem 1.43, digraphs H which admit a min-max ordering are easy cases for the problem $MinHOM(H)$.

In the following theorem, we will see how digraphs with a min-max ordering are characterized by forbidden structures.

Theorem 1.44. *[16] Let H be a digraph. The digraph H admits a min-max ordering if and only if satisfies the following conditions:*

- *(1) : the digraph H does not contain an induced unbalanced oriented cycle of net length greater than one, and*
- *(2) : the digraph H does not contain a pair of vertices (u, v) so that there exist two congruent walks $P = x_1, ..., x_n$ from u to v and $Q = y_1, ..., y_n$ from v to u such that there is no pair of arcs x_iy_{i+1} and y_ix_{i+1} for some i, $1 \leq i \leq n$.*

1.5 List Homomorphism Problems For Graphs and Digraphs

The list CSP dichotomy theorem, Theorem 1.36, provided by Bulatov was inspired by a more specific combinatorial classification for reflexive graphs, described below as Theorem 1.50. Thus a more combinatorial classification scheme was sought also for dichotomy of list homomorphisms for digraphs. Hell and Rafiey [15] distinguished between the tractable and intractable cases of the problems list $CSP(T)$'s in the case of digraphs ($LHOM(H)$) by using structural characterization. Forbidden structure characterizations approach not only is combinatorial and gives us an intuitive explanation as to what property of a graph or a digraph makes the problem hard but also it is easer to test. In this section, we will discuss list homomorphism problem for graphs and digraphs from structural characterization perspective.

Firstly, we shall focus on the problem $LHOM(H)$ for reflexive, irreflexive and general graphs. Secondly, we will move on to the problem $LHOM(H)$ for reflexive, irreflexive and general digraphs. In either case, we will discuss about the dichotomy classification which are given for that category.

1.5.1 List Homomorphisms to Graph H

We will discuss list homomorphism problem primarily in the context of reflexive graphs. The list homomorphism problem for reflexive graphs is able to model many real problems. For instance, suppose we have a group of researchers, and there are some connections between certain pairs of them (i.e. they have common research interest). On the other hand, suppose that we have a set of projects which will be assigned to the researchers, each project is suitable only for some of the researchers and certain pair of projects requiring collaboration. We can formulate this problem as follows: define the graph H with the researchers as vertices and their connections as edges. In addition, define the graph G with the projects as vertices and their collaborations as edges. Moreover, we also have for each project a list of admissible researchers. We can easily see that an assignment of projects to researchers is exactly a list homomorphism of G to H.

Definition 1.45. *An* ***asteroidal triple*** *of a graph H is a set of three vertices $\{a, b, c\} \subseteq V(H)$ which are not adjacent and for each pair $i \neq j \in \{a, b, c\}$, there is a path in H joining i and j not containing any neighbor of the third vertex $k \in \{a, b, c\}$ with $k \neq i, j$.*

Theorem 1.46. *[4] Let H be an reflexive graph. If H contains an asteroidal triple then $LHOM(H)$ is NP-complete.*

Theorem 1.46 is proved by reducing not-all-equal 3-satisfiability without negated variables problem to $LHOM(H)$.

Theorem 1.47. *[4] Let H be an reflexive graph, if H contains a chordless cycle then $LHOM(H)$ is NP-complete.*

On the other hand, the following theorem proved by Lekkerkerker and Boland characterizes the class of interval graphs by asteroidal triples and chordless cycles.

Theorem 1.48. *[21] A graph H is interval if and only if it contains no asteroidal triple and no chordless cycle.*

Theorem 1.49. *[4] If H is an interval graph then $LHOM(H)$ is polynomial time solvable.*

In the proof of Theorem 1.49, Feder and Hell devised a polynomial time reduction of this problem to the problem of 2-satisfiability which is solvable in polynomial time. According to the results of Theorems 1.49 and 1.48, we can now introduce the following theorem which is determining precisely the boundary between easy and hard cases of the problem $LHOM(H)$ based on the structure of H when it is reflexive graph .

Theorem 1.50. *[4] Let H be a reflexive graph, If H is an interval graph then $LHOM(H)$ is polynomial time solvable, Otherwise it is NP-complete.*

Theorem 1.51. *[13] A reflexive graph H is an interval graph if and only if it admits a min ordering.*

According to Theorem 1.51, we are able to recognize whether or not a reflexive graph is interval graph by checking whether it admits a min ordering or not. This property leads to the fact that easy and hard cases of $LHOM(H)$ for reflexive graphs also can be distinguished by min ordering property. Therefore, we can restate Theorem 1.50 based on min ordering in this way. If H admits a min ordering then $LHOM(H)$ is polynomial time solvable, otherwise it is NP-complete.

By Theorem 1.26, if H is an irreflexive graph which is not bipartite then $LHOM(H)$ is NP-complete. Therefore, in order to give a dichotomy classification for $LHOM(H)$ for irreflexive graphs we just need to draw a line between easy and hard bipartite graphs for the problem $LHOM(H)$. First, we will describe for which subclass of irreflexive bipartite graphs, the problem $LHOM(H)$ is polynomial time solvable.

Theorem 1.52. *[5] Let H be an irreflexive bipartite graph. If the complement of H is a circular arc graph, then $LHOM(H)$ is polynomial time solvable.*

Theorem 1.52 is proved by giving a polynomial time reduction of this problem to the problem 2-satisfiability problem

In the following we will describe certain structures whose presence in an irreflexive bipartite graph H implies that $LHOM(H)$ is NP-complete. One class of such structures is the chordless cycles of length greater than four and the second one is a special edge-asteroid defined below.

Theorem 1.53. *[5] If H contains a chordless cycle of length greater than four, then $LHOM(H)$ is NP-complete.*

Theorem 1.53 is proved by presenting a polynomial time reduction from k-colourability.

Definition 1.54. *A **special edge-asteroid** in a bipartite graph with bipartition(A, B) is a set of edges E and a set P of pathes defined below:*

- $E = \{u_0v_0, u_1v_1, ..., u_{2k}v_{2k} | k \geq 1, u_i \in A, v_i \in B\}$
- $P = \{P_{0,1}, P_{1,2}, ..., P_{2k,0}\}$ *such that*
  - *each $P_{i,i+1}$ joins u_i to u_{i+1}, and*
  - *there is no edge between $\{u_i, v_i\}$ and $\{v_{i+k}, v_{i+k+1}\} \cup V(P_{i+k,i+k+1})$ for every $0 \leq i \leq 2k$(subscripts are modulo $2k+1$), and*
- *there is no edge between $\{u_0, v_0\}$ and $\{v_1, v_2, .., v_{2k}\} \cup V(P_{1,2}) \cup V(P_{2,3}) \cup ... \cup V(P_{2k-1,2k})$.*

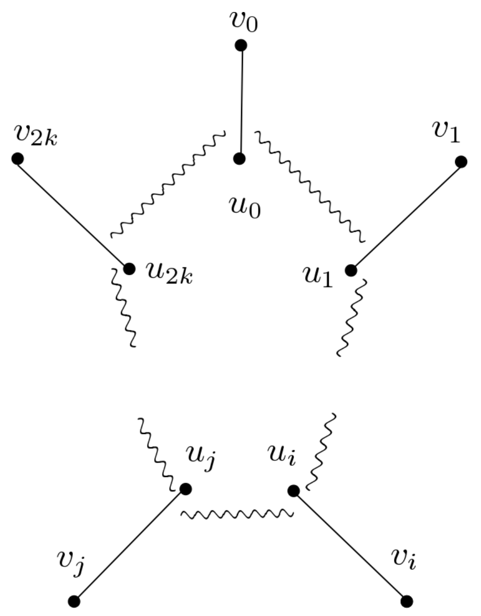


Figure 1.5: An edge-asteroid.

Theorem 1.55. *[5] Let H be an irreflexive bipartite graph. If H contains a special edge-asteroid, then $LHOM(H)$ is NP-complete.*

The above theorem is proved by reducing 3-colourability problem to $LHOM(H)$ when H contains a special edge-asteroidal.

So far we have seen which structures make the problem easy and which ones hard. Now we want to show that the structures whose presence yield $LHOM(H)$ hard are exactly the forbidden subgraphs for circular arc graphs of clique covering two.

Theorem 1.56. *[5] A graph H is the complement of a circular arc graph of clique covering two if and only if H is chordal bipartite and contains no edge-asteroidals.*

In conclusion, we have the following dichotomy classification for $LHOM(H)$ when H is irreflexive graph.

Theorem 1.57. *[5] Let H be an irreflexive graph. If the complement of H is a circular arc graph of clique covering number two, then $LHOM(H)$ is polynomial time solvable, otherwise it is NP-complete.*

Now we shall describe how we can extend the complexity classifications from the reflexive and irreflexive cases discussed above to general graphs.

In order to characterize the easy cases of $LHOM(H)$, We have used the interval representation of graph H for reflexive case and circular arc representation of graph H for irreflexive case. Similarly, we will utilize a new geometric representation called bi-arc for the general case.

Definition 1.58. *Assume that C is a circle with two fixed points p and q on C. A* ***bi-arc*** *is an ordered pair of arcs (N, S) on C such that N contains p but not q, and S contains q but not p.*

Definition 1.59. *A graph H is called* ***bi-arc graph*** *if it can be presented by a family of bi-arcs $\{(N_x, S_x) : x \in V(H)\}$ such that for every $x, y \in V(H)$, not necessarily distinct, the following hold:*

- *If $xy \in E(H)$, then neither N_x intersects S_x nor N_y intersects S_y.*
- *If $xy \notin E(H)$, then both N_x intersects S_x and N_y intersects S_y.*

The class of bi-arc graphs contains all reflexive interval graphs and all complements of irreflexive circular arc graphs of clique covering two. Feder, Hell and Huang [5, 6] showed that the class of reflexive bi-arc graphs is equivalent to the reflexive interval graphs and the class of irreflexive bi-arc graphs is equivalent to the irreflexive circular arc graphs of clique covering two. Thus when H is reflexive or irreflexive we can say that if H is a bi-arc graph then $LHOM(H)$ is polynomial time solvable, otherwise it is NP-complete.

Theorem 1.60. *[6] Let H be a general graph. If H is a bi-arc graph then $LHOM(H)$ is polynomial time solvable, otherwise it is NP-complete.*

In conclusion, based on the above theorem we can draw a line between easy and hard cases of $LHOM(H)$ for general graphs by figuring out whether or not the graph H admits a bi-arc representation.

1.5.2 List Homomorphisms to Digraphs H

The list homomorphism problem to digraphs H is more complex than graphs. Here the corresponding homomorphism must preserve both the adjacency of the vertices and the direction of the edges.

Definition 1.61. *Let $P = x_0, x_1, ..., x_n$ and $Q = y_0, y_1, ..., y_n$ be two congruent walks in a digraph H. We say P* ***avoids*** *Q, if there does not exist an arc $x_i y_{i+1}$ with the same pattern (forward or backward arc) as $y_i y_{i+1}$.*

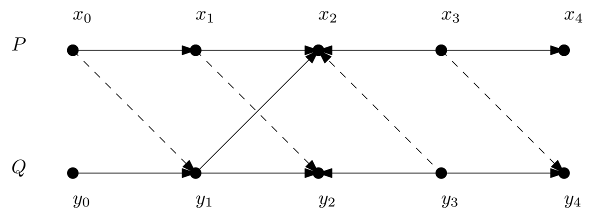


Figure 1.6: dashed edges represent missing edges.

In the figure 1.6, we have two congruent walks P and Q such that P avoids Q but Q does not avoid P.

Proposition 1.62. *Let $P = x_0, x_1, ..., x_n$ and $Q = y_0, y_1, ..., y_n$ be two congruent walks in a digraph H. If P avoids Q, then $\overline{Q}$ avoids $\overline{P}$. This property is called* ***skew symmetry****.*

Definition 1.63. *A pair of vertices (a, b) in a digraph H is called an* ***invertible pair*** *if it satisfies the following conditions:*

- *there exist congruent walks P from a to b and Q from b to a such that P avoids Q, and*
- *there exist congruent walks $P^{'}$ from b to a and $Q^{'}$ from a to b such that P' avoids $Q^{'}$.*

Note that it is possible that $P^{'}$and $Q^{'}$ are the inverses of P and Q respectively, as long as both P avoids Q and Q avoids P.

Reformulating of the definition of invertible pair in terms of an auxiliary digraph in the following way will be seen to be useful.

Definition 1.64. *Let H be a digraph, the pair digraph H^{+} defined as follows:*

- *The vertices of H^{+} are all ordered pairs (u, v) such that $u \in V(H)$ and $v \in V(H)$.*
- *There is an edge from (u, v) to $(u^{'}, v^{'})$ in H^{+} in any of the following situations :*
  - *$uu^{'} \in E(H)$, $vv^{'} \in E(H)$ and $uv^{'} \notin E(H)$ or*
  - *$u^{'}u \in E(H)$, $v^{'}v \in E(H)$ and $v^{'}u \notin E(H)$*

It is not hard to see that a directed walk in H^{+} from (u, v) to (v, u) results in two congruent walks P and Q in H from u to v and v to u respectively such that P avoids Q. On the other hand, such walks P and Q in H lead to a directed walk from (u, v) to (v, u) in H^{+} so we have the following lemma.

Lemma 1.65. *[7] If (u, v) is an invertible pair in H, then (u, v) and (v, u) belong to the same strong component C of the pair digraph H^{+}. In addition, for any (x, y) in C, the corresponding pair (y, x) also belongs to C, hence each pair (x, y) in C is an invertible pair in H.*
If there is not any invertible pair in H, then for each strong component C of H^{+} there exists a reversed strong component $C^{'}$ such that $(x, y) \in C$ if and only if $(y, x) \in C^{'}$.

We easily observe that an invertible pair is an obstruction to the existence of a min ordering .

Lemma 1.66. *[7] If a digraph H has an invertible pair, then H does not admit a min ordering.*

Proof. Let $<$ be a min ordering for H. Suppose now that (a, b) is an invertible pair in H. By Lemma 1.66, two pairs (a, b) and (b, a) belong to the same strong component C of the pair-digraph H^+. Therefore, there exists a directed path $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ from (a, b) to (b, a) in H^+. Now suppose $x_1 = a < y_1 = b$, then we must have $x_2 < y_2$. Otherwise, by the way H^+ is constructed there is no arc from x_1 to y_2 in H which contradicts that $<$ is a min ordering. By similar argument, x_i must appear before y_i in $<$ for every i, $2 \leq i \leq n$. Therefore, finally $x_n = b$ must appear before $y_n = a$ in $<$ which is a contradiction with the fact that $<$ is a linear ordering because first we assumed that $x_1 = a < y_1 = b$.

□

It is interesting to note that the converse of Lemma 1.66 holds when we are studying reflexive digraphs. Specifically that a reflexive digraph H has a min ordering if and only if H has no invertible pair [7].

There is a class of digraphs analogous to interval graphs for which the list homomorphism is also tractable, as was the case for interval graphs.

Definition 1.67. *An interval digraph H is called an* ***adjusted interval graph*** *when the intervals I_v and J_v have the same left end point.*

We have the following theorem that is relating the class of adjusted interval graphs to min ordering. In other words, we can recognize whether a digraph is adjusted interval graph or not by checking whether or not admits a min ordering.

Theorem 1.68. *[7] A reflexive digraph H is an adjusted interval graph if and only if it admits a min ordering.*

Based on Lemma 1.66, we can also restate the above theorem in terms of invertible pairs as follows. A reflexive digraph H is an adjusted interval graph if and only if it has no invertible pair. The following stronger result turns out.

Theorem 1.69. *[7] Suppose that H is a reflexive digraph. The following statements are equivalent.*

- *H is an adjusted interval digraph.*
- *H has a min ordering.*
- *H has no invertible pair.*
- *H the vertices of H^+ can be partitioned into sets D and D' such that*
  - *$(x, y) \in D$ if and only if $(y, x) \in D'$*
  - *$(x, y) \in D$ and $(x, y)(x', y')$ in $E(H^+)$ implies $(x', y') \in D$*
  - *(x, y) and $(y, z) \in D$ implies $(x, z) \in D$.*

In addition, as stated in Theorem 1.40, the list homomorphism problem $LHOM(H)$ for digraphs can be solved in polynomial-time if the graph H admits a min ordering. Hence we have the following theorem determining the easy cases of $LHOM(H)$.

Theorem 1.70. *[7] Let H be a reflexive digraph. If H is an adjusted interval digraph, then the problem $LHOM(H)$ is polynomial time solvable.*

In other words, the class of adjusted interval reflexive digraphs does not contain DAT. There is a following conjecture for reflexive digraph which is the converse of previous theorem.

Conjecture 1.71. *[7] Let H be a reflexive digraph. If H is not an adjusted interval digraph, then the problem $LHOM(H)$ is NP-complete.*

In particular, when H is a reflexive oriented tree then Conjecture 1.71 is true.

Theorem 1.72. *[7] Let H be a reflexive oriented tree. Then the following statements are equivalent.*

- *H is an adjusted interval tree*
- *H has no invertible pair.*
- *H does not contain any of the trees $T_1, .., T_7$, or their reverse as an induced subgraph.*

Corollary 1.73. *[7] Let H be a reflexive oriented tree. If H is adjusted interval digraph, then $LHOM(H)$ is polynomial time solvable. Otherwise it contains one of the trees $T_1, T_2, .., T_7$, or their reverses, as an induced subgraph and $LHOM(H)$ is NP-complete.*

In the proof of Corollary 1.73 we have two cases. If H is an adjusted interval graph, then H has a min ordering so $LHOM(H)$ is polynomial-time solvable. Otherwise, it is proven that when H does not admit a min ordering (having invertible pairs), then H contains one of the trees $T_1, .., T_7$ or their reverses as an induced subgraph.Moreover, there is always a DAT in the trees. Hence, $LHOM(H)$ is NP-complete in this case.

To pursue our discussion, let us take a look at list homomorphism problem for general digraphs. We first introduce the essential definition required to study list homomorphism problem for general digraphs.

Definition 1.74. *Let H be a digraph, a triple of vertices u, v, w is called a* ***permutable triple*** *when there exist six other corresponding vertices $s(u), b(u)$,$s(v), b(v)$ and $s(w), b(w)$ satisfying the following conditions:*
for every permutation x, y, z of $\{u, v, w\}$:

- *There exist a walk P from x to $s(x)$.*
- *There exist a walk Q from y to $b(x)$.*

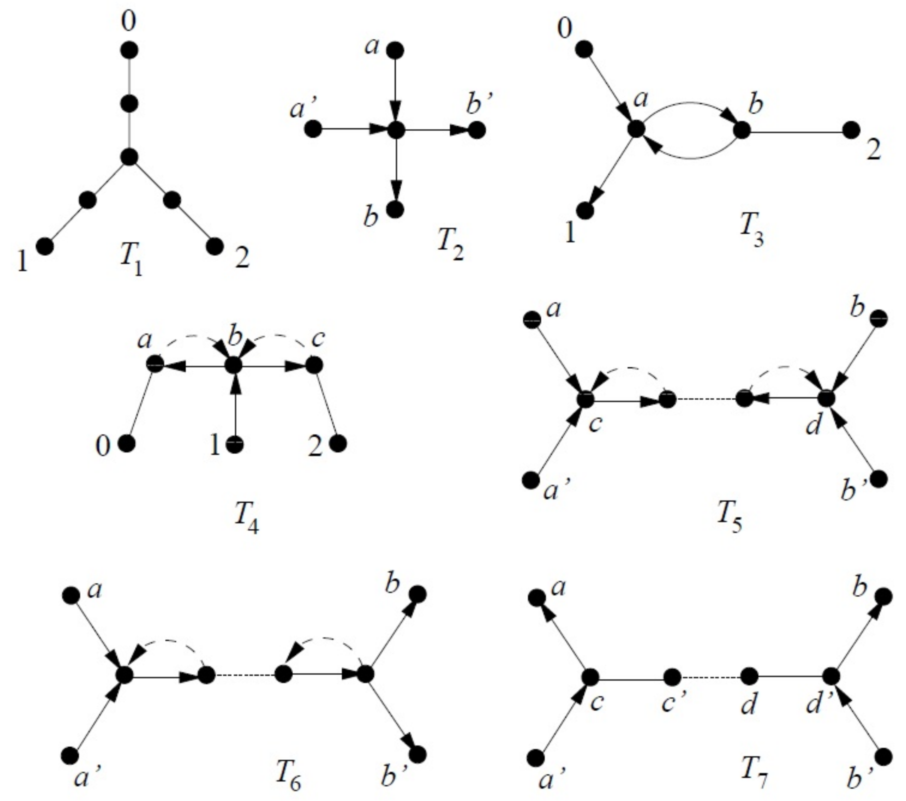


Figure 1.7: Minimal trees with invertible pairs. (Dashed arcs are optional; arcs without directions can be forward, backward, or double; dotted lines denote paths; loops are omitted.[7])

- *There exist a walk S from z to $b(x)$.*
- *The walks P, Q and S are congruent. (these three walks follow a same pattern).*
- *The walk P avoids both Q and S.*

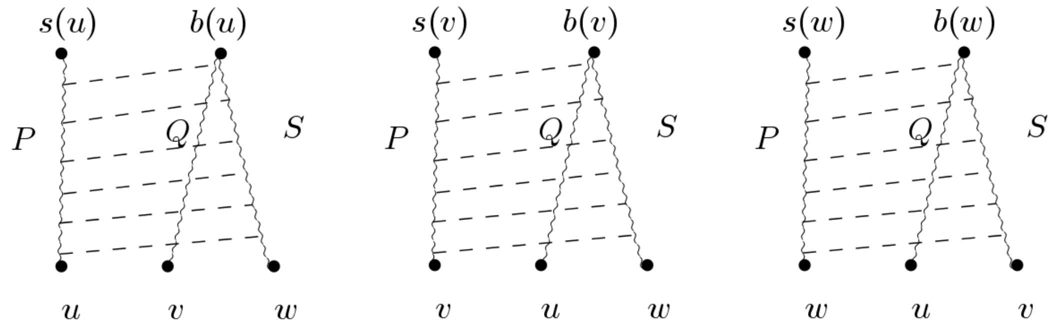


Figure 1.8: Dashed lines are missing arcs. The walk P avoids both walks Q and S.

Definition 1.75. *A permutable triple u, v, w is called a **digraph asteroidal triple** (or DAT) when the following pairs of the vertices $(s(u), b(u))$, $(s(v), b(v))$ and $(s(w), b(w))$ are invertible pairs.*

First, let us look at the hard cases of $LHOM(H)$ when H is a general digraph.

Theorem 1.76. *[15] If general digraph H contains a DAT, then $LHOM(H)$ is NP-complete.*

Theorem 1.77. *[15] Let H be a DAT-free digraph, then H admits a binary polymorphism f and a ternary polymorphism g such that*

- *if u, v is not invertible pair then $f|_{u,v}$ is semi-lattice, and*
- *if u, v is invertible pair then $g|_{u,v}$ is majority.*

According to the main Theorem 1.36 and the above theorem, we can conclude that if H is a DAT-free digraph, then $LHOM(H)$ is polynomial-time solvable. Now we are ready to states the dichotomy classification for the problem $LHOM(H)$ when H is a digraph.

Theorem 1.78. *[15] Let H be a digraph, if H contains a DAT then $LHOM(H)$ is NP-complete. If H is DAT-free then it is polynomial time solvable.*

Based on the dichotomy classification (see Theorem 1.78), we are able to distinguish between tractable and untractable cases of the problem $LHOM(H)$ for digraphs by checking that H contains DAT or not. Therefore, one of the interesting challenges is investigating which class of the digraphs are DAT-free or finding some forbidden structures for DAT-free digraphs.

Theorem 1.79. *[7] Let T be an reflexive oriented tree. Then T has an invertible pair if and only if T contains a DAT.*

The above theorem is what we want to prove for irreflexive oriented trees as well.
In conclusion, Figure 1.9 represents the summary of works have done on the problem $LHOM(H)$ which are discussed through this chapter. The membership in all those classes: bi-arc graphs, DAT-free digraphs (and hence all the others) can be tested in time polynomial in $|V(H)|$.

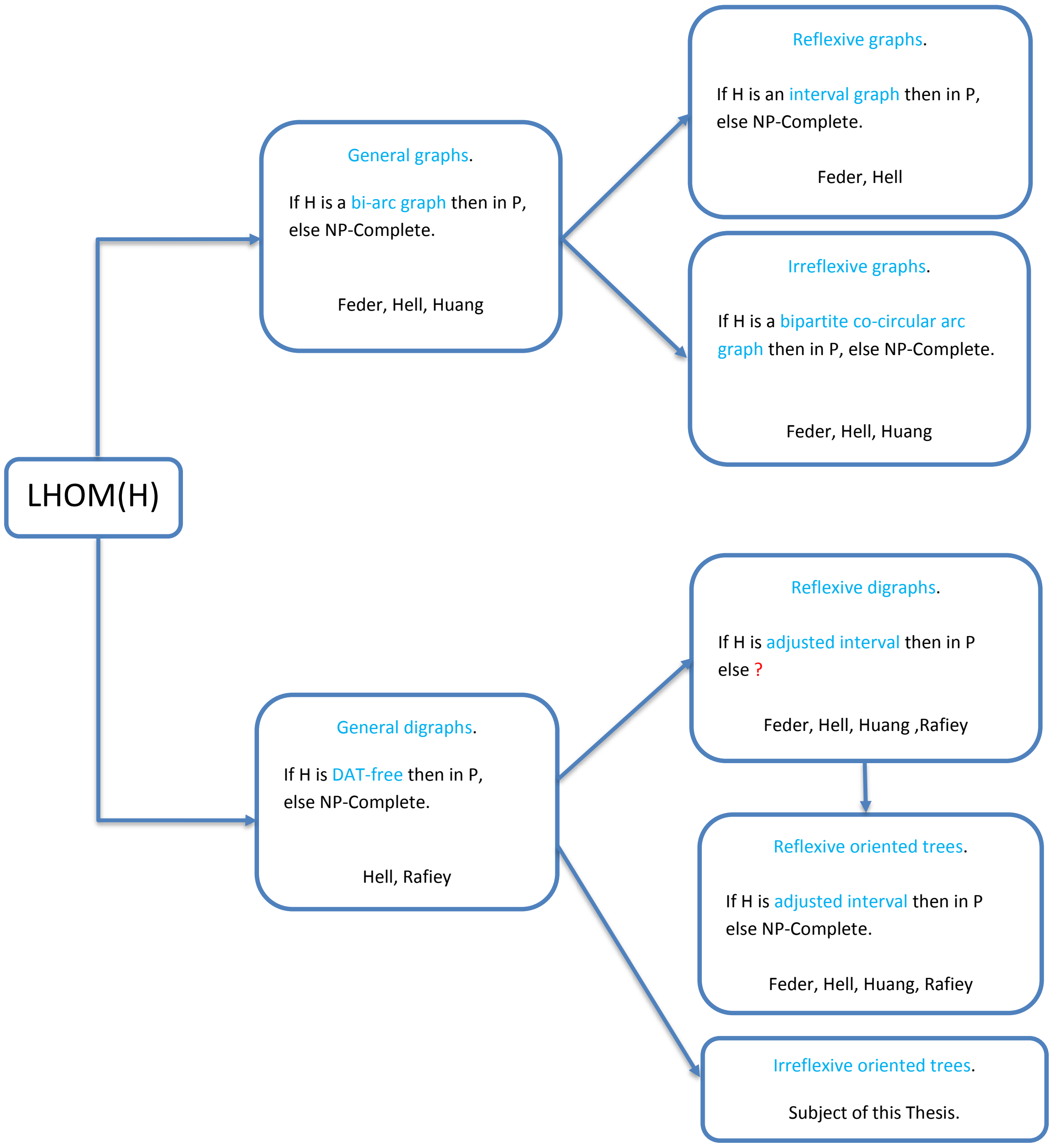

LHOM(H)
General graphs.
If H is a bi-arc graph then in P, else NP-Complete.
Feder, Hell, Huang
Reflexive graphs.
If H is an interval graph then in P, else NP-Complete.
Feder, Hell
Irreflexive graphs.
If H is a bipartite co-circular arc graph then in P, else NP-Complete.
Feder, Hell, Huang
General digraphs.
If H is DAT-free then in P, else NP-Complete.
Hell, Rafiey
Reflexive digraphs.
If H is adjusted interval then in P else ?
Feder, Hell, Huang ,Rafiey
Reflexive oriented trees.
If H is adjusted interval then in P else NP-Complete.
Feder, Hell, Huang, Rafiey
Irreflexive oriented trees.
Subject of this Thesis.

Figure 1.9: The problem $LHOM(H)$ for graphs and digraphs.

Chapter 2

Smooth Trees of Height Less Than Three

According to Theorem 1.72, for reflexive trees T, there exists a concrete forbidden induced subgraph characterization to have a min ordering. On the other hand, by getting an irreflexive analogue of Theorem 1.72 has turned out to be hard. For irreflexive trees T, there exists many types of obstructions to existence of a min ordering, but we were at least fully successful in a special case studied in this chapter, namely Theorem 2.15.
For this special case, we will show a concrete forbidden induced subgraph characterization to have a min ordering and a DAT. Moreover, we will show that we can draw a line between easy and hard cases of the problem $LHOM(T)$ for this special case of trees T by checking whether or not the tree T contains one of the obstructions as an induced subtree, namely Corollary 2.16.
Moreover, we introduce a new type of drawing a tree on levels (parallel lines), which we call it a layout. The layout yields an easy recognition of trees which admit a min ordering. In fact, we show that a tree has a layout if and only if it admits a min ordering, namely Proposition 2.16. The concept of layout is turned out to be a useful tool for characterizing the minimal obstructions for general trees.

2.1 Definitions

We first present the essential definitions needed to study list homomorphism problems for irreflexive oriented trees. In this chapter we shall abbreviate "irreflexive oriented tree" to "tree".

Definition 2.1. *A tree T is called **smooth** when $d^+(v) = 0$ or $d^-(v) = 0$ or $d^+(v) = d^-(v) = 1$ for every $v \in V(T)$.*

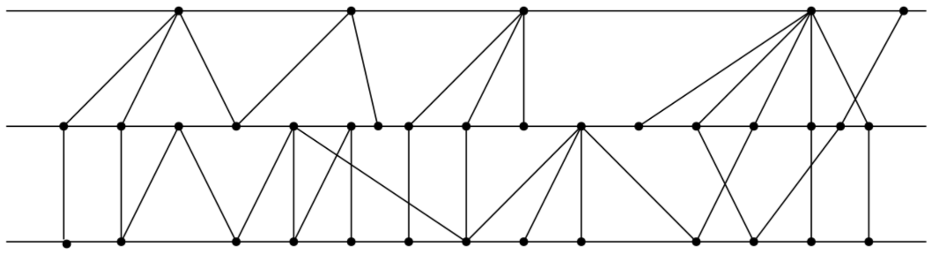


Figure 2.1: An example of a smooth tree of height 2. All arcs go from some level i to the next level $i+1$, $i \in \{0,1\}$.

Definition 2.2. *Let $P = x_1, x_2, .., x_n$ be a walk in a digraph G. We assign weights to the edges of G as follows: $w(x_i, x_{i+1}) = 1$ if $x_i x_{i+1}$ is a forward arc and $w(x_i, x_{i+1}) = -1$ if $x_i x_{i+1}$ is a backward arc. The **net length** of the walk P is defined to be $\sum_{1 \leq i < n} w(x_i, x_{i+1})$.*

In other words, the net length of the walk P is the difference between the number of the forward arcs and the number of the backward arcs in the walk P.

Definition 2.3. *If a closed walk $W = w_1, w_2, .., w_n$ has net length zero, then it is called a **balanced closed walk**. Otherwise, it is called an **unbalanced closed walk**.*

Definition 2.4. *If a digraph G contains an unbalanced closed walk then it is called an **unbalanced digraph**. Otherwise, it it called a **balanced digraph**.*

Proposition 2.5. *Every tree is balanced.*

Proof. Any closed walk C in a tree T traverses each arc the same numbers of times in the forward direction as in the backward direction. Therefore, the net length of any closed walk in T is zero. Hence, trees are balanced. □

Definition 2.6. *Let T be a tree. The height of T is the net length of a walk W in T such that W has the maximum net length among all walks in T.*

Definition 2.7. *Let $P = x_1, x_2, ..., x_n$ be a walk of net length k. The walk P is called **constricted**, if for any $0 \leq j \leq k$ the net length of the sub-walk $P[x_0, x_j]$ is non-negative and at most k.*

Lemma 2.8. *[18] Let W_1 and W_2 be two constricted walks of net length k. There exist a constricted path P of net length k, and homomorphisms $f_1 : V(P) \to V(W_1)$ and $f_2 : V(P) \to V(W_2)$ such that each f_i,$i = 1, 2$, maps the starting vertex of P to the starting vertex of W_i and the ending vertex of P to the ending vertex of W_i.*

In other words, let W_1 and W_2 be two constricted walks of net length k, there exist two constricted and congruent walks $P_i^{'}$, $i = 1, 2$, of net length k from the starting vertex of W_i to the ending vertex of W_i such that $V(P_i^{'}) \subseteq V(W_i)$.
The constricted path P is called the ***common pre-image*** of W_1 and W_2.

2.2 Layouts

Since each tree T is balanced, we can represent it on n levels (parallel lines) so that each arc of T goes from some level i to the next level $i+1$ (here n is the height of T, and the levels are numbered $0, 1, ..., n$). Let $V(i)$ denote all vertices of T on level i.

Proposition 2.9. *Let T be a tree of height n. Then T admits a min ordering if and only if each set $V(i)$ has an ordering π_i, $0 \leq i \leq n$, such that if $\pi_i(a) < \pi_i(b)$, $\pi_{i+1}(c) < \pi_{i+1}(d)$ and $ad, bc \in E(T)$, then ac is also an arc of T.*

Proof. First we prove the necessity. Suppose that for each i, $0 \leq i \leq n$, we have an ordering π_i for the vertices $V(i)$ such that if $\pi_i(a) < \pi_i(b)$, $\pi_{i+1}(c) < \pi_{i+1}(d)$ and $ad, bc \in E(T)$, then ac is also an arc of T. The ordering π defined below is a min ordering for T.

$$\pi(v) = \begin{cases} \pi_0(v) & \text{if } v \in V(0) \\ \pi_i(v) + \Sigma_{0 \leq k \leq i-1} |V(k)| & \text{if } v \in V(i), i > 0 \end{cases}$$

Indeed, let ad be an arc of T goes from some level i to the next level $i+1$, and let bc be an arc of T goes from some level j to the next level $j+1$. Suppose that i is equal to j. By the definition of π, we have $\pi(a) = \pi_i(a) + k$, $\pi(b) = \pi_i(a) + k$, for some constant k, and we have $\pi(c) = \pi_{i+1}(c) + k^{'}$, $\pi(d) = \pi_{i+1}(d) + k^{'}$, for some constant $k^{'}$. Therefore, if $\pi(a) < \pi(b)$ and $\pi(c) < \pi(d)$, then $\pi_i(a) < \pi_i(b)$ and $\pi_{i+1}(c) < \pi_{i+1}(d)$. Hence, based on the assumption we know that there exists an arc between a and c.
Now without loss of generality suppose that i is less than j. Let u be a vertex in $V(s)$, $0 \leq s \leq n$, and let v be a vertex in $V(t)$, $0 \leq s < t \leq n$. By the definition of the ordering π, the value of $\pi(u)$ is less than $\pi(v)$. Therefore, we have $\pi(a) < \pi(b)$ and $\pi(d) < \pi(c)$. In addition, based on the assumption we know that there exists an arc between a and d.

Second, we prove the sufficiency. Suppose that π is a min ordering for T. For each i, $0 \leq i \leq n$, we define an ordering π_i for the vertices $V(i)$ as follows:

$$\pi_0(v) = \pi(v)$$
$$\pi_i(v) = \pi(v) - \Sigma_{0 \leq k \leq i-1} |V(k)|$$

Suppose that ad and bc are two arcs in T such that $\pi_i(a) < \pi_i(b)$, $\pi_{i+1}(c) < \pi_{i+1}(d)$ for some $0 \leq i \leq n-1$. By the definition of π_i's, we have $\pi_i(a) = \pi(a) - k$ and $\pi_i(b) = \pi(b) - k$, for some constant k, and we have $\pi_{i+1}(c) = \pi(c) - k^{'}$ and $\pi_{i+1}(d) = \pi(d) - k^{'}$, for some constant $k^{'}$. Therefore, we can see that $\pi(a) < \pi(b)$ and $\pi(c) < \pi(d)$. Since π is a min ordering, then ac is an arc of T. □

Let $\pi_0, \pi_1, .., \pi_n$ be a sequence of the orderings of the vertices $V(0), V(1), ..., V(n)$ respectively in the above proposition such that if $\pi_i(a) < \pi_i(b)$, $\pi_{i+1}(c) < \pi_{i+1}(d)$ and $ad, bc \in E(T)$, then ac is also an arc of T. The ordering π_i's are best visualized in the form of a layout. A ***layout*** of a tree T of height n is a placement of the vertices of T on n levels (parallel lines) so that each arc of T goes from a vertex on some level i to a vertex on the next level $i+1$ in a specific order on each line, so that configurations Z_1 and Z_2 do not occur in the layout (see Figure 2.2). We can observe that $\pi_0, \pi_1, .., \pi_n$ guarantee a layout by placing the vertices of each $V(i)$ on level i, $0 \leq i \leq n$, from left to right, in order of π_i. Conversely, each layout defines an ordering π_i for the vertices $V(i)$, $0 \leq i \leq n$, by enumerating the vertices of level i from left to right.

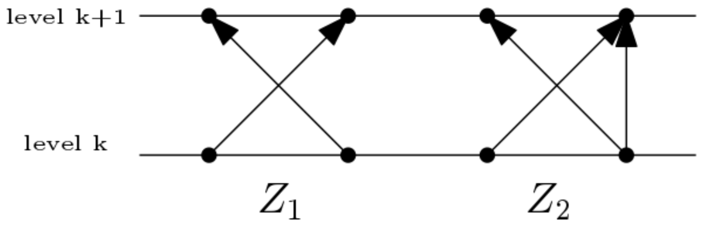


Figure 2.2: The configurations Z_1 and Z_2 for a layout.

Theorem 2.10. *Every tree T with maximum degree two has a min ordering.*

Proof. We may assume T is weakly connected. Now it is clear that T is an oriented path $P = v_1, v_2, \ldots, v_n$ in which $E = \{(v_{i-1}, v_i) | 2 \leqslant i \leqslant n\}$. Let $\pi : V \to \{1, .., n\}$ be an ordering with $\pi(v_i) = i$. We claim the ordering π defined above is a min ordering for T. Consider (v_i, v_j) and $(v_{i'}, v_{j'}) \in E$, without loss of generality we can easily see that $\pi(v_i), \pi(v_j) \leqslant \pi(v_{i'}), \pi(v_{j'})$; therefore, $min(\pi(v_i), \pi(v_{i'})) = \pi(v_i)$ and $min(\pi(v_j), \pi(v_{j'})) = \pi(v_j)$ so because $(v_i, v_j) \in E$ we can conclude that π is a min ordering for T.

□

2.3 Height One Trees

First, we shall study trees of height one. We will prove that the tree T_1 in Figure 2.3 is the only minimal *I-pair* tree of height one which is also the only minimal *SI-pair* tree of height one .

Theorem 2.11. *Let T be a tree of height one. T admits a min ordering if does not include T_1 or its reverse as an induced sub-graph.*

Proof. let (a, b) be a pair of vertices in T with the greatest distance of any pair of vertices in T. Let $P = w_1, w_2, , , , , w_{n-1}, w_n$ be the unique path from a to b.

First, we can see that T can not include a branch rooted at w_2 or w_{n-1} of length more than one, and a branch rooted at w_3 or w_{n-2} of length more than two, because including this kind of branches contradicts that (a, b) has the greatest distance.

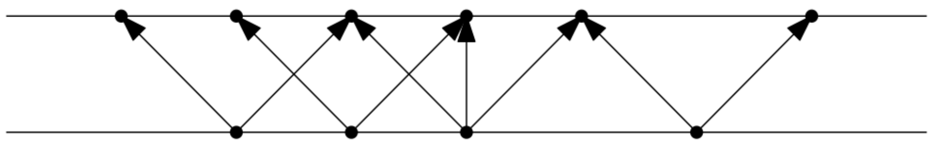


Figure 2.3: T_1: The only minimal tree of height one which has an invertible pair.

Second, T can not include a branch rooted at other vertices are placed on P of length more than two. Because existing such branch means that T_1 is an induced subgraph of T.

Without loss of generality assume that a and b are placed on level one. Now we introduce a min ordering for T as follows: Let π_0 be an ordering for the vertices w_i's with an even index i, $\pi_0(w_i) = k$ whenever $i = 2k$. Let π_1 be an ordering for the vertices w_i's with an odd index i, $\pi_1(w_i) = k$ whenever $i = 2k + 1$. We place the vertices w_i's with an odd index i on level 1, from left to right, in order of π_1 , and we place the vertices w_i's with an even index i on level 0, from left to right, in order of π_0. For every vertex $w_i \in V(P)$, we place the branches rooted at w_i around it by the way illustrated in the Figure 2.4. Figure 2.4 is a layout for T hence it admits a min ordering.

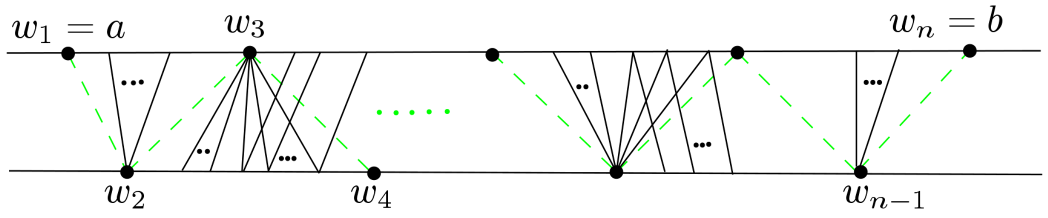


Figure 2.4: T_1: The path P is drawn by dashed lines. All arcs go from level 0 to level 1.

□

2.4 Patterns

Let $P = x_1, ..., x_n$ be a path in a digraph G. We define a sequence $S = s_1, s_2, ..., s_{n-1}$ with respect to P as follows: if x_ix_{i+1} is a forward arc then $s_i = F$. If x_ix_{i+1} is a backward arc then $s_i = B$. The sequence S is called the ***pattern*** of the path P. A pattern S followed by an asterisk $(*)$ matches zero or more occurrences of the pattern. By placing a part of a pattern inside parentheses, then we can group that part of the pattern. This allow us to apply an asterisk $(*)$ to part of a pattern.
For instance, let $P = x_1, .., x_n$ be a path with pattern $(BF)(BF)^*F$ such that $x_1 = a$. This means the path P going backward from the vertex a, then forward, possibly repeating backward, forward, but then ends with forward.

2.5 Height Two Smooth Trees

Let T be a smooth tree of height two. Suppose that c is a vertex which has $n >$ neighbors, $N(c) = \{a_1, a_2, ..., a_n\}$, $n > 2$. Let T_i be a subtree (or branch) rooted at a_i for every $1 \leq i \leq n$.
If c is on level 0, then the branches T_i's can be classified into following types: we say the branch T_i is of type:

- (A1) : If it contain a path $P = x_1, .., x_l$ with pattern $BF(BF)^*F$ such that $x_1 = a_i$.
- (A2) : If it contains a path $P = x_1, .., x_l$ with pattern $BF(BF)^*$ such that $x_1 = a_i$, and it is of overall height one.
- (A3) : If it is just a star from a_i towards any number of vertices on the level that c is placed.
- (A4) : If it contains a path $P = x_1, .., x_l$ with pattern $FB(FB)^*B$ such that $x_1 = a_i$.
- (A5) : If it contains a path $P = x_1, .., x_l$ with pattern $F(BF)^*$ such that $x_1 = a_i$, and it is of overall height one.

If c is on level one and its neighbors $N(c) = \{a_1, a_2, ..., a_n\}$ are on level two, then the branches T_i's can be classified into following types: we say the branch T_i is of type:

- (B1) : If it contains a path a path $P = x_1, .., x_l$ with pattern $BF(BF)^*BB$ such that $x_1 = a_i$.
- (B2) : If it contains a path $P = x_1, .., x_l$ with pattern $BF(BF)^*$ such that $x_1 = a_i$, and it is of overall height one.
- (B3) : If it is just a star from a_i toward any number of vertices on the level that c is placed.
- (B4) : If it contains a path $P = x_1, .., x_l$ with pattern $BB(FB)^*FF$ such that $x_1 = a_i$.
- (B5) : If it contains a path $P = x_1, .., x_l$ with pattern $BB(FB)^*$ such that $x_1 = a_i$ with no vertices $x \neq a_i$ on the level that a_i is placed.

If c is on level two, then the branches T_i's can be classified into following types: we say the branch T_i is of type:

- (C1) : If it contain a path $P = x_1, .., x_l$ with pattern $FB(FB)^*B$ such that $x_1 = a_i$.
- (C2) : If it contains a path $P = x_1, .., x_l$ with pattern $FB(FB)^*$ such that $x_1 = a_i$, and it is of overall height one.

- (C3) : If it is just a star from a_i towards any number of vertices on the level that c is placed.
- (C4) : If it contains a path $P = x_1, .., x_l$ with pattern $BF(BF)^*F$ such that $x_1 = a_i$.
- (C5) : If it contains a path $P = x_1, .., x_l$ with pattern $B(FB)^*$ such that $x_1 = a_i$, and it is of overall height one.

If c is on level one and its neighbors $N(c) = \{a_1, a_2, ..., a_n\}$ are on level zero, then the branches T_i's can be classified into following types: we say the branch T_i is of type:

- (D1) : If it contains a path a path $P = x_1, .., x_l$ with pattern $FB(FB)^*FF$ such that $x_1 = a_i$.
- (D2) : If it contains a path $P = x_1, .., x_l$ with pattern $FB(FB)^*$ such that $x_1 = a_i$, and it is of overall height one.
- (D3) : If it is just a star from a_i toward any number of vertices on the level that c is placed.
- (D4) : If it contains a path $P = x_1, .., x_l$ with pattern $FF(BF)^*FF$ such that $x_1 = a_i$.
- (D5) : If it contains a path $P = x_1, .., x_l$ with pattern $FF(BF)^*$ such that $x_1 = a_i$ with no vertices $x \neq a_i$ on the level that a_i is placed.

When we represent a family of trees T in a way illustrated in Figure 2.5 means that every tree G in the family T is constructed as follows: for every vertex a_i, $1 \leq i \leq n$, one of the subtrees $T_1, ..., T_k$ must be rooted at a_i.

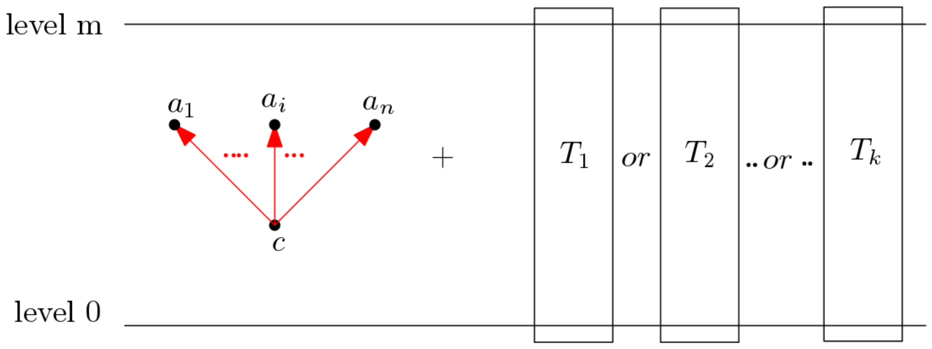


Figure 2.5: A way to represent a family of trees.

For instance, in Figure 2.6, the tree G is a member of family T. The subtrees T_2,T_2,T_3 rooted at the vertices a_1,a_2,a_3 respectively.

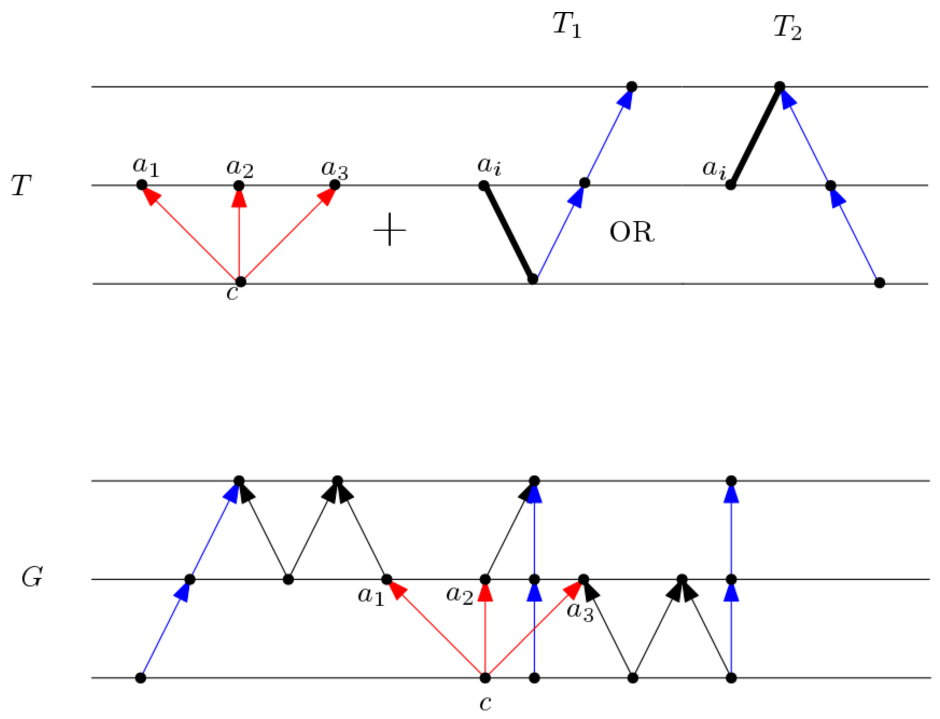


Figure 2.6: The tree G is a member of family T. The thick line can be an arc or a path of height one.

Lemma 2.12. *Each tree from the families O_1, O_2 and O_3 contains an I-pair.*

Proof. Suppose that T is a tree of family O_s, for some $1 \leq s \leq 3$. Let $P[h_i, h_j]$, $1 \leq i \neq j \leq 3$, be a constricted path of net length two from h_i to h_j in T. Let $P[h_i, l_i]$, $1 \leq i \leq 3$, be a constricted path of net length two from h_i to l_i in T. There is no arc uv or vu such that $u \in V(P[h_i, l_i])$ and $v \in P[h_i, h_j]$ if so we have a cycle which is a contradiction. By Lemma 2.8, for every three distinct integers $1 \leq i, j, k \leq 3$, there exist two congruent walks $W[h_i, h_j]$ and $W[h_k, h_k]$ from h_i to h_j and h_k to h_k in T which are pre-images of $P[h_i, h_j]$ and $P[h_i, l_i]$ respectively that avoid each other. Therefore, suppose that $W_1 = W[h_1, h_2] + W[h_2, h_2] + W[h_2, h_3]$ is a walk from h_1 to h_3 in T, and $W_1 = W[h_3, h_3] + W[h_3, h_1] + W[h_1, h_1]$ is a walk from h_3 to h_1 in T. We can conclude that W_1 and W_2 are two congruent walks from h_1 to h_3 and h_3 to h_1 respectively that avoid each other. Therefore, (h_3, h_1) is an *I-pair* in T. By similarity, we can see that every pair (h_i, h_j), $1 \leq i \neq j \leq 3$, is an *I-pair*. $\square$

Lemma 2.13. *Each tree from the families O_1, O_2 and O_3 contain a DAT.*

Proof. Suppose that T is a tree of family O_s, for some $1 \leq s \leq 3$. We shall to show that l_1, l_2, l_3 is a DAT. Let l_i be an arbitrary vertex from l_1, l_2, l_3. Moreover, let l_k and l_j be the other two vertices from l_1, l_2, l_3. There exists a constricted path $P[l_i, h_i] =$ of net length 2

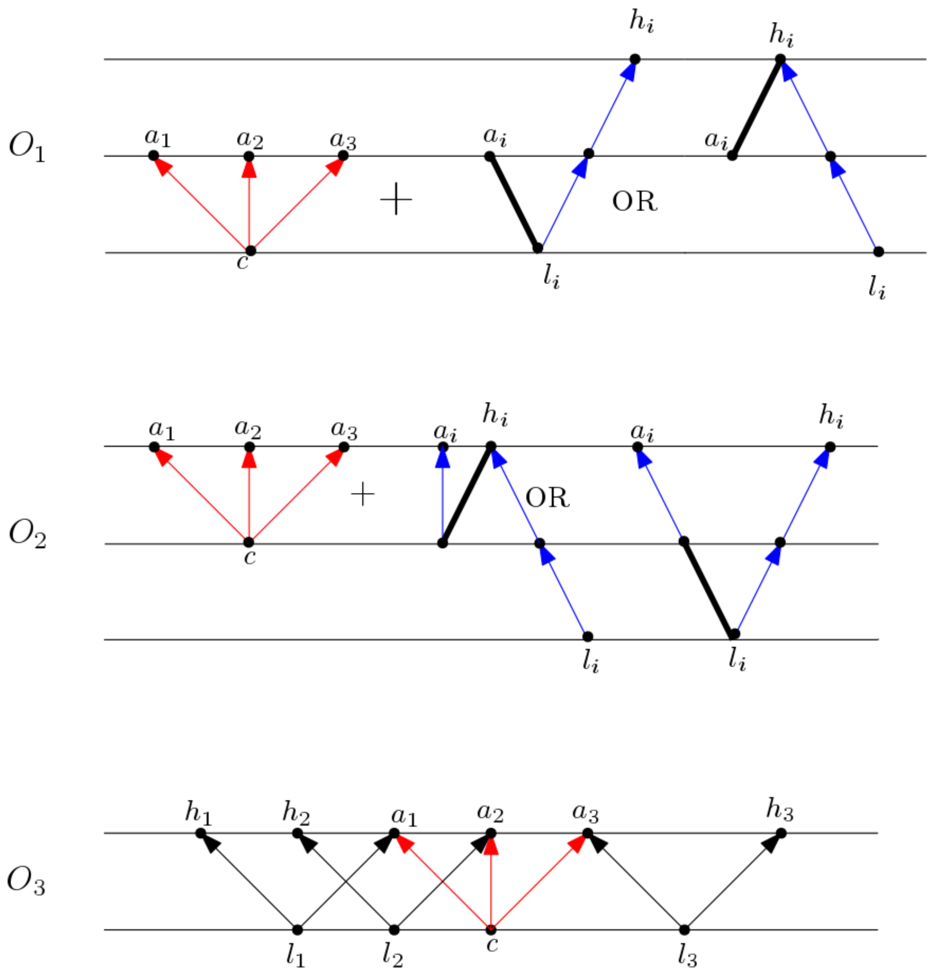


Figure 2.7: The thick lines can be an arc or a path of height one.

from l_i to h_i in T, $i = 1, 2, 3$, and there exists a constricted path $P[l_j, h_i] =$ of net length two from l_j to h_i in T. In addition, there exists a constricted path $P[l_k, h_k]$ of net length two from l_k to h_k. Now we can apply Lemma 2.8, let X, Y, Z be congruent walks that are pre-images of $P[l_i, h_i]$, $P[l_j, h_i]$, $P[l_k, h_k]$ respectively such that $V(X) \subseteq V(P[l_i, h_i])$, $V(Y) \subseteq V(P[l_j, h_i])$ and $V(Z) \subseteq V(P[l_k, h_k])$. We claim that X avoids Y and Z because there is no arc uv or vu such that $u \in V(P[l_k, h_k])$ and $v \in V(P[l_i, h_i]) \cup V(P[l_j, h_i])$, if so we have a cycle which is a contradiction. On the other hand, By lemma 2.12, we proved that every pair (h_r, h_l), $1 \leq r \neq l \leq 3$, is an *I-pair*. Hence (h_k, h_i) is an *I-pair* so we can conclude that l_1, l_2, l_3 is a DAT.

□

Theorem 2.14. *Let T be a smooth tree of height two. The tree T has a min ordering if and only if T satisfies the following conditions:*

- *(1) for every vertex $c \in V(T)$ which is on level 0, with $n > 2$ out-neighbors $a_1, a_2, ..., a_n$ on level 1, no three branches of types A1 or A2 , and no three branches of types A1 or A4 rooted at a_i's, and*
- *(2) for every vertex $c \in V(T)$ which is on level 1, with $n > 2$ out-neighbors $a_1, a_2, ..., a_n$ on level 2, no three branches of types B1 or B2 and no three branches of types B1 or B4 rooted at a_i's, and*
- *(3) for every vertex $c \in V(T)$ which is on level 1, with $n > 2$ in-neighbors $a_1, a_2, ..., a_n$ on level 0, no three branches of types C1 or C2, and no three branches of types C1 or C4 rooted at a_i's, and*
- *(4) for every vertex $c \in V(T)$ which is on level 2, with $n > 2$ in-neighbors $a_1, a_2, ..., a_n$ on level 1, no three branches of types D1 or D2, and no three branches of types D1 or D4 rooted at a_i's.*

Proof. First we prove sufficiency. If T does not satisfy the conditions (1-4), then we have one the following cases.

Case 1: there exists a vertex $c \in V(T)$ on level 0, with $n > 2$ out-neighbors $a_1, a_2, ..., a_n$ on level 1, such that there exist three branches of types A1 or A2, or three branches of types A1 or A4 rooted at a_i's. If there exist three branches of types A1 or A2 rooted at a_i's, then T contains O_3 as an induced subgraph. If there exist three branches of types A1 or A4 rooted at a_i's, then T contains O_1 as an induced subgraph. In this case, T contains O_1 or O_3 hence it contains an *I-pair*. therefore, T does not admit a min ordering.

Case 2: there exists a vertex $c \in V(T)$ on level 1, with $n > 2$ out-neighbors $a_1, a_2, ..., a_n$ on level 2, such that there exist three branches of types B1 or B2 or three branches of types B1 or B4 rooted at a_i's. If there exist three branches of types B1 or B2 rooted at a_i's, then T contains O_3 as an induced subgraph. If there exist three branches of types B1 or B4 rooted at a_i's, then T contains O_2 as an induced subgraph. In this case, T contains O_2 or O_3 hence it contains an *I-pair*. therefore, T does not admit a min ordering.

Case 3: there exists a vertex $c \in V(T)$ on level 1, with $n > 2$ in-neighbors $a_1, a_2, ..., a_n$ on level 0, such that there exist three branches of types C1 or C2 or three branches of types C1 or C4 rooted at a_i's. If there exist three branches of types C1 or C2 rooted at a_i's, then T contains the reverse of O_3 as an induced subgraph. If there exist three branches of types C1 or C4 rooted at a_i's, then T contains the reverse of O_2 as an induced subgraph. In this case, T contains the reverse of O_2 or the reverse of O_3 hence it contains an *I-pair*. therefore, T does not admit a min ordering.

Case 4: there exists a vertex $c \in V(T)$ on level 2, with $n > 2$ in-neighbors $a_1, a_2, ..., a_n$ on level 1, such that there exist three branches of types A1 or A2 or three branches of types D1 or D4 rooted at a_i's. If there exist three branches of types D1 or D2 rooted at a_i's, then T contains the inverse of O_3 as an induced subgraph. If there exist three branches of types D1 or D4 rooted at a_i's, then T contains the inverse of O_1 as an induced subgraph. In this case, T contains the inverse of O_1 or the inverse of O_3 hence it contains an *I-pair*. therefore, T does not admit a min ordering.

Second we prove necessity. We proceed by induction on $|V(T)|$. To begin the induction, by Theorem 2.10 any directed path admits a min ordering. Moreover, it is easy to see that any tree with < 4 vertices is a path. Suppose that every tree with size less than k which satisfies the conditions admits a min ordering. Now let T be a tree of size k which satisfies the above conditions. For every vertex $c \in V(T)$, with $n > 2$ neighbors, $a_1, a_2, ..., a_n$, let $B_c = \{T_1, T_2, .., T_n\}$ be the set of branches of c such that the branch T_i rooted at a_i. We have one of the following cases:
If c is on level 0, then B_c satisfies one of the following cases:

Case 1: B_c contains two branches of type A1 and no branches of types A2 and A4.
If there exists a branch $T_i \in B_c$ of type A3, then we remove T_i from T. By the inductive hypothesis, the subtree $T \setminus T_i$ has a layout R and a corresponding min ordering π_R. Now we can insert T_i in the layout R of $T \setminus T_i$ by preserving the properties of the layout R as follows: let x be the vertex on level 0 such that $\pi_R(x) = \pi_R(c) + 1$. We can place T_i in the free space between c and x on level zero (see Figure 2.8).
If there exist a branch T_j of type A5, then we remove T_j from T. By the inductive hypothesis, $T \setminus T_j$ has a layout R and a corresponding min ordering π_R. Now we can insert T_j in the layout R of $T \setminus T_j$ by preserving the properties of the layout R as follows: let a_s and a_t be the vertices on level one such that $\pi_R(a_s) = \pi_R(a_j) - 1$ and $\pi_R(a_t) = \pi_R(a_j) + 1$. Now let $x \in V(T_s)$ be a vertex on level two with the maximum value of π_R. Also let $y \in V(T_t)$ be a vertex on level two with the minimum value of π_R. The vertices x, y, a_s and a_t form a free space between the level one and the level two in R. By the inductive hypothesis, T_j has also a layout R'. Therefore, we can insert T_j in the free space which is formed by the four vertices (see Figure 2.9).

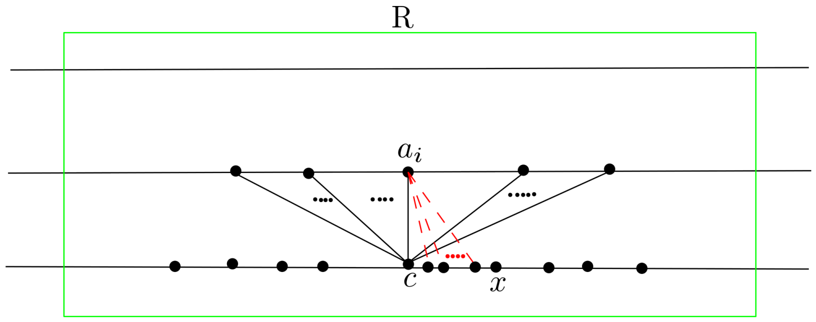


Figure 2.8: Dashed arcs form T_i which is a star from a_i toward any number of vertices on level 0.

Case 2: B_c contains one branch of type A1, at most one branch of type A2, and at most one branch of type A4.

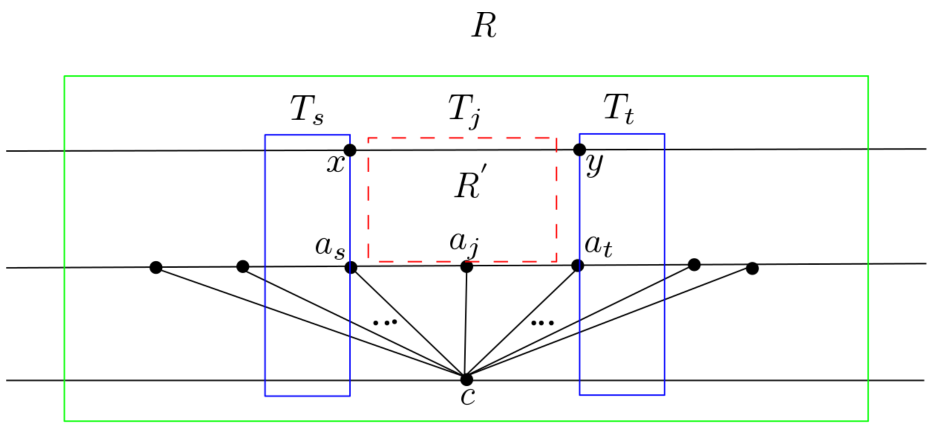


Figure 2.9: The dashed box is the subtree T_j which is placed between T_s and T_t in the layout R.

If there exists a branch $T_i \in B_c$ of types A3 or A5, then we have the same argument as in the previous case. Otherwise, we have $B_c = \{T_1, T_2, T_3\}$ such that T_1, T_2, T_3 are the branches of types $A1$,$A2$ and $A4$ respectively. We remove T_2 from T. By the inductive hypothesis, $T \setminus T_2$ has a layout R and a corresponding min ordering π_R. If an arc $e_1 \in T_1$ crosses an arc $e_2 \in T_3$ in the layout R, then we have the configuration Z_1 (see Figure 2.7) in R which is not possible. Therefore, without loss of generality T_1 is placed in the left side of c in R, and T_3 is placed in the right side of c in R such that they do not intersect each other. Let $x \in V(T_3)$ be a vertex on level zero with the minimum value of π_R. Moreover, let y be a vertex on level one which is the neighbor of x. Therefore, we can place T_2 in the free space in R which is formed by the four vertices c, a_3, x and y between the level zero and the level one (see Figure 2.10).

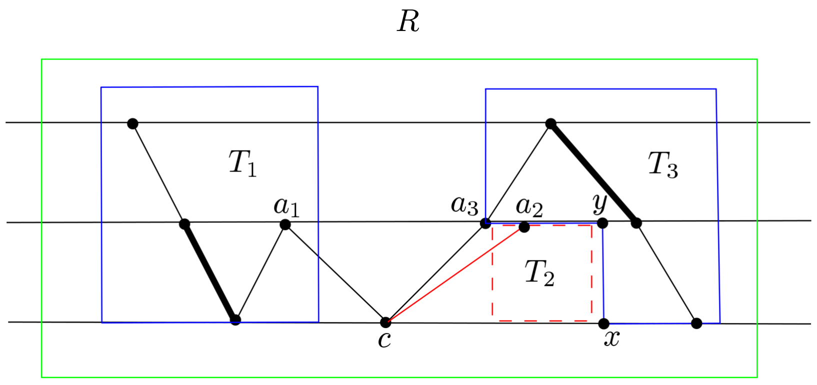


Figure 2.10: The dashed box is T_2 which is placed under T_3 in the layout R.

Case 3: B_c contains no branches of type A1, at most two branches of type A2, and at most two branches of type A4.

If there exists a branch $T_i \in B_c$ of type A3 or A5, then we have the same argument as in the previous case. Otherwise, it must contains a branch T_i of type A2 and a branch T_j of type A4 (c has more than two neighbors). By the inductive hypothesis, $T \setminus T_i$ has a layout R and a corresponding min ordering π_R. Without loss of generality, suppose that the branch T_j is placed in the right side of c. Let $x \in V(T_j)$ be a vertex on level zero with the minimum value of π_R. Moreover, let y be a vertex on level one which is the neighbor of x. Therefore, we can place T_i in the free space in R which is formed by the four vertices c, a_j, x and y between the level zero and the level one (see Figure 2.11).

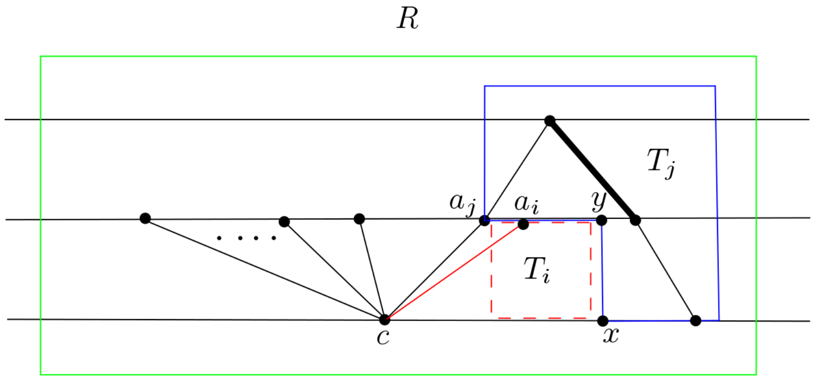


Figure 2.11: The dashed box is T_i which is placed under T_j in the layout R.

If c is on level 1, and has neighbors on level 2, then B_c satisfy one of the following cases:

Case 1: B_c contains two branches of type B1 and no branches of types B2 and B4. Without loss of generality, let T_1,$T_2 \in B_c$ be the branches of type B1 . If there exists a branch $T_i \in B_c$ of type B3, then we remove T_i from T. By the inductive hypothesis, the subtree $T \setminus T_i$ has a layout R and a corresponding min ordering π_R. Now we can insert T_i in the layout R of $T \setminus T_i$ by preserving the properties of the layout R as follows: let x be a vertex on level 1 such that $\pi_R(x) = \pi_R(c) + 1$. We can place T_i in the free space between c and x on level zero (see Figure 2.12).

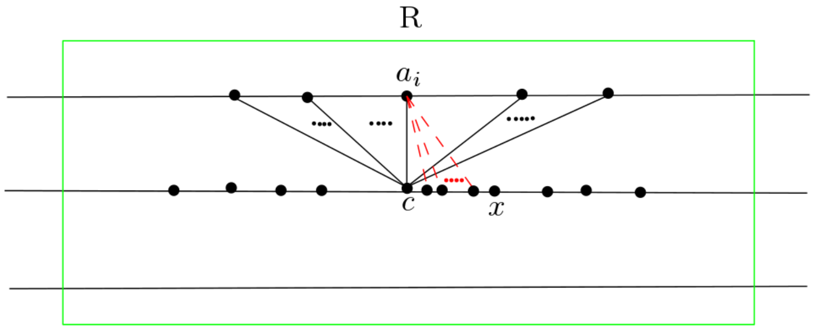


Figure 2.12: Dashed arcs form T_i which is a star from a_i toward any number of vertices on level 1.

If there exists a branch T_j of type B5, then we remove T_j from T. By the inductive hypothesis, $T \setminus T_j$ has a layout R and a corresponding min ordering π_R. If an arc $e_1 \in T_1$ crosses an arc $e_2 \in T_2$ in the layout R, then we have the obstruction Z_1 (see Figure 2.7) in R which is not possible. Therefore, without loss of generality suppose that T_1 is placed in the left side of c in R, and T_2 is placed in the right side of c in R such that they do not intersect each other. Let $x \in V(T_2)$ be a vertex on level zero with the minimum value of π_R. Also let y be the neighbor of x on the level one. Moreover, let z be a vertex on level one such that $\pi_R(x) = \pi_R(c)+1$. Let $v_1, v_2, .., v_k$ be the neighbors of a_j on level one, and let H_l be a branch rooted at v_l, for every $1 \leq l \leq k$. We place the vertices $v_1, v_2, .., v_k$ between c and z, and place the branches $H_1, ..., H_k$ in the free space which is formed by the vertices c, x, y under the branch T_2 in R (see Figure 2.13).

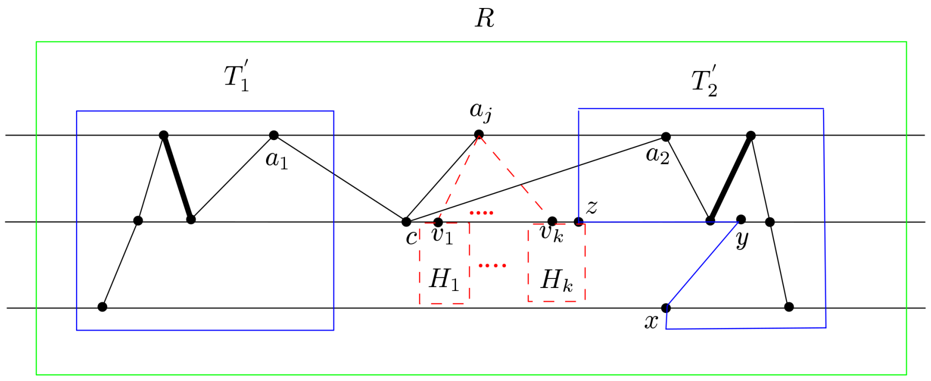


Figure 2.13: The dashed boxes H_1,...,H_k and dashed arcs altogether form T_j.

Case 2: B_c contains one branch of type B1, at most one branch of type B2, and at most one branch of type B4.

If there exists a branch $T_i \in B_c$ of type B3 or B5, then we have the same argument as in the previous case. Otherwise, we have $B_c = \{T_1, T_2, T_3\}$ such that T_1, T_2, T_3 are the branches of types $B1$,$B2$ and $B4$ respectively. We remove T_3 from T. By the inductive hypothesis, $T \setminus T_3$ has a layout R and a corresponding min ordering π_R. If an arc $e_1 \in T_1$ crosses an arc $e_2 \in T_2$ in the layout R, then we have the configuration Z_1 (see Figure 2.7) in R which is not possible. Therefore, without loss of generality T_1 is placed in the left side of c in R, and T_2 is placed in the right side of c in R such that they do not intersect each other. Let $v_1, v_2, .., v_k$ be the neighbors of a_3 on level one, and let H_l be a branch rooted at v_l, for every $1 \leq l \leq k$. There exist only one branch H_r, $1 \leq r \leq k$, which contains a vertex on level two. Otherwise, we have three branches of types D1 or D4 rooted at a_3 which is a contradiction. Let z be a vertex on level one such that $\pi_R(x) = \pi_R(c) + 1$. We place the vertices $v_1, v_2, .., v_k$ between c and z such that v_r is placed immediately before z. We place the branches $H_1, ..., H_k$ in the free space under T_2', and H_r in the free space right side of T_2 (see Figure 2.14).

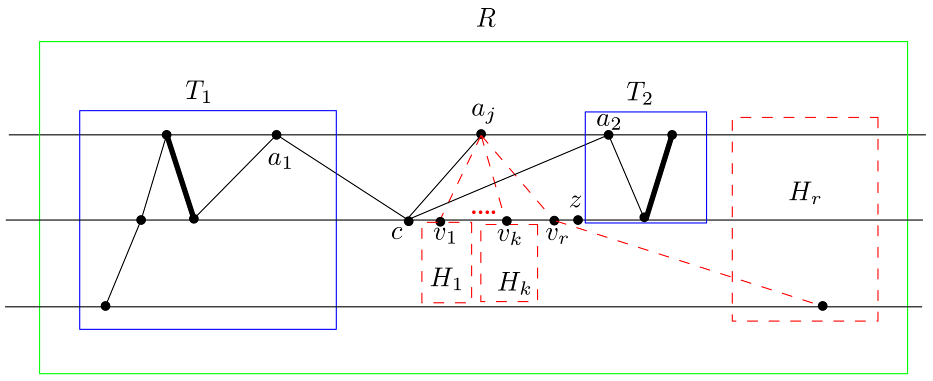


Figure 2.14: The dashed boxes H_1,...,H_k and dashed arcs altogether form T_j.

Case 3: B_c contains no branches of type B1, at most two branches of type B2, and at most two branches of type B4.
If there exists a branch $T_i \in B_c$ of type B3 or B5, then we have the same argument as in the previous case. Otherwise, it must contains a branch T_i of type B2 and a branch T_j of type B4 (c has more than two neighbors). By the inductive hypothesis, $T \setminus T_j$ has a layout R and a corresponding min ordering π_R. Without loss of generality, suppose that T_i is placed in the right side of c. Let $v_1, v_2, .., v_k$ be the neighbors of a_j on level one, and let H_l be a branch rooted at v_l, for every $1 \leq l \leq k$. First, suppose that there exist a branch $T_s \in B_c$ of type B_4, $s \neq j$, then there must exist only one branch H_r, $1 \leq r \leq k$ which contains a vertex on level two. Otherwise, we have three branches of types D1 or D2 rooted at a_3 which is a contradiction. In this case, Let z be a vertex on level one such that $\pi_R(x) = \pi_R(c) + 1$. We place the vertices $v_1, v_2, .., v_k$ between c and z such that v_r is placed immediately before z. We place the branches $H_1, ..., H_k$ in the free space under T_i, and place H_r in the free space right side of T_i. Second, suppose that T_i is the only branch of type B4 in B_c, then there could exist at most two branches H_r and H_s of type B4, $1 \leq r, s \leq k$. In this case, let z be a vertex on level one such that $\pi_R(x) = \pi_R(c) + 1$. We place the vertices $v_1, v_2, .., v_k$ between c and z such that v_r is placed immediately before z and v_s is placed immediately after c. Let p and q be the vertices on level two which has minimum and maximum value of π_R respectively. We place the branches $H_1, ..., H_k$ in the free space under T_i, and place H_r in the free space right side of q in R, and place H_s in the free space left side of p in R (see Figure 2.15).

If c is on level one, and has $n > 2$ neighbors on level 0, then the branches T_i's are classified into following types $D1, D2, D3, D4, D5$. It it easy to see that the patterns of the corresponding paths of types $D1, D2, D3, D4, D5$ are reverse of the patterns of the

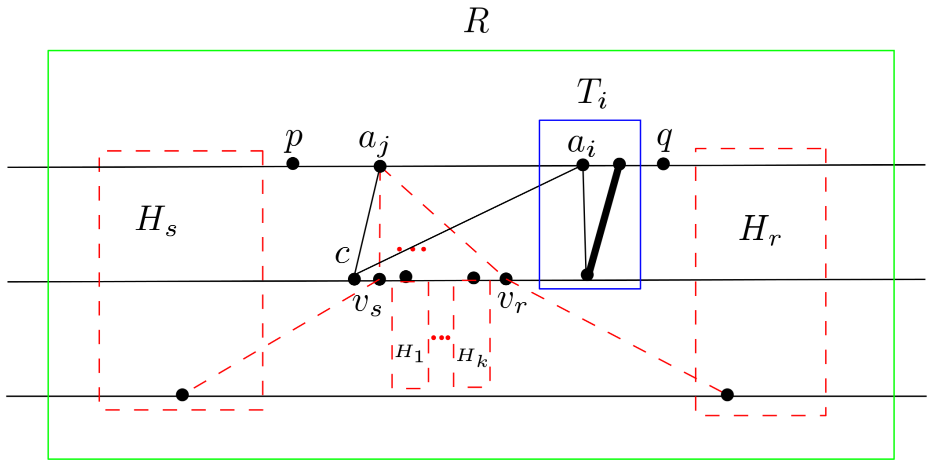


Figure 2.15: The dashed boxes H_1,...,H_k and dashed arcs altogether form T_j.

corresponding paths of types $B1, B2, B3, B4, B5$ respectively. Therefore, by symmetry we have a same argument as in the case when it has $n > 2$ neighbors on level two.

In addition, if c is on level two, then the branches T_i's are classified into following types $C1, C2, C3, C4, C5$. It it easy to see that the patterns of the corresponding paths of types $C1, C2, C3, C4, C5$ are reverse of the patterns of the corresponding paths of types $A1, A2, A3, A4, A5$ respectively. Therefore, by symmetry we have a same argument as in the case when the vertex c is on level one.

□

Theorem 2.15. *Let T be a smooth tree of height less than three. The following statements are equivalent:*

- *(1) T has a DAT*
- *(2) T does not admit a min ordering*
- *(3) T contains O_1, O_2, O_3 or their reverse as an induced subgraph (see Figure 2.7).*

Proof. To show that 1 implies 2, suppose that T has a DAT, then it must have an *I-pair*. Therefore, by Lemma 1.66, it does not admit a min ordering.

To see that 2 implies 3. Suppose by contradiction that T does not contain O_1, O_2,O_3 and their reverse as an induced subgraph, then T satisfies one of the conditions (1-4) in Theorem 2.14. Therefore, T admits a min ordering. Finally, it remains to show that 3 implies 1. Assume that T contains O_1 or O_2 or O_3 or their reverse as induced subgraphs, then by Lemma 2.13, there exists a DAT in T. □

Corollary 2.16. *Let T be a smooth tree of height less than three. If T contains O_1, O_2, O_3 or their reverse as an induced subgraph, then $LHOM(T)$ is NP-complete. Otherwise it is polynomial time solvable.*

Proof. Suppose that T contains O_1, O_2, O_3 or their reverse as an induced subgraph, then by Theorem 2.15, it has a DAT. Therefore, by Theorem 1.76 we can conclude that the problem $LHOM(T)$ is NP-complete. Otherwise T satisfies one of the condition (1-4) in Theorem 2.14 so it admits a min ordering. Hence by Theorem 1.40 we can say that the problem $LHOM(T)$ is polynomial time solvable. □

For smooth trees of height less than three, we proved that having an I- pair is equivalent to having a DAT (Theorem 2.15). We believe that this is true for trees of arbitrary height. However, in general there exists a digraph which has an I-pair but it does not have a DAT (e.g. a directed cycle of length four).

Chapter 3

Remarks on General Trees

In this chapter, first we will discuss the possible results for trees of height two which are not smooth, and present a list of examples of minimal trees of height two that do not admit a min ordering. We will introduce some new forbidden structures by relaxing and tightening the conditions of the invertible pairs. It turns out that these new forbidden structures, which we call symmetrically invertible pairs and strictly half invertible pairs are useful for finding a concrete forbidden induced subtree characterization to have a min ordering, as suggested by Theorem 3.5.

3.1 Definitions

In this chapter we shall abbreviate "irreflexive oriented tree" to "tree". Recall that a pair of vertices (a, b) in a digraph H is called an ***invertible pair*** (*I-pair*) if it satisfies the following conditions:

- there exist congruent walks P from a to b and Q from b to a such that P avoids Q, and
- there exist congruent walks $P^{'}$ from b to a and $Q^{'}$ from a to b such that P' avoids $Q^{'}$.

Now we will define a special case of invertible pairs which is called symmetric invertible pairs.

Definition 3.1. *An* ***symmetrically invertible pair*** *(SI-pair) in a digraph G is a pair of vertices (a, b) such that there exist congruent walks P from a to b and Q from b to a such that P avoids Q and Q avoids P.*

Definition 3.2. *An* ***half-invertible pair*** *(HI-pair) in a digraph G is a pair of vertices (a, b) such that*

- *there exist congruent walks P from a to b and Q from b to a such that P avoids Q.*

The half-invertible pair is derived from relaxing the conditions of the invertible pair. It turns out to be useful for finding the minimal I-$pair$ trees. If (a, b) is an I-$pair$ then both (a, b) and (b, a) are HI-$pair$.

Definition 3.3. *A **strictly half-invertible pair** (SHI-pair) in a digraph G is an half-invertible pair which is not an invertible pair.*

Every HI-$pair$ is an SHI-$pair$ or an I-$pair$. If (a, b) is an I-$pair$, then (b, a) is also an I-$pair$. On the other hand, if (a, b) is an SHI-$pair$, then (b, a) is not an SHI-$pair$. Therefore, I-$pair$ and SI-$pair$ are symmetric, but HI-$pair$ and SHI-$pair$ are not.

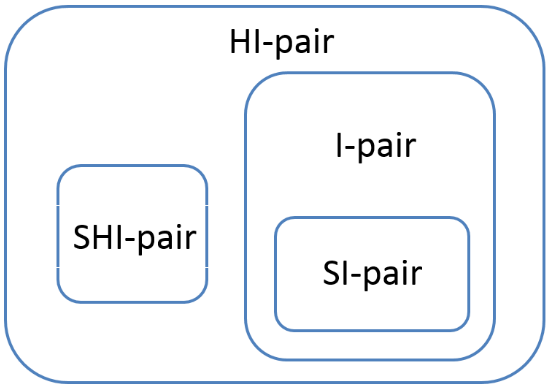


Figure 3.1: The relation among different types of invertible pairs.

Definition 3.4. *A tree T is called:*

- *a **minimal I-pair tree** if*
  - *T has an I-pair, and*
  - *No proper induced subgraph of T has an I-pair.*
- *a **minimal SI-pair tree** if*
  - *T has an SI-pair, and*
  - *No proper induced subgraph of T has an SI-pair.*
- *a **minimal SHI-pair tree** if*
  - *T has an SHI-pair, and*
  - *No proper induced subgraph of T has an SHI-pair.*
- *a **minimal HI-pair tree** if*
  - *T has an HI-pair, and*
  - *No proper induced subgraph of T has an HI-pair.*

3.2 Structural Relations Between HI-pair Trees

According to the definition of *HI-pair* trees, if T is an *HI-pair* tree then there is a pair of vertices (a, b) in T such that there exist congruent walks P and Q from a to b and b to a respectively. Therefore, one approach to characterizing the minimal *HI-pair* trees is by analyzing the corresponding walks. Here, we will show some properties of *HI-pair* trees, *SI-pair* trees and *I-pair* trees which are derived by analyzing the corresponding walks. Moreover, we will show a structural relation between *I-pair* trees and *SHI-pair* trees. According to this relation, every minimal *I-pair* tree is a minimal *SI-pair* tree or it is somehow constructed from two minimal *SHI-pair* trees. Hence, in order to find minimal *I-pair* trees, first, one can find minimal *SI-pair* trees and minimal *SHI-pair* trees. Second, try to find *I-pair* trees which are not *SI-pair* trees by joining two *SHI-pair* trees in some way.
We observe that all trees of height one are smooth. Moreover, we have found that all I-pair smooth trees of height less than three are SI-pair trees.

Theorem 3.5. *Every minimal I-pair tree T is a minimal SI-pair tree or there exist two distinct subsets of vertices $X, Y \subset V(T)$ such that induced subtrees $T(X)$ and $T(Y)$ are two minimal SHI-pair trees.*

Proof. By the definition of minimal *SI-pair* trees, the set of minimal *SI-pair* trees is a subset of minimal *I-pair* trees. Now suppose that T is a minimal *I-pair* tree which is not a minimal *SI-pair* tree. Let (a, b) be an *I-pair* in T. There exits two congruent walks $P = x_1, ..., x_n$ from a to b and $Q = y_1, ..., y_n$ from b to a such that P avoids Q. In addition, there exits two congruent walks $P^{'} = x^{'}_1, ..., x^{'}_m$ from b to a and $Q^{'} = y^{'}_1, ..., y^{'}_m$ from a to b such that $P^{'}$ avoids $Q^{'}$. We assumed that T is not a minimal *SI-pair* tree so there must exist an arc $y_i x_{i+1}$ for some i, $1 \leq i \leq n$, from Q to P, and there must exist an arc $y^{'}_i x^{'}_{i+1}$ for some j, $1 \leq j \leq m$, from $Q^{'}$ to $P^{'}$. Now let X be the union of the vertices $V(P)$ and $V(Q)$, and let Y be the union of the vertices $V(P^{'})$ and $V(Q^{'})$. The induced subtrees $T(X)$ and $T(Y)$ are two distinct minimal *SHI-pair* trees. □

For instance, the trees $T_1, ..., T_4$ (Figure 3.3) are some minimal *SI-pair* trees of height two, and the trees $T^{'}_1, T^{'}_2$ (Figure 3.2) are some minimal *SHI-pair* trees of height two. The trees $T^{''}_1, ..., T^{''}_6$ (Figure 3.4) are some minimal *I-pair* trees of height two which contain two distinct minimal *SHI-pair* trees as induced subtrees. More specifically, the trees $T^{''}_1, ..., T^{''}_5$ contain two minimal *SHI-pair* trees of type $T^{'}_1$ as induced subtrees, and $T^{''}_6$ contain two minimal *SHI-pair* trees of type $T^{'}_2$ as induced subtrees.

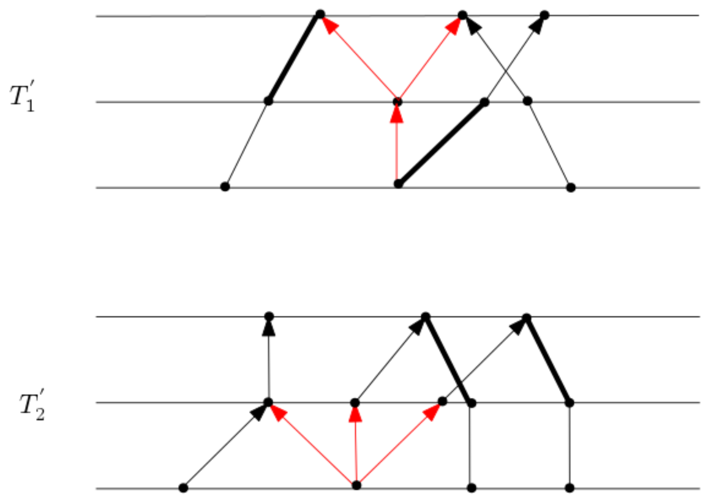


Figure 3.2: The known minimal *SHI-pair* trees of height 2. The thick line can be an edge or a path of overall height one.

3.3 Some Properties of HI-pair Trees

The following lemmas and theorems demonstrate some properties of *HI-pair* trees, *SI-pair* trees and *SHI-pair* trees. These lemmas and theorems can be useful when we study trees with the arbitrary height. According to Theorem 3.14, every *SI-pair* tree contains a vertex c such that when we remove that vertex, we have three disjoint subtrees with some required vertices at each subtree. Therefore, if we completely identify the properties of these required vertices, then we can characterize minimal *SI-pair* trees.

Definition 3.6. *Congruent walks $P = x_1, x_2, .., x_n$ from a to b, and $Q = y_1, y_2, .., y_n$ from b to a, are called a* ***simple pair of congruent walks*** *if there do not exist integers $1 \leq i < j \leq n$ such that $x_i = x_j$ and $y_i = y_j$.*

Proposition 3.7. *Let $P = x_1, x_2, .., x_n$ and $Q = y_1, y_2, .., y_n$ be two congruent walks from a to b and b to a respectively. There exist integers $1 \leq i < j \leq n$ such that $P[x_i, x_j]$ and $Q[y_i, y_j]$ are a pair of simple congruent walks.*

Proof. We prove the statement by induction on the length k of the walks.

If there do not exist integers $1 \leq i < j \leq n$ such that $x_i = x_j$ and $y_i = y_j$, then P and Q are a pair of simple congruent walks. Otherwise, suppose that for any pair of congruent walks C from a to b and D from b to a with length $k < n$, there exist integers $1 \leq s < t \leq n$ such that $P[x_s, x_t]$ and $Q[y_s, y_t]$ are a pair of simple congruent walks. Let $1 \leq i < j \leq n$ be the integers such that $x_i = x_j$ and $y_i = y_j$, then by deleting all vertices $x_i, ..., x_{j-1}$ in P and $y_i, ..., y_{j-1}$ in Q, we get a new pair of congruent walks P' from a to b and Q' from b to a. These walks have length less than n. By the inductive hypothesis, it contains a pair of simple congruent walks.

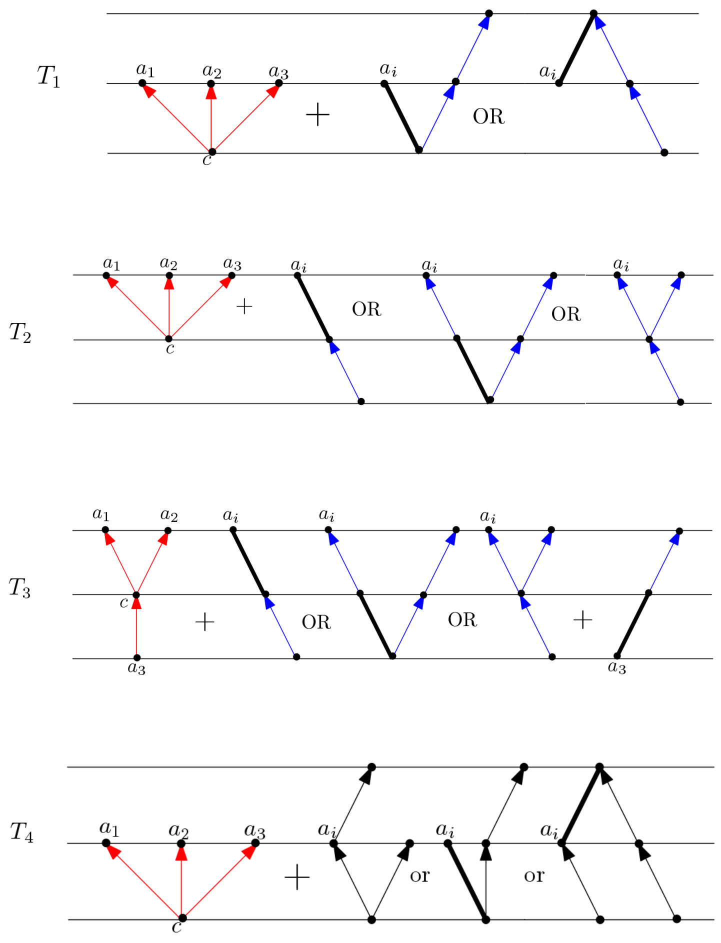


Figure 3.3: The known minimal *SI-pair* trees of height 2. The thick line can be an edge or a path of overall height one.

□

The following proposition is an immediate result of Proposition 3.7.

Proposition 3.8. *If (a, b) is an HI-pair, then there exist a pair of simple congruent walks P from b to a and Q from a to b such that P avoids Q.*

Lemma 3.9. *Let (a, b) be an HI-pair in a minimal HI-pair tree T. For any simple pair of congruent walks $P = x_1, x_2, ..., x_n$ from a to b and $Q = y_1, y_2, ..., y_n$ from b to a such that P avoids Q, a or b occur in the interior of Q or P.*

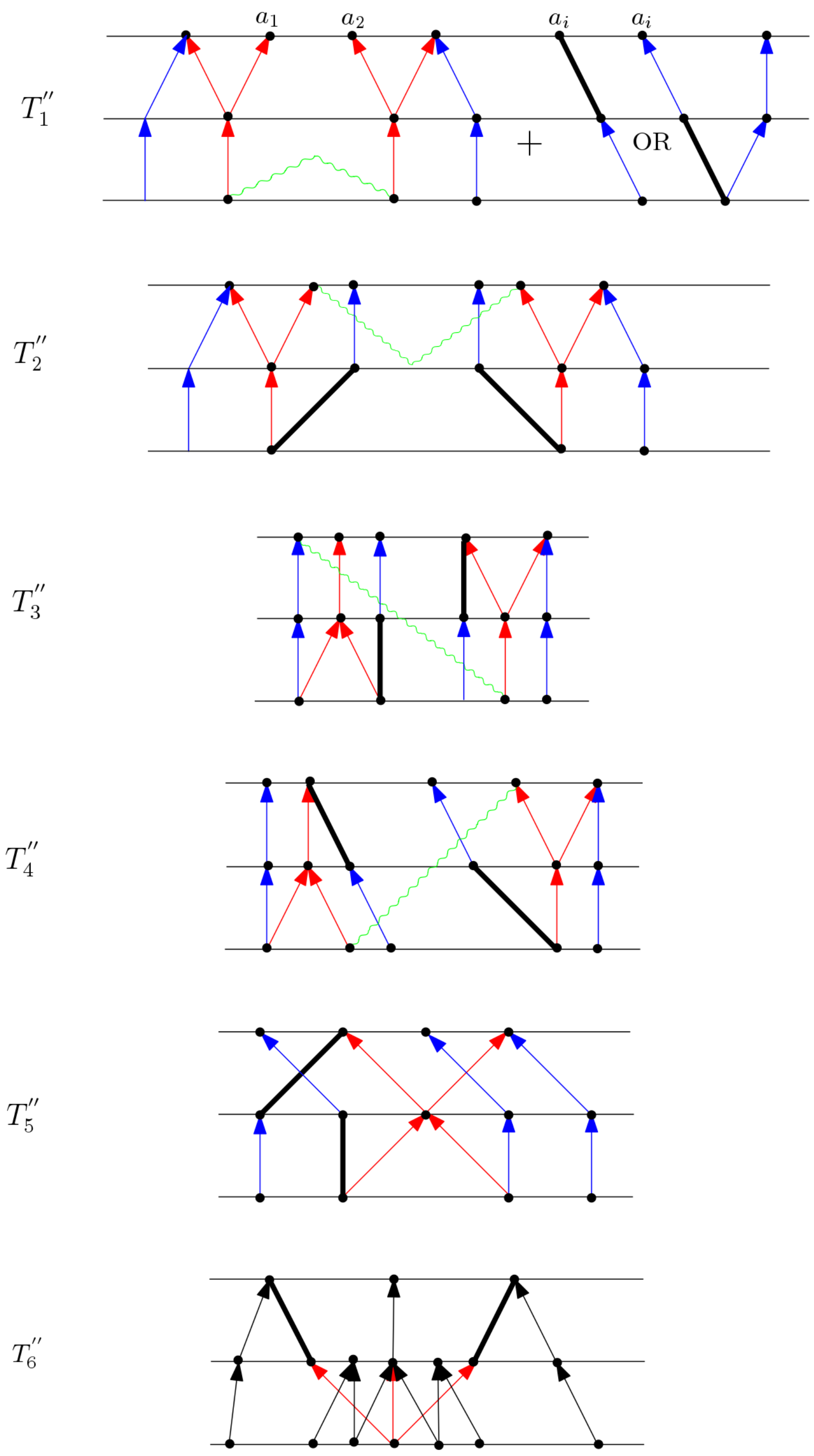


Figure 3.4: The known minimal *I-pair* trees of height 2 which are not *SI-pair* trees, and contain two *SHI-pair* trees as induced subtrees. The thick line can be an edge or a path of overall height one. The wiggly line can be any path between its two end points.

Proof. Assume that a and b do not occur in the interior of P and Q:

$$P = a \to t - s' \leftarrow b$$
$$Q = b \to s - t' \leftarrow a$$

- If $t = t'$,$s = s'$, then there exist congruent walks $P[t, s]$ from t to s and $Q[s, t]$ from s to t such that $P[t, s]$ avoids $Q[s, t]$ in the subtree induced by the vertices $P[t, s]$ and $Q[s, t]$ which is a tree. Therefore, (s, t) is an *HI-pair* in this subtree of T which is a contradiction with minimality of T.
- If $t \neq t'$ or $s \neq s'$, then without loss of generality assume that $s \neq s'$. Consider a vertex $y \neq b$ which is the left most vertex in Q appearing in P (it could be the vertex a). The walk $Q[b, y] + P[y, b]$ is a closed walk. Therefore, it contains a cycle in T which is a contradiction.

$$P = a \to t - - - - - - - - - - - - - - - y - - - - s' \leftarrow b$$
$$Q = b \to s - - - - y - - - - - - - - - - - - - - - t' \leftarrow a$$

□

Lemma 3.10. *Let (a, b) be an HI-pair in a minimal HI-pair tree T. There exist congruent walks $C = c_1, c_2, ..., c_n$ from a to b and $D = d_1, d_2, ..., d_n$ from b to a such that C avoids D and one of the following conditions holds:*

- *$\exists k$, $1 < k < n$, such that for every $i < k$, $d_i \neq a$, $d_k = a$, for every $i \leq k$, $c_i \neq b$. (first time D reaches a, C is in a vertex x which is different from b, and b does not appear in C before c_k.)*

$$C = a - - - - - - - - - c_k = x - - - - - - - - - - - b$$
$$D = b - - - - - - - - - d_k = a - - - - - - - - - - - a$$

- *$\exists k$, $1 < k < n$, such that for every $i < k$, $c_i \neq b$, $c_k = b$, and for every $i \leq k$, $d_i \neq a$. (first time C reaches b, D is in a vertex x which is different from a, and a does not appear in D before d_k.)*

$$C = a - - - - - - - - - c_k = b - - - - - - - - - - - b$$
$$D = b - - - - - - - - - d_k = x - - - - - - - - - - - a$$

Proof. There exist a pair of simple congruent walks $P = x_1, x_2, ..., x_n$ from a to b and $Q = y_1, y_2, ..., y_n$ from b to a such that P avoids Q. By Lemma 3.9, the vertices a or b must occur in the interior of P or Q.
If a appears in the interior of Q or b appears in the interior of P, then let $1 < i, j \leq n$ be

the minimum integers such that $y_i = a$ and $x_j = b$. Consider P and Q in place of C and D respectively. It is easy to see that if $i < j$ then the first condition holds. Otherwise the second condition holds.
If b appears in the interior of Q or a appears in the interior of P, then let $1 \leq i, j < n$ be the maximum integers such that $y_i = b$ and $x_j = a$. Consider $\overline{Q}$ and $\overline{P}$ in place of C and D respectively. By the skew symmetry property, it is easy to see that if $i > j$ then the second condition holds. Otherwise the first condition holds.

□

Lemma 3.11. *Let (a, b) be a HI-pair in a tree T. Then the vertices a and b must have the same level on T.*

Proof. Since (a, b) is an *HI-pair* in T, there exist congruent walks P from a to b and Q from b to a such that P avoid Q. Consider the walk $C = P + Q$. It is easy to see that C is a closed walk from a to a.
Suppose a and b does not have the same level on T. Hence, the net length of the walks P and Q is the same nonzero. This implies that the closed walk C is unbalanced which contradicts Proposition 2.5. □

Lemma 3.12. *Let T be a minimal SI-pair tree. There exist in T an SI-pair (l_1, l_2) such that l_1, l_2 are on the lowest level of T, and an SI-pair (h_1, h_2) such that h_1, h_2 are on the highest level of T.*

Proof. There exist an *SI-pair* (a, b) and the walks P from a to b and Q from b to a such that P avoids Q and Q avoids P. Let l_1 be any vertex with the lowest level on the walk P. Let l_2 be the corresponding vertex on Q. We assumed that P and Q avoid each other; therefore, we can say that the following sub-walks $\overline{P[a, l_1]}$, $\overline{Q[b, l_2]}$ also avoid each other. It is easy to see that $E = \overline{P[a, l_1]} + P + Q[b, l_2]$ is a walk from l_1 to l_2 and $F = \overline{Q[b, l_2]} + Q + P[a, l_1]$ is a walk from l_2 to l_1 such that E and F avoid each other. Therefore, (l_1, l_2) is a *SI-pair* such that l_1, l_2 are on the lowest level of T. Similarity, we can show that there exist also an SIP (h_1, h_2) in T such that h_1, h_2 are on the highest level of T. □

Definition 3.13. *Suppose T is a tree, and P is a path from $a \in V(T)$ to $b \in V(T)$. A vertex $i \in V(T)$ is given. The closest vertex $j \in V(P)$ to i in the underlying graph $U(T)$ is called the* ***projection*** *of i on P. We denote by $proj(i, P)$ the projection of i on P.*

Let G be a digraph, and let a, b be two vertices in G. Recall that $d(a, b)$ denotes the distance between a and b in the underlying graph $U(G)$.

Theorem 3.14. *Let T be a minimal SI-pair tree. There exist a "central" vertex $c \in V(T)$ such that the subgraph $T \setminus c$ consists three disjoint subtrees T_1, T_2 and T_3 which satisfy the following conditions:*

- *there exist $a \in V(T_1)$ and $b \in V(T_2)$ such that (a, b) is an SI-pair, and*
- *there exist $x \in V(T_3)$ such that $level(x) = level(a) = level(b)$.*

Proof. Suppose (a, b) is an *SI-pair* in T such that a, b are on the lowest level of T and $d(a, b) \geq d(i, j)$ for every SI-pair (i, j) such that i, j are on the lowest level of T. Hence, there exist a pair of minimal congruent walks $P = x_1, ..., x_n$ from a to b and $Q = y_1, ..., y_n$ from b to a such that P and Q avoid each other. Without loss of generality by Lemma 3.10 assume that P and Q satisfy the following condition: $\exists k$, $1 < k < n$, such that for every $i < k$, $x_i \neq b$, $x_k = b$, and for every $i \leq k$, $y_i \neq a$.

$$P = a - - - - - - - - - x_k = b - - - - - - - - - - - b$$
$$Q = b - - - - - - - - - y_k = x - - - - - - - - - - - a$$

There is an unique path $P_{a,b}$ from a to b in T. We shall show that $proj(x, P_{a,b})$ can not be a or b. If $proj(x, P_{a,b}) = a$, then the vertex a must appear in the interior of the walk $Q[b, x]$ which is a contradiction. If $proj(x, P_{a,b}) = b$, then let h_1 be the vertex with highest level on $P[a, b]$. Suppose the vertex h_2 is the corresponding vertex on the walk $P[b, x]$. We know that P avoids Q also Q avoids P. Hence, $P[a, h_1]$ avoids $Q[b, h_2]$, $\overline{P[a, h_1]}$ avoids $\overline{Q[b, h_2]}$. Since $level(a) = level(x)$, $level(h_1) = level(h_2)$; based on Lemma 2.8, there exist congruent walks X from a to h_1 and Y from x to h_2 such that X avoids Y and Y avoids Q. Let E and F be the walks $X + \overline{P[a, h_1]}$ and $Y + \overline{Q[b, h_2]}$ respectively. It is easy to see that $E + P + \overline{F}$ is a walk from a to x, and $F + Q + \overline{E}$ is a walk from x to a such that avoid each other. Hence, we can conclude (a, x) is an *SI-pair* such that $d(a, x) > d(a, b)$ which is a contradiction.

$$a - - - h_1 - - - a - - - h_1 - - - b - - - b - - - h_2 - - - x$$
$$x - - - h_2 - - - b - - - h_2 - - - x - - - a - - - h_1 - - - a$$

It is easy to see that $proj(x, p_{a,b})$ can be the central vertex c when $proj(x, p_{a,b}) \in V(P_{a,b}) \setminus \{a, b\}$, $x \notin V(P_{a,b})$. Finally, if $proj(x, p_{a,b}) = x$, then the vertex x must appear in the interior of $P[a, b]$. Consider the first time x appear in $P[a, b]$, y is the corresponding vertex on $Q[b, x]$. If $proj(y, p_{a,b}) \in V(p_{a,b}) \setminus \{a, b\}$, then $proj(y, p_{a,b})$ can be the central vertex c. Otherwise, if $proj(y, p_{a,b}) = a$, then a must appear in $Q[b, y]$ which is a contradiction. If $proj(y, p_{a,b}) = b$, then let h_2 be the vertex with highest level in $Q[b, y]$. Let h_1 be the corresponding vertex on $P[a, x]$. Similar to the argument presented for showing that $proj(x, P_{a,b}) = b$ can not possible , we can show that (a, y) is an *SI-pair* such that $d(a, y) > d(a, b)$ which is a contradiction.

□

Lemma 3.15. *Let T be a minimal SHI-pair tree, and let π be a min ordering for $V(T)$. If (a, b) is an SHI-pair, then $\pi(b)$ must be less than $\pi(a)$.*

Proof. There exist two congruent walks $P = x_1, ..., x_n$ from a to b and $Q = y_1, ..., y_n$ from b to a such that P avoids Q. Now suppose that $\pi(b = y_1) > \pi(a = x_1)$, then we must have $\pi(y_2) > \pi(x_2)$ because there is no edge between x_1 and y_2. By the same argument we must have $\pi(y_i) > \pi(x_i)$ for every i, $2 \leq i \leq n$. Therefore we have $\pi(a = y_n) > \pi(b = x_n)$ which is a contradiction. $\square$

3.4 General Obstructions

We can generalize the known minimal *SI-pair* trees of height 2 (see Figure 3.3), the known minimal *SHI-pair* trees of height 2 (see Figure 3.2) and the known minimal *I-pair* trees of height 2 which are constructed from two minimal *SHI-pair* trees (see Figure 3.4). The trees H_1, H_2, H_3, H_4 as shown in Figure 3.5 are some minimal *SI-pair* trees, the trees $H_1^{'}, H_2^{'}$ as shown in Figure 3.6 are some minimal *SHI-pair* trees and the trees $H_1^{''}, H_2^{''}, H_3^{''}, H_4^{''}, H_5^{''}$ as shown in Figure 3.7 are some minimal *I-pair* trees which are not *SI-pair* trees and they are constructed from two minimal *SHI-pair* trees.

3.5 Conclusions

For reflexive trees T, it is known that there exits a concrete forbidden induced subgraph characterization to have a min ordering. On the other hand, for irreflexive trees T, the existence of a min-ordering turned out to be quite harder, as there are many types of obstructions to its existence. For trees of height one and smooth trees of height less than three T, we showed a concrete forbidden induced subgraph characterization to have a min ordering and an *I-pair* and a DAT. Moreover, we showed that for this case, the tree T has a DAT if and only if there is an *I-pair* in T. Furthermore, we proved that if T contains one of the forbidden obstructions, then the problem $LHOM(T)$ is NP-complete. Otherwise, it is polynomial time solvable.
We also introduced some new forbidden structures by relaxing and tightening the conditions of the *I-pair*'s which turned out to be useful for finding minimal irreflexive trees which admit a min ordering. Furthermore, we introduced a new type of drawing a tree on levels (parallel lines), which we called it a layout. The layout yields an easy recognition of trees which admit a min ordering. In fact, we showed that a tree has a layout if and only if it admits a min ordering. The concept of layout is a very useful tool for characterizing the minimal obstructions for general trees. For future works, we hope to use these new forbidden structures and the concept of layouts to extend our results for irreflexive trees of the arbitrary height.

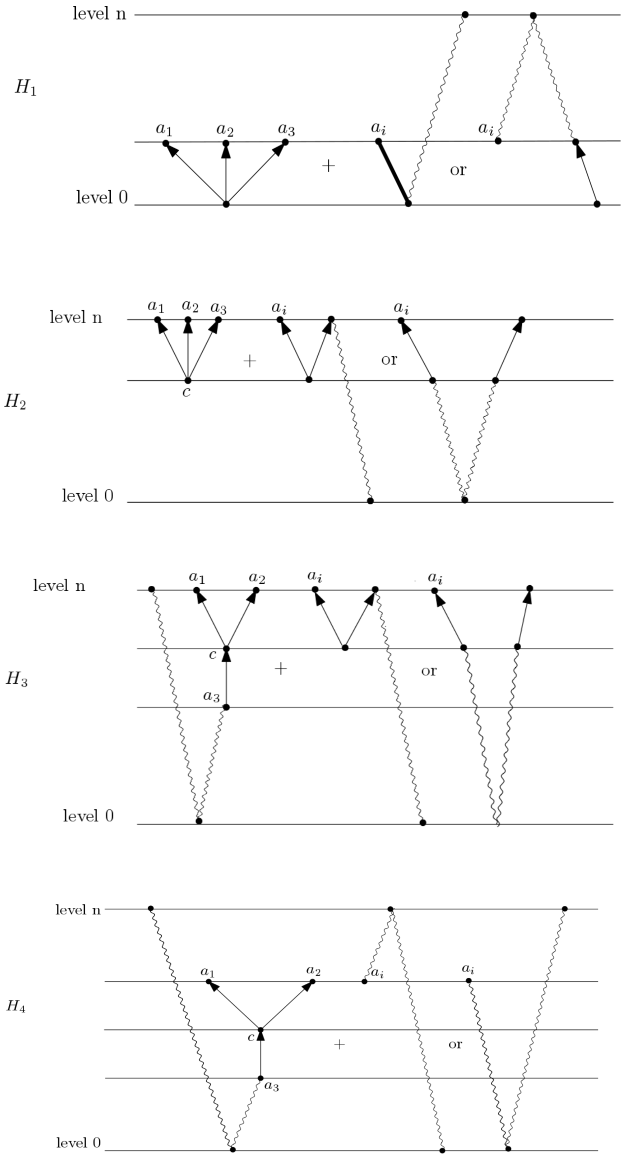


Figure 3.5: The known minimal *SI-pair* trees. The thick line can be an edge or a path of overall height one. The wiggly line can be any path between its two end points.

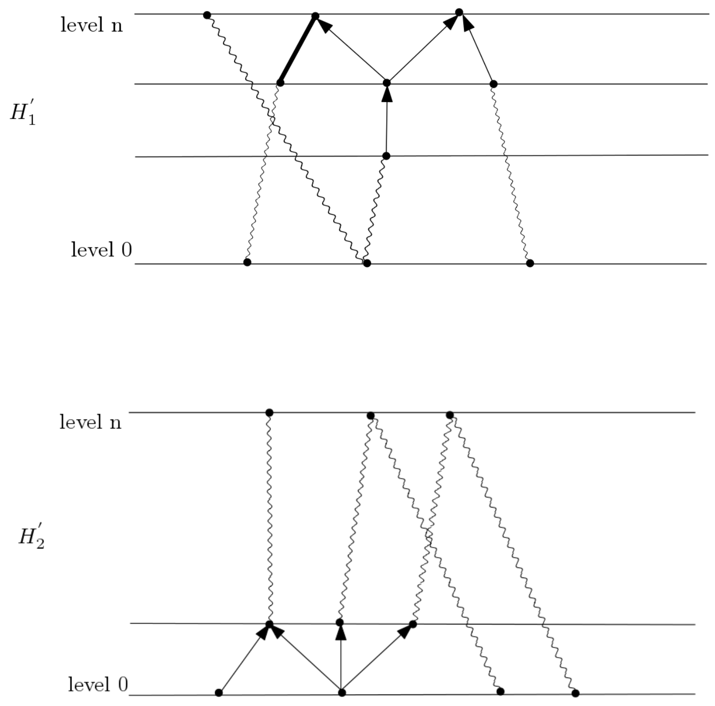


Figure 3.6: The known minimal *SHI-pair* trees. The thick line can be an edge or a path of overall height one. The wiggly line can be any path between its two end points.

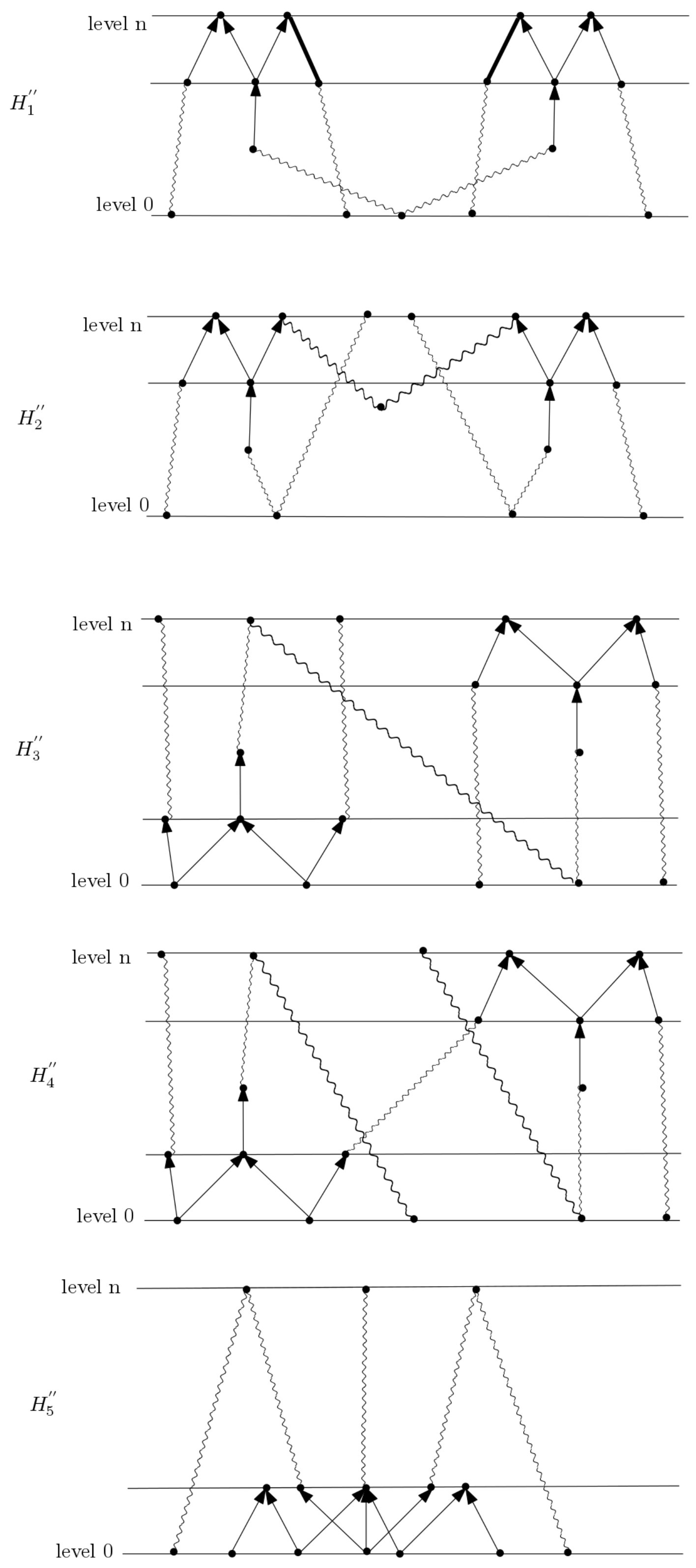


Figure 3.7: The known minimal *I-pair* trees which are not *SI-pair* trees, and contain two *SHI-pair* trees as induced subtrees. The thick line can be an edge or a path of overall height one. The wiggly line can be any path between its two end points.

Bibliography

[1] Andrei A Bulatov. Tractable conservative constraint satisfaction problems. In *Logic in Computer Science, 2003. Proceedings. 18th Annual IEEE Symposium on*, pages 321–330. IEEE, 2003.

[2] Andrei A Bulatov. A dichotomy theorem for nonuniform csps. *arXiv preprint arXiv:1703.03021*, 2017.

[3] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[4] Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236–250, 1998.

[5] Tomas Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.

[6] Tomas Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003.

[7] Tomás Feder, Pavol Hell, Jing Huang, and Arash Rafiey. Adjusted interval digraphs. *Electronic Notes in Discrete Mathematics*, 32:83–91, 2009.

[8] Tomás Feder and Moshe Y Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[9] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[10] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

[11] Gregory Gutin, Arash Rafiey, and Anders Yeo. Minimum cost and list homomorphisms to semicomplete digraphs. *Discrete Applied Mathematics*, 154(6):890–897, 2006.

[12] Pavol Hell and Jaroslav Nesetril. *Graphs and homomorphisms.* Oxford University Press, 2004.

[13] Pavol Hell and Jaroslav Nešetřil. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

[14] Pavol Hell and Mayssam Mohammadi Nevisi. Minimum cost homomorphisms with constrained costs. In *International Computing and Combinatorics Conference*, pages 194–206. Springer, 2016.

[15] Pavol Hell and Arash Rafiey. The dichotomy of list homomorphisms for digraphs. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1703–1713, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics.

[16] Pavol Hell and Arash Rafiey. The dichotomy of minimum cost homomorphism problems for digraphs. *SIAM Journal on Discrete Mathematics*, 26(4):1597–1608, 2012.

[17] Pavol Hell and Arash Rafiey. Monotone proper interval digraphs and min-max orderings. *SIAM Journal on Discrete Mathematics*, 26(4):1576–1596, 2012.

[18] Pavol Hell and Arash Rafiey. Bi-arc digraphs and conservative polymorphisms. *arXiv preprint arXiv:1608.03368*, 2016.

[19] Donald L Kreher and Douglas R Stinson. *Combinatorial algorithms: generation, enumeration, and search*, volume 7. CRC press, 1998.

[20] Richard E Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171, 1975.

[21] C Lekkeikerker and J Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

[22] Leonid Levin. Universal'nyîe perebornyîe zadachi (universal search problems, in russian). *Problemy Peredachi Informatsii*, 9:265–266, 1973.

[23] Hermann A Maurer, JH Sudborough, and Emo Welzl. On the complexity of the general coloring problem. *Information and control*, 51(2):128–145, 1981.

[24] Arash Rafiey, Jeff Kinne, and Tomás Feder. Dichotomy for digraph homomorphism problems. *arXiv preprint arXiv:1701.02409*, 2017.

[25] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[26] Dmitriy Zhuk. The proof of csp dichotomy conjecture. *arXiv preprint arXiv:1704.01914*, 2017.