Karlsruhe Institute of Technology

Weighted Disk Contact Graphs

Diplomarbeit von

Boris Klemz

An der Fakultät für Informatik
Institut für theoretische Informatik
Lehrstuhl Algorithmik I

Gutachter:	Dr. Martin Nöllenburg
Zweitgutachter:	Prof. Dr. Peter Sanders
Betreuende Mitarbeiter:	Dr. Martin Nöllenburg
	Dipl.-Inform. Roman Prutkin

Bearbeitungszeit: 1. Mai 2014 – 31. Oktober 2014

Acknowledgements

I want to thank my advisors Dr. Martin Nöllenburg and Dipl.-Inform. Roman Prutkin for the many hours of interesting discussions and great support in the form of helpful advises and feedback during the writing process.

I declare that I have developed and written the enclosed thesis by myself, and have not used sources or means without declaration in the text.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 31. Oktober 2014

. . . . . .

(**Boris Klemz**)

Abstract

Disk touching graphs realize graphs by representing each vertex as a disk in the plane such that disks touch each other if and only if the corresponding vertices are adjacent. Application areas for disk graphs include modeling physical problems and visualizing statistical data. Deciding whether a graph can be realized such that the disks' radii coincide with some predefined values or such that the disks cover some specified points in the plane has been proven to be $\mathcal{NP}$-hard [BK98, AdCC$^+$12].

In this thesis we take a close look at several variations of these two scenarios and analyze what can be guaranteed for special graph classes. We show that many of the considered problems remain $\mathcal{NP}$-hard even for very basic graph classes. We thereby strengthen several previous $\mathcal{NP}$-hardness results and provide $\mathcal{NP}$-hardness proofs for new problem variations. In particular, we show that it is $\mathcal{NP}$-hard to decide whether a realization with uniform radii exists even if the input graph is outerplanar and even if a combinatorial embedding is provided. If not necessarily uniform radii are assigned, the problem becomes $\mathcal{NP}$-hard even for stars (but can be solved in linear time in a Real RAM model if a combinatorial embedding is provided). If to-be-covered points are assigned, the recognition problem remains $\mathcal{NP}$-hard even for trees and if cover points as well as (unit) radii are assigned, the problem becomes $\mathcal{NP}$-hard, even for paths and, therefore, even if a combinatorial embedding is provided. On the other hand, we present some linear-time algorithms for graph class/problem combinations for which we do not prove $\mathcal{NP}$-hardness. For example, we show that deciding whether a caterpillar can be realized as a disk touching graph with uniform radii can be decided in linear time.

Deutsche Zusammenfassung

In einer Kreis-Kontaktdarstellung wird ein Graph realisiert, indem jeder Knoten als ein Kreis dargestellt wird und sich zwei der Kreise genau dann berühren, wenn die entsprechenden Knoten benachbart sind. Derartige Darstellungen finden unter anderem Anwendung beim Modellieren physikalischer Prozesse und in der Datenvisualisierung. Es ist bekannt [BK98, AdCC$^+$12], dass wenn im Vorfeld die Kreisradien spezifiziert oder bestimmte Punkte festlegt werden, die von den Kreise abgedeckt werden sollen, es $\mathcal{NP}$-schwer zu entscheiden ist, ob ein eine Realisierung existiert, die diesen Spezifikationen genügt.

In dieser Diplomarbeit untersuchen wir diverse Variationen dieser zwei Szenarien und zeigen, dass viele der entsprechenden Entscheidungsprobleme selbst dann noch $\mathcal{NP}$-schwer sind, wenn die Eingabe auf sehr einfache Graphklassen eingeschränkt wird. Hierdurch bekräftigen wir einige bisherige Resultate, die sich auf allgemeinere Graphklassen beziehen und zeigen $\mathcal{NP}$-schwere für neue Problemvarianten. Speziell zeigen wir, dass wenn ein Einheitsradius gefordert wird, das Entscheidungsproblem selbst für außenplanare Graphen und selbst wenn eine kombinatorische Einbettung gestellt wird $\mathcal{NP}$-schwer bleibt. Für nicht zwangsläufigerweise einheitliche Radienzuweisungen ist das Problem selbst für Sterne $\mathcal{NP}$-schwer (kann jedoch in einem Real RAM Modell in Linearzeit gelöst werden, wenn eine kombinatorische Einbettung zur Verfügung steht). Sind zu überdeckende Punkte spezifiziert, so ist das Problem selbst für Bäume noch $\mathcal{NP}$-schwer und wenn zusätzlich (Einheits-) Radien spezifiziert werden, ist es selbst für Pfade (und damit selbst dann, wenn eine Einbettung gestellt wird) noch $\mathcal{NP}$-schwer zu entscheiden, ob eine Realisierung existiert. Für einige Graphklassen/Problem Kombinationen, für die nicht $\mathcal{NP}$-schwere gezeigt wurde, werden außerdem Linearzeit-Algorithmen vorgestellt, zum Beispiel kann in Linearzeit entschieden werden, ob für einen Caterpillar eine Realisierung mit Einheitsradius existiert.

Contents

1. Introduction

A disk intersection graph consists of a set of disks in the plane and it can be interpreted as a graph that contains a vertex for each of its disks and and an edge for each pair of intersecting disks. Disk intersection graphs generalize disk touching graphs, in which any two disks that intersect each other do so in exactly one point. Koebe's Theorem [Koe36] is a classic result in graph theory, that states that any planar graph can be represented by a disk touching graph. On the other hand, for any disk touching graph we can obtain a planar drawing of the realized graph by connecting the centers of each pair of touching disks with a straight-line segment. The set of planar graphs therefore coincides with the set of graphs that can be represented by disk touching graphs, which, therefore, can be recognized in linear time [HT74].

Application areas for disk intersection/touching graphs include modeling physical problems like wireless communication networks [Hal80], covering problems like geometric facility location [RT90, Wel91], visual representation problems like the generation of area cartograms [Dor96] and many more (various examples are given by Clark et al. [CCJ90]). In this context, one is often interested in recognizing or generating disk graphs that do not only realize the input graph, but also satisfy additional requirements. It might be desirable to generate a disk graph the realizes a vertex-weighted graph such that the disks' radii or areas reflect the corresponding vertices' weights. For example, Figure 1.1 depicts a disk touching graph that acts as an area cartogram that visualizes the relative sizes of the populations of Germany and its neighboring countries. Clearly, there exist vertex-weighted planar graphs that can not be realized as disk touching graphs and the corresponding recognition problem is $\mathcal{NP}$-hard even if all vertices are weighted uniformly [BK98]. In another scenario, one could designate a point in the plane to each vertex and be interested in recognizing or generating a disk touching graph such that each vertex's corresponding disk covers the vertex's assigned point. The corresponding decision problem is also $\mathcal{NP}$-hard [AdCC$^+$12].

In this thesis, we examine the two aforementioned scenarios, as well as variations and combinations of them, more closely and explore what can be guaranteed for special graph classes. We strengthen several known $\mathcal{NP}$-hardness results by showing that they hold true even for very basic graph classes like paths, stars and trees and prove $\mathcal{NP}$-hardness for new problem variations. For some of the variations for which we do not prove $\mathcal{NP}$-hardness, we present linear-time construction algorithms or existence statements. A detailed contribution and section overview is presented in Section 1.3. Prior, we introduce some basic

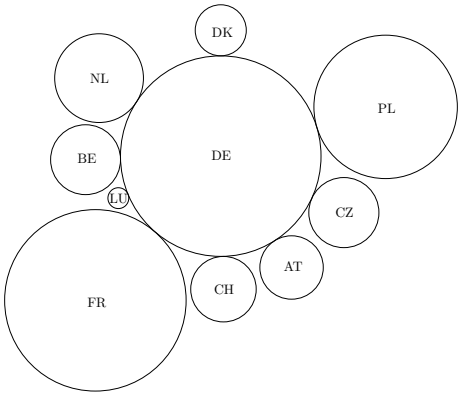


Figure 1.1.: A disk touching graph that realizes a vertex-weighted star and that acts as an area cartogram which visualizes the relative sizes of the populations of Germany and its neighboring countries.

concepts and provide formal problem definitions in Section 1.1 and review some related work in Section 1.2.

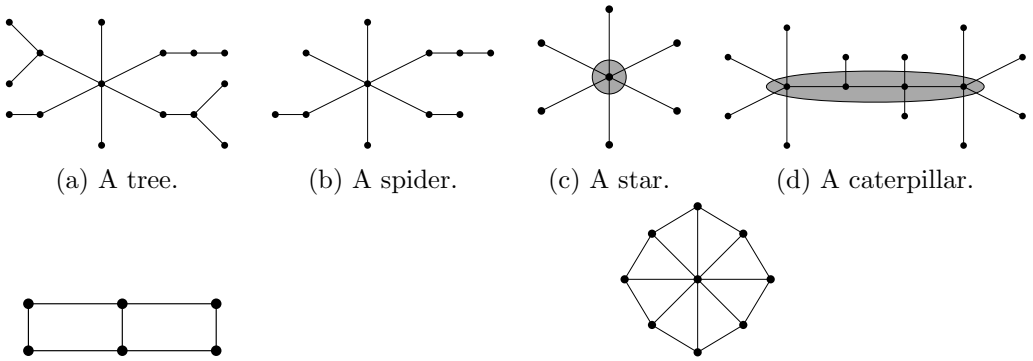
1.1. Preliminaries

In this section we recall some basic concepts and introduce elementary definitions that will be used throughout this thesis. Section 1.1.1 focuses on graph-theoretic concepts. In Section 1.1.2 we specify the $\mathcal{NP}$-hard problems that are utilized in the polynomial-time reductions in this thesis. We also specify a Real RAM model used for some of our positive results. Finally, in Section 1.1.3 we formally introduce disk graphs and the related recognition problems.

1.1.1. Graph theory

A *graph* G is an ordered pair (V, E), where V is a set of *vertices* and E is a set of *edges*, which are subsets of V with cardinality 2. If there exists an edge $\{u, v\} \in E$ for every pair of distinct vertices $v, u \in V$, then G is called *complete* graph. A vertex $v \in V$ and an edge $e \in E$ are said to be *incident* if $v \in e$. In general, in a (*simple*) graph there exists at most one edge per vertex pair. A *multigraph*, however, is a graph in which we explicitly allow the existence of multiple edges incident to the same pair of vertices. The *degree* $\deg(v) = |\{e \in E \mid v \in e\}|$ of a vertex $v \in V$ is the number of edges incident to v. Two vertices $u, v \in V$ are *neighbors* (or *adjacent*) in G if there exists an edge that is incident to both u and v. A *subgraph* of G is a graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$. The graph G' is said to be *induced* by vertex set V' if $E' = \{\{v, u\} \in E \mid v, u \in V'\}$ and it is said to be *induced* by edge set E' if $V' = \{v \in V \mid \exists e' \in E' : v \in e'\}$. In general graphs edges are unordered vertex pairs. In a *directed* (multi-) graph $\overrightarrow{G} = (V, \overrightarrow{E})$, however, each edge is an ordered pair from V^2. We say that an edge $(u, v) \in \overrightarrow{E}$ is *directed* from u to v. The *outdegree* of a vertex $v \in V$ is $|\{(v_1, v_2) \in \overrightarrow{E} \mid v_1 = v\}|$ and its *indegree* is $|\{(v_1, v_2) \in \overrightarrow{E} \mid v_2 = v\}|$.

Let $G = (V, E)$ be a graph and $p = (v_1, \ldots, v_k) \in V^k$ be a sequence of vertices. If a corresponding sequence of edges $p = (e_1, \ldots, e_{k-1}) \in E^{k-1}$ with $e_i = \{v_i, v_{i+1}\}$ for $1 \leq i \leq k-1$ exists, we refer to p as a *path* in G between v_1 and v_k. We also refer to the graph $P = (\{v_1, \ldots, v_k\}, \{e_1, \ldots, e_{k-1}\})$ as a path. In this thesis, we exclusively consider *simple* paths (and therefore omit this term), meaning that the vertices $v_1, \ldots, v_k$ except for maybe v_1 and v_k are pairwise distinct. If $v_1 = v_k$, we refer to the path as a *cycle*.



(a) A tree. (b) A spider. (c) A star. (d) A caterpillar.

(e) An outerplanar graph. (f) Internally triangulated drawing of a 2-outerplanar graph.

Figure 1.2.: Planar drawings visualizing the different graph classes and properties.

Two vertices of V are said to be *connected* in G if there exists a path with edges of E between them. A *connected component* $C = (V_C, E_C)$ of the graph G is a subgraph of G such that the vertices in V_C are pairwise connected and there exists no path between a vertex of V_C and a vertex of $V \setminus V_C$. A graph is *connected* if it has a single connected component. Let $s, t \in V$ be two non-adjacent vertices. The vertices s and t are said to be *k-connected* if $k \in \mathbb{N}_0$ is the smallest number of vertices that needs to be removed from V such that in the induced subgraph, s and t are no longer connected. Menger's Theorem [Men27] states that in this case, there exist k *internally vertex-disjoint* paths between s and t, which means that there exist k paths that connect s and t but that do not share any common vertices except for s and t. Two adjacent vertices from V are said to be $(|V| - 1)$-*connected*. The graph G is said to be *k-connected* if any pair of distinct vertices is at least k-connected.

A connected graph whose edge set does not contain a subset that induces a cycle is called a *tree*. In a tree vertices with degree 1 are called *leafs*. A *spider* is a tree that contains at most one vertex with a degree 3 or higher. A *star* is a spider that contains at most one non-leaf vertex. If such a vertex exists, we refer to it as the star's *central* vertex. A *caterpillar* is a tree $C = (V, E)$ such that there exists a path $p = (v_1, \ldots, v_k) \in V^k$ and any vertex in $V \setminus \{v_1, \ldots, v_k\}$ is a leaf. Such a path p is called *inner* path of C. Figure 1.2 visualizes several examples as planar drawings, which are defined in the following paragraph. The star's central vertex and the caterpillar's inner path are highlighted in gray in Figure 1.2.

A *curve* in the plane is a continuous function $c : [a, b] \to \mathbb{R}^2$ that maps elements of a real interval to points in the plane. The points $c(a)$ and $c(b)$ are called *endpoints* of c. The *image* of c is the set of points $\text{image}(c) = \{p \in \mathbb{R}^2 \mid \exists d \in [a, b] : c(d) = p\}$. Let $d, e \in (a, b)$ be real numbers. The curve c is called *simple* if $c(d) = c(e)$ implies that $d = e$. A *drawing* of graph $G = (V, E)$ can be obtained by mapping the vertices of V to distinct points in the plane and representing the edges of E with simple curves in the plane such that the endpoints of the curve associated with an edge $\{v, u\} \in E$ are the points associated with u and v. A drawing of G is called *planar* if the images of the curves associated with the edges of E only intersect at their respective endpoints and therefore only at points associated with vertices. The graph G is called *planar* if there exists a planar drawing of G. A planar drawing of G uniquely defines a cyclic order of edges incident to a vertex v for each vertex $v \in V$. The set of all these cyclic orders for a planar drawing is called its *combinatorial embedding* Γ (for G). Note that there exists an infinite amount of planar drawings for G whose combinatorial embedding is also Γ. The regions bounded by the images of the curves in a planar drawing of a graph are called *faces*. The outer, infinitely large face is called *outer* face and all other faces are called *inner* faces. Let f be a face of a planar drawing of G, let E_f be the set of edges that correspond to the curves whose images

bound f in the drawing and let $G_f = (V_f, E_f)$ be the subgraph of G induced by E_f. The vertices in V_f are said to be *adjacent* to f. If all inner faces of the drawing are adjacent to exactly three vertices, the drawing is called *internally triangulated*. If there exists a planar drawing of the graph $G = (V, E)$ such that all vertices in V are adjacent to the outer face of the drawing, then both the graph G and its drawing are called *outerplanar* or 1-*outerplanar*. It is known that in each outerplanar graph there exists at least one vertex with degree at most 2 [LW70]. The graph G is called k-*outerplanar*, with $k \geq 2$, if there exists a planar drawing of G such that if all vertices adjacent to the outer face of the drawing are removed from V, all of the connected components of the subgraph of G induced by the remaining vertices are $(k-1)$-outerplanar. Figure 1.2 depicts some planar drawings that visualize these properties.

1.1.2. $\mathcal{NP}$-hard problems and the Real RAM model

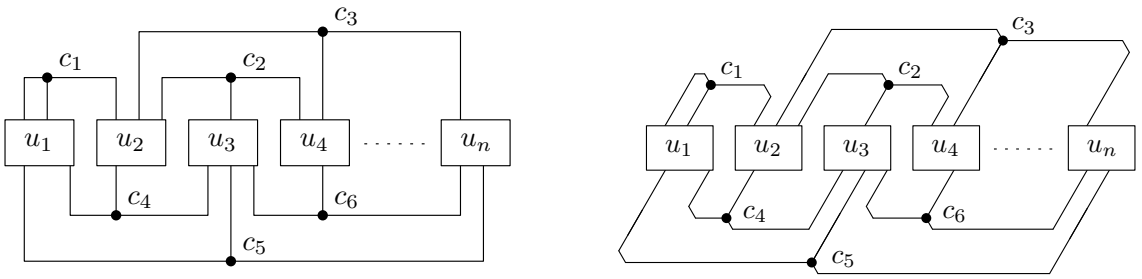
For the purpose of this thesis, we assume that the reader is familiar with the theory of $\mathcal{NP}$-hardness. We refer to [CLRS09, Chapter 34] for an excellent introduction to this topic and to [GJ79] for an in-depth view. The latter reference also inspired most of the definitions for the recognition problems presented in this section. These logic and partitioning problems are used for the polynomial-time reductions in this thesis.

Let U be a set of *variables* and let $u \in U$ be a variable. We say that u is *true* or *false* with respect to *truth assignment* $t : U \to \{true, false\}$ for U if $t(u) = true$ or $t(u) = false$ respectively. For variable $u \in U$ we say that u and $\overline{u}$ are *literals* over U and that u ($\overline{u}$) is the *positive* (*negative*) literal of variable u. Literal u is *true* (*false*) with respect to t if and only if variable u is true (false) and literal $\overline{u}$ is *true* (*false*) with respect to t if and only if variable u is false (true). A *clause* over U is a sequence of (not necessarily pairwise distinct) literals over U. Intuitively, a clause represents the disjunction of its literals and a collection of clauses represents the conjunction of its clauses. For example, the clause $(u_1, \overline{u}_1, \overline{u}_2, u_4)$ represents Boolean formula $(u_1 \vee \overline{u}_1 \vee \overline{u}_2 \vee u_4)$. We therefore say that a clause over U is *satisfied* by t if and only if at least one of its literals is true with respect to t; we say that a collection of clauses over U is *satisfied* by t if and only if all of its clauses are satisfied by t; we say that a collection of clauses C over U is *satisfiable* if there exists a truth assignment for U which satisfies C. With these definitions, we are now prepared to define the (Boolean) Satisfiability problem as well as three variants of it:

Satisfiability (SAT): A SAT instance consists of a set U of variables and a collection C of clauses over U and the question is whether C satisfiable.

- **3-Satisfiability (3SAT):** The cardinality of any clause $c \in C$ is $|c| = 3$.
- **($\leq$,3,3)-Satisfiability ($\leq$33SAT):** The cardinality of any clause $c \in C$ is $|c| \leq 3$. For any $u \in U$ there exist at most three clauses in C that contain either u or $\overline{u}$.
- **Planar 3-Satisfiability (P3SAT):** The cardinality of any clause $c \in C$ is $|c| = 3$. There exists a planar (multi-) graph $H = (V_U \cup V_C, E)$ where V_U contains a *variable* vertex for each variable of U, V_C contains a *clause* vertex for each clause of C and for each clause of C, E contains three edges connecting the corresponding clause vertex and the corresponding variable vertices.

SAT and all three variants are $\mathcal{NP}$-hard [Coo71, GJ79, Lic82]. The (multi-) graph H in the P3SAT definition can be drawn on a rectangular grid of polynomial size such that all variable vertices are placed on a horizontal line and the clause vertices are connected in a comb-shaped manner from above and below that line [KR92], see Figure 1.3a. This drawing can furthermore be slanted such that all angles are multiples of 60 degrees [CDR07], see Figure 1.3b. These Satisfiability problems are used for the reductions in Chapter 2 and 4. In Chapter 3 we utilize the following partition problem:



(a) Rectangular drawing of H.

(b) Slanted drawing of H.

Figure 1.3.: Planar 3-Satisfiability instances can be drawn on grids of polynomial size.

3-Partition (3P): A 3P instance consists of a set $A \subset \mathbb{N}$ of positive integers with $|A| = 3n$ with $n \in \mathbb{N}$ and a bound $B \in \mathbb{N}$ such that $B/4 < a < B/2, \forall a \in A$ and such that $\sum_{a \in A} a = nB$. The question is whether it is possible to partition A into n disjoint sets $A_1, \ldots, A_n$ with $\bigcup_{1 \leq i \leq n} A_i = A$ such that $\sum_{a \in A_i} a = B$ for any $1 \leq i \leq n$.

The 3-Partition problem is $\mathcal{NP}$-complete in the *strong* sense [GJ75], which means that it remains $\mathcal{NP}$-complete even if its numerical parameters (the bound B and the integers in A) are bounded by a polynomial in the size of the input (the number of integers $|A| = 3n$). As noted by the authors, the bounds of $B/4$ and $B/2$ for the integer values imply that $|A_i| = 3$ for every $1 \leq i \leq n$ [GJ79]. Observe that by multiplying all integers as well as the bound by some natural number, we obtain a problem instance that is a yes-instance if and only if the original problem instance is a yes-instance. We can and will therefore without loss of generality assume that $B > 12$ and that $B = 0 \mod 4$ in any problem instance considered in this thesis. We call an integer triple from A *feasible* if the sum of its integers is B and *infeasible* if the sum is larger than B.

Dealing with geometric objects like line segments or circles often gives rise to complicated $\mathbb{R}^2$-coordinates and the precise computation of such coordinates can technically take an infinite amount of time. In a Real RAM (random access machine) model, however, one assumes that a designated set of arithmetic operations (in our case $+$, $-$, $\cdot$, $/$, square roots, sin, cos, arcsin) can be performed in constant time [PS85]. This model is a tool for nevertheless providing positive results. This can be useful since, in practice, depending on the application, heuristical approaches might suffice to calculate the coordinates. Several of the constructive algorithms presented in this thesis are shown to have linear runtime in this Real RAM model. Note, however, that we can not use this model in any of the reductions used to prove $\mathcal{NP}$-hardness. Instead we have to argue why the required coordinates can be computed in polynomial time.

1.1.3. Disk graphs and problem definitions

A *disk* D is a region in the plane bounded by a circle. A disk can be uniquely described by its bounding circle's radius $r \in \mathbb{R}^+$ and center $c \in \mathbb{R}^2$. A disk is called *closed* if it contains the points of its bounding circle and *open* otherwise. A *disk intersection graph* $\mathcal{G} = (G, \mathcal{V})$ consists of a graph $G = (V, E)$, a set of closed disks $\mathcal{V}$ and a bijection from the set of vertices V to $\mathcal{V}$ such that two vertices of V are adjacent in G if and only if their corresponding disks in $\mathcal{V}$ intersect. A *disk touching graph* $\mathcal{G} = (G, \mathcal{V})$ is a disk intersection graph such that the interiors of the disks in $\mathcal{V}$ are pairwise disjoint. The centers of the disks in $\mathcal{V}$ together with straight-line segments that connect the centers of all pairs of touching disks *induce* a planar drawing of G. In a *unit* disk intersection (touching) graph all disks share one uniform radius. In a *ρ-bounded* disk intersection (touching) graph the radius of all disks is taken from the interval $[1, \rho]$ for a value $\rho \geq 1$. Note, that a 1-bounded disk intersection (touching) graph is a unit disk intersection (touching) graph.

For the sake of simplicity, throughout this thesis we often refer to disks and their corresponding vertices synonymously. For example, we might simply say 'we create a disk D_2 with radius r_2 that touches disk D_1' instead of saying 'we create a vertex v_2, a corresponding disk D_2 with radius r_2 and an edge between v_2 and vertex v_1 whose corresponding disk is D_1'.

Let $G = (V, E)$ be a graph. We say that G has a *realization* as a disk intersection (touching) graph, if there exist a set of disks $\mathcal{V}$ and a bijection from V to $\mathcal{V}$ such that $\mathcal{G} = (G, \mathcal{V})$ is a disk intersection (touching) graph. In this case, we say $\mathcal{G}$ *realizes* G. Let $D_v \in \mathcal{V}$ be the disk of $\mathcal{G}$ corresponding to vertex v for any $v \in V$. A *radius assignment* for G is a function $r : V \to \mathbb{R}^+$ that assigns a positive real number to each vertex of G. If the radius of disk $D_v \in \mathcal{V}$ is equal to $r(v)$ for every $v \in V$, then $\mathcal{G}$ is said to *respect* r. A *seed assignment* for G is a function $\sigma : V \to \mathbb{R}^2$ that assigns a point in the plane to each vertex of G. If $\sigma(v) \in D_v$ for every $v \in V$, then $\mathcal{G}$ is said to *respect* σ. Let Γ be a combinatorial embedding for G. If $\mathcal{G}$ is a disk touching graph and if the cyclic order of disks touched by D_v corresponds to the cyclic order of edges incident to the vertex v for any $v \in V$, then $\mathcal{G}$ is said to *respect* Γ.

We consider the following family of decision problems, in which the dots (...) are a placeholder for one, multiple or none of the enlisted variants.

(Unit/ρ-bounded) Disk Intersection/Touching Graph Recognition (with ...): The problem instance is a graph $G = (V, E)$ and the question is whether it is possible to realize G as a (unit/ρ-bounded) disk intersection/touching graph (which respects ...).

- **... fixed Radii:** ... a given radius assignment r for G.
- **... fixed Embedding:** ... a given combinatorial embedding Γ for G.
- **... fixed Seeds:** ... a given seed assignment σ for G.

In particular, we consider the following problems:

- Unit Disk Touching Graph Recognition (UDT)
- Unit Disk Touching Graph Recognition with fixed Embedding (UDTE)
- ρ-bounded Disk Touching Graph Recognition (ρ-BDT)
- Disk Touching Graph Recognition with fixed Radii (DTR)
- Disk Touching Graph Recognition with fixed Radii and Embedding (DTRE)
- Disk Touching Graph Recognition with fixed Seeds (DTS)
- Unit Disk Touching Graph Recognition with fixed Seeds (UDTS)
- Unit Disk Touching Graph Recognition with fixed Seeds and Embedding (UDTSE)

1.2. Related work

As mentioned in the beginning of this chapter, Koebe's Theorem [Koe36] implies that the Disk Touching Graph Recognition problem can be solved in linear time. On the other hand, Hliněný and Kratochvíl showed that the Disk Intersection Graph Recognition problem is $\mathcal{NP}$-hard [HK01].

A result by Breu and Kirkpatrick states that the Unit Disk Intersection/Touching Graph Recognition problems are $\mathcal{NP}$-hard [BK98], implying that the Disk Intersection/Touching Graph Recognition with fixed Radii problems are also $\mathcal{NP}$-hard. There exists some heuristics for generating disk touching graphs with fixed radii [Dor96, Ino11] for the application of cartogram generation.

Breu and Kirkpatrick generalized their results by showing that the ρ-bounded Disk Intersection/Touching Graph Recognition problems are $\mathcal{NP}$-hard for any fixed $\rho \geq 1$ [BK96]. Alam et al. [AEG+14] argue that for any tree, for any cactus (which is a connected graph in which each edge is contained in at most one cycle), for any k-outerplanar graph with bounded maximum degree and $k \in O(\log n)$ and for any planar graph with bounded tree-depth there exists a realizing ρ-bounded disk touching graph where ρ is a polynomial in the number of vertices.

Atienza et al. show that the Disk Touching Graph Recognition with fixed Seeds problem is $\mathcal{NP}$-hard [AdCC+12].

1.3. Contribution and section overview

In Chapter 2 we investigate recognition problems with uniform (and ρ-bounded) radii. In Section 2.1 we consider the Unit Disk Touching Graph Recognition (UDT) problem and strengthen the result of Breu and Kirkpatrick [BK98] by showing that the UDT problem is $\mathcal{NP}$-hard even for outerplanar graphs. On the positive side, we provide a linear-time algorithm for deciding the UDT problem in caterpillars. In this section we also briefly consider ρ-bounded Disk Touching Graph Recognition and show that for spiders this problem can be solved in linear time in the Real RAM model. In Section 2.2 we extend our result from the previous section by showing that the Unit Disk Touching Graph Recognition with fixed Embedding (UDTE) problem is also $\mathcal{NP}$-hard, even for outerplanar graphs.

In Chapter 3 we explore the more general scenario with fixed but not necessarily uniform radii. In Section 3.1 we consider the Disk Touching Graph Recognition with fixed Radii (DTR) problem and strengthen the result by Breu and Kirkpatrick [BK98] by showing that the DTR problem is $\mathcal{NP}$-hard even for stars. We also show that for any cycle and a corresponding radius assignment there exists a realizing disk touching graph. In contrast, in Section 3.2 we devise a linear-time algorithm for deciding the Disk Touching Graph Recognition with fixed Radii and Embedding (DTRE) problem for stars in the Real RAM model.

In Chapter 4 we concern ourself with the scenario in which a seed assignment is required to be respected. In Section 4.1 we strengthen the result of Atienza et al. [AdCC+12] by showing that the Disk Touching Graph Recognition with fixed Seeds (DTS) problem is $\mathcal{NP}$-hard even for trees. In Section 4.2 we combine this scenario with assigning fixed radii, more specifically uniform radii. We show that the Unit Disk Touching Graph Recognition with fixed Seeds (UDTS) problem is $\mathcal{NP}$-hard, even for paths, implying that the Unit Disk Touching Graph Recognition with fixed Seeds and Embedding (UDTSE) problem is also $\mathcal{NP}$-hard even for paths.

We conclude our work in Chapter 5.

2. Recognition problems with unit radii

In this chapter we consider the Unit Disk Touching Graph Recognition (with fixed Embedding) (UDT and UDTE) problem and, briefly, the ρ-bounded Disk Touching Graph Recognition (ρ-BDT) problem. Recall that in these problem we need to decide whether a given graph can be realized as a unit/ρ-bounded disk touching graph (that respects a given combinatorial embedding). Furthermore, recall that both UDT and ρ-BDT for any $\rho \geq 1$ are $\mathcal{NP}$-hard [BK98, BK96]. In Section 2.1 we strengthen the $\mathcal{NP}$-hardness result by Breu and Kirkpatrick by showing that UDT remains $\mathcal{NP}$-hard even for outerplanar graphs. We additionally show that for caterpillars the problem can be decided in linear time and that in the Real RAM model the ρ-BDT can be solved in linear time for caterpillars for any $\rho \geq 1$. In Section 2.2 we extend our $\mathcal{NP}$-hardness result for UDT and show that it holds true even if a combinatorial embedding is specified and, therefore, that the UDTE is $\mathcal{NP}$-hard, even for outerplanar graphs. For both problems we also provide existence arguments for realizations of paths and cycles and simple decision approaches for stars and spiders.

2.1. Unit Disk Touching Graph Recognition

In this section we consider the Unit Disk Touching Graph Recognition (UDT) problem and show that it is $\mathcal{NP}$-hard even for outerplanar graphs. Prior, however, we provide simple observations concerning UDT/ρ-BDT for paths, cycles, stars and spiders and proceed by presenting a linear-time algorithm for deciding the UDT problem for caterpillars.

Observation 1. *Any path can be realized as a unit disk touching graph.*

Observation 2. *Any cycle can be realized as a unit disk touching graph.*

Observation 3. *Let $\mathcal{G} = (G, \mathcal{V})$ be a unit disk touching graph that realizes $G = (V, E)$. There exists no vertex $v \in V$ with $\deg(v) > 6$ and if there exists a vertex $v' \in V$ with $\deg(v') = 6$, then the disks in $\mathcal{V}$ corresponding to the neighbours of v' touch each other consecutively.*

Corollary 1. *For spiders the Unit Disk Touching Graph Recognition problem can be decided in $O(n)$ time where n is the number of vertices of the given spider. In the Real RAM model, a realization can be constructed in $O(n)$ time (if one exists).*

Observation 1 is trivial. Observation 2 follows from the fact that in order to realize a cycle with n vertices, one can simply center n disks with unit radius r_u at the n corners of a

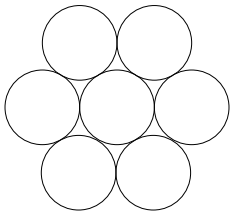


Figure 2.1.: An optimal packing of seven disks with unit radius.

regular n-sided polygon with side length $2 \cdot r_u$. Observation 3 has been stated by several authors, for example [BK98], and it immediately yields Corollary 1. Observation 3 can be intuitively verified with Figure 2.1, which depicts an optimal packing of 7 disks with unit radius. However, we additionally provide a formal proof for Observation 3 with the following more general statement for ρ-bounded disk touching graphs.

Lemma 1. *Let $\mathcal{G} = (G, \mathcal{V})$ be a ρ-bounded disk touching graph that realizes $G = (V, E)$. There exists no vertex $v \in V$ with $\deg(v) > \lfloor \pi / \arcsin(1/(\rho+1)) \rfloor$ and if there exists a vertex $v' \in V$ with $\deg(v') = \pi / \arcsin(1/(\rho+1))$, then the disks in $\mathcal{V}$ corresponding to the neighbors of v' touch each other consecutively.*

Proof. We begin by calculating the radius r of a disk adjacent to $n \in \mathbb{N}$ disks with radius 1 such that the n disks touch each other consecutively. With basic trigonometry we obtain that $2 = 2(r+1)\sin((2\pi/n)/2)$ and therefore $r = 1/\sin(\pi/n) - 1$. The number n of disks can now be expressed as $n = \pi / \arcsin(1/(r+1))$. Since $\mathcal{G}$ is a ρ-bounded disk touching graph that realizes $G = (V, E)$, the maximum degree of any vertex in V is certainly bounded by the number $n' \in \mathbb{N}$ of disks with the smallest possible radius 1 that can be placed adjacent to a disk with the largest possible radius ρ. By our previous elaborations, this number is $n' = \lfloor \pi / \arcsin(1/(\rho+1)) \rfloor$ and furthermore, if $(\pi / \arcsin(1/(\rho+1))) \in \mathbb{N}$, then any disk of $\mathcal{V}$ that corresponds to vertex of V with degree n' has the largest possible radius ρ and its neighbors touch each other consecutively. □

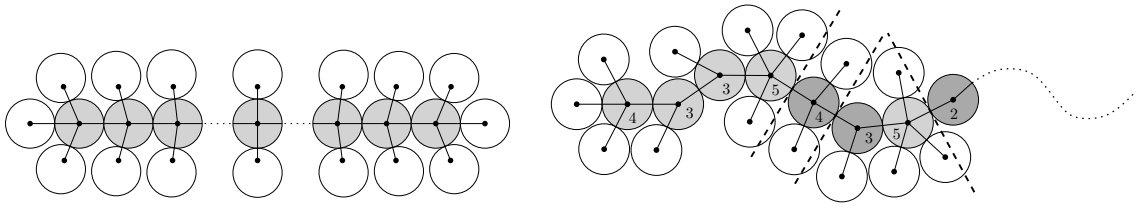
Corollary 2. *For spiders the ρ-bounded Disk Touching Graph Recognition (ρ-BDT) problem can be decided in $O(n)$ time in the Real RAM model for any $\rho \geq 1$, where n is the number of vertices of the given spider. A realization can be constructed in $O(n)$ time (if one exists) in the Real RAM model.*

We now return to the Unit Disk Touching Graph Recognition problem and show that it can be decided efficiently if the input graph is a caterpillar.

Theorem 1. *For caterpillars the Unit Disk Touching Graph Recognition (UDT) problem can be decided in $O(n)$ time, where n is the number of vertices of the given caterpillar. In the Real RAM model, a realization can be constructed in $O(n)$ time (if one exists).*

Proof. Let $C = (V, E)$ be a caterpillar and $p = (v_1, \ldots, v_k)$ be its inner path, where $1 < k \leq |V|$. If C is a yes-instance, then by Observation 3, we know that $\deg(v) \leq 5$ for any $v \in V$, which can be verified in $O(n)$ time. If furthermore $\deg(v) \leq 4$ for any $v \in V$, it is easy to construct a realization of C as a unit disk touching graph, as illustrated in Figure 2.2a. However, note that if V contains vertices of degree 5, it is not always possible to realize C as a unit disk touching graph, for example, if $\deg(v_i) = \deg(v_{i+1}) = 5$ for any $1 \leq i < k$.

We present very simple algorithm for deciding the UDT problem in caterpillars. Let $1 \leq f_1, \ldots, f_h, \leq k$ be the indices of vertices with degree 5 in p. Caterpillar C is a yes-instance



(a) If no vertex has degree 5, there always exists a realization.

(b) (Degree 5)-vertices can be realized if the path is not narrow (darkgray disks).

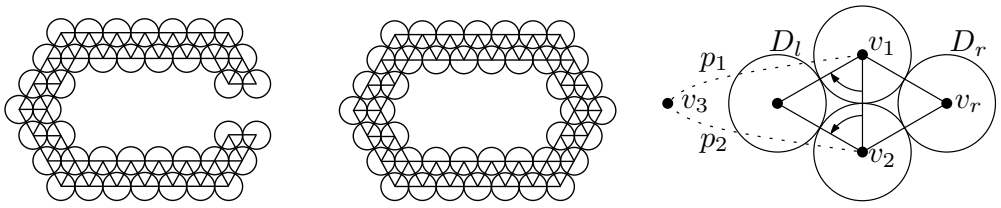
Figure 2.2.: Realizing caterpillars as unit disk touching graphs.

if and only if for any $1 \leq i < h$ there exists an index $f_i < j < f_{i+1}$ such that $\deg(v_j) \leq 3$, which can be tested by performing a simple linear sweep through p.

The correctness of the aforementioned algorithm is implied by the following constructive approach. We refer to the disks representing $v_1, \ldots, v_k$ as $D_1, \ldots, D_k$ respectively. We start by creating disk D_1. We place disks $D_1^1, \ldots, D_1^{\deg(v_1)-1}$ close together and adjacent to D_1. These disks represent the leafs adjacent to v_1. We place the disk D_2 in the center of the free space around D_1 that is not occupied by any of the disks $D_1^1, \ldots, D_1^{\deg(v_1)-1}$. We add disks for the remaining inner vertices of p and their leafs iteratively. Consider index $1 \leq i < k$. The disks $D_i^1, \ldots D_i^{\deg(v_i)-2}$, which represent the leafs adjacent to v_i, are placed iteratively touching D_i and as close to D_{i-1} as possible. If there is enough free space around D_i to place D_{i+1}, we place D_{i+1} centered in said free space. If, however, there is not enough space remaining, we report that C is a no-instance of the UDT problem. In order to test whether there exists enough free space around D_i, we consider the line l_i which is a tangent of both D_{i-1} and D_i and which intersects $D_{i-1} \cap D_i$. If none of the previously added disks, except for D_{i-1} and D_i, intersect l_i there exists enough free space around D_i for all of its neighbors even if the degree of v_i is $\deg(v_i) = 5$. Note that if we handle odd numbers of leafs in a balanced fashion, we can ensure that the disks representing p follow some direction monotonously and therefore that only leafs of D_{i-1} can intersect l_i, see Figure 2.2b. If this is the case, we say that p is *narrow* at v_i. If p is *narrow* at v_i and if $\deg(v_i) \leq 4$, there is still enough space to place all neighbors of D_i. If, however, the degree of v_i is $\deg(v_i) = 5$, there is not enough space remaining to place D_{i+1} and the three leafs of D_i.

For the correctness of the decision algorithm described earlier, consider the following facts. Path p is narrow at any vertex with degree 5. If p is narrow at some vertex v_i and if $\deg(v_{i+1}) = 4$, it follows that p is narrow at v_{i+1} as well. However, if $\deg(v_{i+1}) \leq 3$, path p is not narrow at v_{i+1}. Note that, unlike the constructive algorithm, the decision algorithm does no require the Real RAM model. □

After these positive results, in the remainder of this section we show that UDT remains $\mathcal{NP}$-hard even if the input graph is outerplanar. We begin by defining the following property, which is an essential tool for the upcoming $\mathcal{NP}$-hardness proof. Let $G = (V, E)$ be a graph and let $\mathcal{G} = (G, \mathcal{V})$ be a realization of G as a unit disk touching graph. Graph G is called *UDT-rigid* if and only if any realization of G as a unit disk touching graph is congruent to $\mathcal{G}$. Both Figure 2.3a and Figure 2.3b depict realizations of graphs that are UDT-rigid, but only the first figure satisfies the preconditions of the following lemma, which provides a sufficient but not necessary condition for UDT-rigidity. This condition also applies to many of the components utilized in the proof of the upcoming Theorem 2.



(a) UDT-rigid by Lemma 2. (b) UDT-rigid. (c) Proof for Lemma 2.

Figure 2.3.: Lemma 2 provides a sufficient but not necessary condition for UDT-rigidity.

Lemma 2. *Let $G = (V, E)$ be a graph and $\mathcal{G}$ be a realization of G as a unit disk touching graph. If G is 2-connected and the planar drawing of G induced by $\mathcal{G}$ is outerplanar and internally triangulated, then G is UDT-rigid.*

Proof. We show by induction that our hypothesis is true for any natural number $n = |V|$ of vertices. For the induction base case we consider $1 \leq n \leq 3$. If $n = 1$, then G is obviously UDT-rigid. If $n = 2$ then G is not 2-connected. If $n = 3$ and G is not 2-connected, there is nothing to show. If, however, G is 2-connected, then G is a complete graph. In this case G is obviously UDT-rigid, which concludes the induction base case.

For the induction step, consider any $n > 3$ and assume that our hypothesis holds true for all graphs with at most $n-1$ vertices. If G is not 2-connected or there exists no realization of G that meets the stated preconditions, there is nothing to show. Let therefore $\mathcal{G}$ be a realization of G that does meet our preconditions and assume G is 2-connected. Graph G is outerplanar since $\mathcal{G}$ induces an outerplanar drawing of G. Since G is outerplanar there exists a vertex $v_r \in V$ with $\deg(v_r) \leq 2$ and since G is 2-connected and $n > 3$ we specifically know that $\deg(v_r) = 2$. Let $v_1, v_2 \in V$ be the neighbors of v_r. Removing the disk corresponding to v_r from $\mathcal{G}$ yields a unit disk touching graph $\mathcal{G}'$ that realizes the subgraph $G' = (V', E')$ of G that is induced by the vertex set $V' = V \setminus \{v_r\}$. The induced planar drawing of $\mathcal{G}'$ is obviously still outerplanar and internally triangulated.

The following steps are illustrated in Figure 2.3c. The number of vertices of G is $n > 3$ and G is 2-connected. By Menger's Theorem there exists two internally vertex-disjoint paths p_1, p_2 (via v_1 and v_2 respectively) between v_r and some vertex $v_3 \in V, V'$ with $v_3 \neq v_r, v_1, v_2$. The existence of p_1 and p_2 together with $\deg(v_r) = 2$ and with the fact the planar drawing of $\mathcal{G}'$ is internally triangulated imply that $e = \{v_1, v_2\} \in E, E'$. The existence of e, on the other hand, implies that G' is 2-connected since G is 2-connected and since $|V|' \geq 3$. G' and $\mathcal{G}'$, therefore, meet all of our preconditions and by our induction hypothesis we know that G' is UDT-rigid since $|V'| = n - 1$.

We now re-add the vertex v_r and edges $\{v_r, v_1\}, \{v_r, v_2\}$ to G' (resulting in G) and add a corresponding disk D_r to $\mathcal{G}'$. In the following paragraphs we argue that there exists exactly one location where we can place D_r (namely the same location used in $\mathcal{G}$) implying that G is also UDT-rigid since the obtained realization is then congruent to $\mathcal{G}$, which concludes the induction step and the proof.

Due to the existence of e, we know that the two disks D_1 and D_2 that correspond to v_1 and v_2 respectively touch each other. Hence, there exist exactly two locations for disks that correspond to common neighbors of both v_1 and v_2. One of these locations is the location of the disk corresponding to v_r in $\mathcal{G}$. It suffices to show that in both $\mathcal{G}$ and $\mathcal{G}'$ there exists a disk D_l that occupies the other possible location.

The existence of p_1 and p_2 implies that the degree of both v_1 and v_2 is at least 3 and that there exists a path p_3 between v_1 and v_2 that contains v_3 and that does not contain v_r

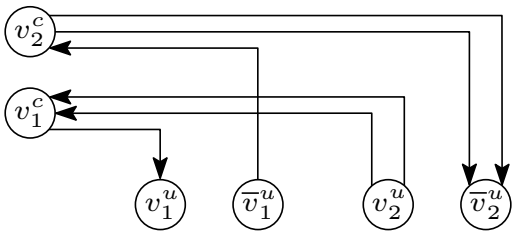


Figure 2.4.: Orientable auxiliary multigraph for 3SAT formula $(u_1 \vee u_2 \vee u_2) \wedge (\overline{u}_1 \vee \overline{u}_2 \vee \overline{u}_2)$.

or e. Assume, without loss of generality, that in the drawing induced by $\mathcal{G}$ v_r is the neighbor of v_1 that clockwise precedes v_2 and that v_r is the neighbor of v_2 that clockwise succeeds v_1. Let v_1' be the neighbor of v_1 that clockwise succeeds v_2 and let v_2' be the neighbor of v_2 that clockwise precedes v_1. By the fact that the drawing of G induced by $\mathcal{G}$ is internally triangulated and that v_r has to be adjacent to the outer face of the drawing, we can conclude that $v_1' = v_2'$, which, therefore, is another common neighbor of v_1 and v_2 whose corresponding disk has to be D_l. □

We are now prepared to prove the final result of this section.

Theorem 2. *The Unit Disk Touching Graph Recognition (UDT) problem is $\mathcal{NP}$-hard even for outerplanar graphs.*

Proof. We perform a polynomial-time reduction from the 3-Satisfiability (3SAT) problem. Let (U, C) be a 3SAT instance where $U = \{u_1, \ldots, u_n\}$ with $n = |U|$ is a set of variables and $C = \{c_1, \ldots, c_m\}$ with $m = |C|$ is a set of clauses over U. Recall that in a 3SAT instance the cardinality of any clause is 3. The general outline for this proof is as follows. We start by creating an auxiliary multigraph G' that encodes (U, C). We construct a huge outerplanar graph G that resembles G' such that G can be realized as a unit disk touching graph if and only if C is satisfiable. In order to construct G we utilize a grid with multiple layers. This long proof is divided into multiple sections. In Section (a) we introduce the auxiliary multigraph G' that encodes (U, C). In Section (b) we describe the *high-level* grid R on which we place different types of gadgets. In Section (c), by utilizing a *mid-level* grid R_W, we present a schematic view for each of the gadgets and discuss their purposes. In Section (d) we get more technical. We use a *low-level* grid D to describe how to construct a specific unit disk graph for each of the gadgets. The thereby realized graphs will serve as subgraphs for the graph G, which is our UDT instance. Finally, in Section (e) we conclude our proof by putting all pieces together.

(a) The auxiliary multigraph G'

Breu and Kirkpatrick were able to show that the Unit Disk Touching Graph Recognition problem is $\mathcal{NP}$-hard in planar graphs [BK98] by formulating a ($\leq$,3,3)-Satisfiability ($\leq$33SAT) instance as an auxiliary graph. They introduced the notion of this graph being 'orientable', a property that is satisfied if and only if the $\leq$33SAT instance is a yes-instance. This first step of our proof is heavily inspired by this concept. Our auxiliary multigraph is defined similarly to Breu and Kirkpatrick's auxiliary graph and we also borrow the notion of this multigraph being orientable. However, in order to strengthen their result to the outerplanar case we require a completely different strategy for constructing our UDT instance G, which is the main part our proof explained in the remaining Sections (b) – (e).

Our auxiliary multigraph $G' = (V', E')$ contains a *clause* vertex for each clause in C and a *literal* vertex for each literal over U. The edge set E' contains an edge between a clause vertex and a literal vertex if and only if the corresponding clause contains the corresponding literal. More precisely, the vertex set is $V' = V_C \cup V_U \cup \overline{V}_U$, where $V_C = \{v_1^c, \ldots, v_m^c\}$

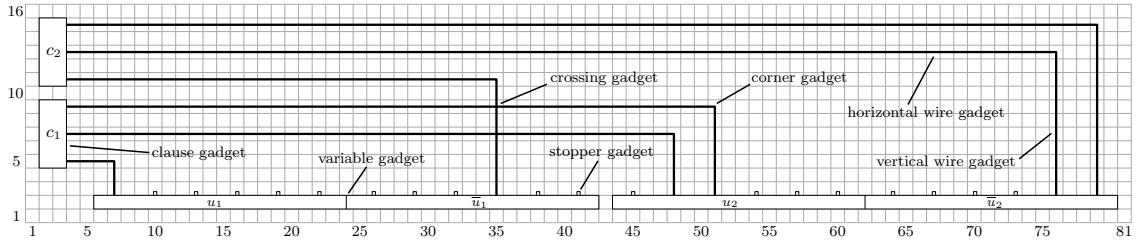


Figure 2.5.: High-level grid for 3SAT formula $(u_1 \vee u_2 \vee u_2) \wedge (\overline{u}_1 \vee \overline{u}_2 \vee \overline{u}_2)$.

contains a clause vertex for each clause in C and the vertex sets $V_U = \{v_1^u, \ldots, v_n^u\}$ and $\overline{V}_U = \{\overline{v}_1^u, \ldots, \overline{v}_n^u\}$ contain literal vertices, one for each literal over U. The edge set is $E' = \{\{v_i^c, v_j^u\} \mid u_j \in c_i, 1 \leq i \leq m, 1 \leq j \leq n\} \cup \{\{v_i^c, \overline{v}_j^u\} \mid \overline{u}_j \in c_i, 1 \leq i \leq m, 1 \leq j \leq n\}$. See Figure 2.4 for an illustration of this transformation.

We say G' is *orientable* if and only if we can assign a direction to each edge such that (1) each clause vertex has an outdegree of at least 1 and (2) at least one vertex of every pair of literal vertices has indegree 0. More precisely, G' is orientable if and only if there exists a directed multigraph $\overrightarrow{G}' = (V', \overrightarrow{E}')$ where $\overrightarrow{E}'$ contains either directed edge (v^c, v^l) or (v^l, v^c) for any edge $\{v^c, v^l\} \in E'$ and (1) the outdegree of any vertex in V_C is at least 1 and (2) the indegree of v_i^u or $\overline{v}_i^u$ is 0 for any $1 \leq i \leq |U|$. Breu and Kirkpatrick presented the following interpretation: An edge directed from a clause vertex v^c (representing some clause c) to a literal vertex v^l (representing some literal l) means that v^c suggests that l should be true in order for c to be satisfied. With this interpretation, condition (1) means that any clause vertex makes such a suggestion. If we could meet all these suggestions, the corresponding truth assignment would satisfy C. Of course, this is not always possible as distinct clause vertices may suggest opposing literals. However, if condition (2) holds, this is not the case and C must be satisfiable. On the other hand, if C is satisfiable then there exists a truth assignment t for U that ensures that at least one literal in each clause is true with respect to t and since whenever a literal is true with respect t its counterpart has to be false with respect to t, we know that G' is orientable and therefore, (U, C) is a yes-instance if and only if G' is orientable.

(b) High-level construction of the UDT instance G

For the high-level construction of our UDT instance G we utilize the $(6 \cdot |C| + 4) \times ((18 \cdot |C| + 2) \cdot |U| + 5)$ high-level grid R with square *cells*, as illustrated in Figure 2.5. Observe the resemblance of the high-level structure depicted in this figure to the auxiliary multigraph displayed in Figure 2.4. We refer to the cell in the i-th row from the bottom and the j-th column from the left as $R[i][j]$, for example, $R[6 \cdot |C| + 4][(18 \cdot |C| + 2) \cdot |U| + 5]$ refers to the top-right cell.

On our grid, we position different kinds of gadgets that take up one or more cells each. First we add $|C|$ *clause* gadgets, each of which takes up 10 cells in a 5×2 pattern. These gadgets represent the clauses of our 3SAT instance. The bottom-left cell of the of the i-th clause gadget is placed in cell $R[6 \cdot (i-1) + 5][2]$. Next we place $|U|$ *variable* gadgets, each of which takes up $18 \cdot |C| + 1$ cells in a $1 \times (18 \cdot |C| + 1)$ pattern. These gadget represent the variables in U, the left half of each gadget representing the positive and the right half representing the negative literal of the corresponding variable. The leftmost cell of the i-th variable gadget is placed in cell $R[2][(18 \cdot |C| + 2) \cdot (i-1) + 6]$.

Let $c_{i,j}$ be the j-th literal of the i-th clause. This literal is represented by the $(2 \cdot (j-1) + 1)$-th row (from the bottom) of the clause gadget representing c_i. We want to connect the right cell of this row of the clause gadget with the correct half of the variable gadget representing the variable of $c_{i,j}$. To this end, we utilize a structure called wire, which is

realized by four different gadget types. First we place a 1×1 *corner* gadget in the row that represents $c_{i,j}$ and in the column that contains the cell of the variable gadget that we want to connect the clause gadget to. More precisely, for any $1 \leq i \leq |C|$ and any $1 \leq j \leq 3$ we place a corner gadget in cell $R[6i+2j-3][(18|C|+2)(k-1)+9(i-1)+3(j-1)+7]$ if $c_{i,j}$ is the positve literal of u_k and in cell $R[6i+2j-3][(18|C|+2)(k-1)+9(i-1)+3(j-1)+26]$ if it is the negative literal, where k is the index of the variable of U that belongs to $c_{i,j}$.

In each of the cells between a corner and its corresponding clause gadget we place a 1×1 *horizontal wire* gadget and in each of the cells between a corner and its corresponding variable gadget we place a 1×1 *vertical wire* gadget. More precisely, let $R[i][j]$ be a cell that contains a corner gadget. We place a horizontal wire gadget in each of the cells $R[i][4]$, $R[i][5]$, $R[i][6]$,...,$R[i][j-1]$ and we place a vertical wire gadget in each of the cells $R[i-1][j]$, $R[i-2][j]$, $R[i-3][j]$,...,$R[3][j]$. We repeat this process for any corner. In any cell in which we added a horizontal, as well as a vertical wire gadget, we remove these two gadgets and insert a 1×1 *crossing* gadget instead. Finally, for any $1 \leq k, i \leq |C|$ and any $1 \leq j \leq 3$ we place a 1×1 *stopper* gadget at cells $R[3][(18|C|+2)(k-1)+9(i-1)+3(j-1)+7]$ and $R[3][(18|C|+2)(k-1)+9(i-1)+3(j-1)+26]$ if these do not contain vertical wire gadgets.

(c) Mid-level gadget construction

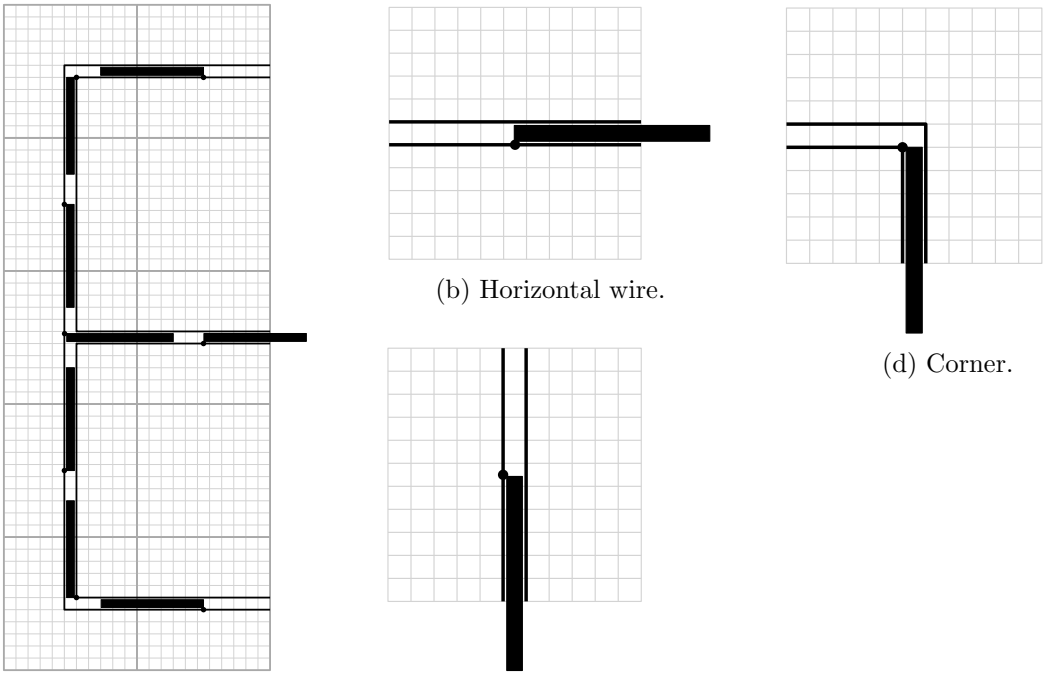
Each of our gadgets ultimately corresponds to a subgraph of our UDT instance G and in Section (d) we will construct these subgraphs as unit disk graphs on a low-level grid D. As a preparation for this section, sneak a peek at Figure 2.9 from Section (d). This figure represents a horizontal wire gadget realized as a unit disk graph on the low-level grid D which is composed of unit disks. Note how the black disks assume a rectangular shape and observe that the graph realized by these disks is (UDT-) rigid by Lemma 2. In a similar fashion, we can construct rigid subgraphs whose realizations assume any kind of rectangular shape. In this section we present a schematic view for each of our gadgets in which we represent the different components by simple rectangular shapes that resemble the actual disk graphs that are constructed in more detail in Section (d).

As seen in Section (b), each gadget consists of one or multiple cells of the high-level grid R. For our schematics, we subdivide each of these cells further into W^2 square *regions* by utilizing a $W \times W$ *mid-level* grid R_W, where $W \in \mathbb{N}$ is an odd natural number with $W - 2 \equiv 0 \mod 3$. We shall later specify $W = 47$, however, for improved readability, our illustrative figures are often created using smaller values for W. We refer to the region in the i's row from the bottom and the j's column of R_W from the left as $R_W[i][j]$.

(c.1) The horizontal wire, vertical wire and corner gadgets

Figure 2.6b depicts the schematic of a horizontal wire gadget. It consists of two rigid, parallel line segments and a rigid rectangle. The line segments are parallel to the bottom of the cell, encompass its entire width and are located a the center of its height. More precisely, the two line segments are the borders between the $\lfloor W/2 \rfloor$-th and $(\lfloor W/2 \rfloor + 1)$-th and the $\lceil W/2 \rceil$-th and the $(\lceil W/2 \rceil + 1)$-th row of R_W. The distance between the two line segments is just barely larger than the height of the rectangle that is 'anchored' in the middle of the lower line segment, meaning that this *anchor* point is the only point where the rectangle actually touches the line segments. The width of the rectangle is almost $W/2 + (W-2)/3$ regions long so that it sticks out of the cell by $(W-2)/3 = 15$ regions for $W = 47$.

In the low-level unit disk graph version of this gadget, the anchor point is realized as a single 'anchor' disk of the bottom line segment (the gray disk in Figure 2.9) that touches a so called 'chain' disk of the rectangle, which itself touches two of the disks realizings the rectangle. We do not consider a fixed combinatorial embedding, therefore, the rectangle



(a) Clause. (b) Horizontal wire. (c) Vertical wire. (d) Corner.

Figure 2.6.: Mid-level schematics for the different gadgets with $W = 11$.

can be embedded such that it sticks out to the right, as depicted in Figure 2.6b, or, alternatively, it can be embedded such that it sticks out to the left. Accordingly, we say that the gadget is *oriented* to the right or left. Imagine two of these gadgets being placed next to each other such that the two line segments of both gadgets align and thereby form two longer line segments whose length is two times the length of a cell. If the right gadget is oriented to the left, then the left gadget has to be oriented to the left as well since the space needed to embed its rectangle to the right is already taken up by the rectangle of the right gadget. Now imagine three horizontal wire gadgets being placed next to each other. If the right gadget is oriented to the left and the left gadget is oriented to the right it is impossible to embed the rectangle of the middle gadget, which, on disk graph level, means that the subgraph of G that corresponds to these three gadgets can not be realized as a unit disk touching graph.

The overarching idea is now to define such an orientation concept for the vertical wire, corner and crossing gadgets as well and to design the clause and variable gadgets such that G can be realized as a unit disk touching graph if and only if the multigraph G' is orientable and therefore if and only if C is satisfiable.

(c.2) The vertical wire and corner gadgets

The idea for the vertical wire and the corner gadgets, illustrated in Figure 2.6c and Figure 2.6d, and their corresponding orientation concepts is very similar to the idea for the horizontal wire gadget. The vertical wire gadget is identical to a horizontal wire gadget, except that it is rotated clockwise by 90°. A vertical wire can be *oriented* to the top or to the bottom. The corner gadget enables us to connect a horizontal wire to a vertical wire. It consists of two rigid polylines and another rectangle. The two polylines describe the border of the union of the regions $R_W[\lfloor W/2 \rfloor + 1][1], \ldots, R_W[\lfloor W/2 \rfloor + 1][\lfloor W/2 \rfloor + 1], \ldots, R_W[1][\lfloor W/2 \rfloor + 1]$. A rectangle is anchored in the bottom-left of region $R_W[\lfloor W/2 \rfloor + 1][\lfloor W/2 \rfloor + 1]$ such the cell can be *oriented* to the left or to the bottom. The length of the rectangle is once again chosen such that it sticks out of the cell by $(W-2)/3$ regions. Now imagine some horizontal wires being placed next to each other

followed by a corner and some vertical wires. If the orientation of the leftmost horizontal wire is 'right', than this orientation propagates to the lowest vertical wire, which then has to be oriented to the bottom. The same holds true for the opposite direction, so that it is not possible to embed the rectangle of a cell of which both adjacent cells are oriented towards it.

(c.3) The clause gadget

Recall that each clause in C has three literals and that if auxiliary multigraph G' is orientable, each clause vertex has outdegree at least 1 in any corresponding directed multigraph $\overrightarrow{G}'$. In order to emulate this concept, our clause gadget should be designed such that at least one of the three connected horizontal wires has to be oriented to the right. An illustration of the schematic for the clause gadget is depicted in Figure 2.6a. Recall that the clause gadget occupies 5×2 cells of the high level grid. The core of this gadget is a 'T-type' crossing in which a rectangle is anchored such that it can be embedded in three possible ways: To the top, to the right or to the bottom. This 'T-type' crossing is located in the middle-left cell. It consists of a rigid line segment and two rigid polylines that together describe the border of the union of the regions $R_W[1][\lfloor W/2 \rfloor + 1], \ldots, R_W[W][\lfloor W/2 \rfloor + 1]$ and $R_W[\lfloor W/2 \rfloor + 1][\lfloor W/2 \rfloor + 2], \ldots, R_W[\lfloor W/2 \rfloor + 1][W]$. The cell above (below) the 'T-type' cell is simply a vertical wire gadget. Above (below) this vertical wire gadget we place a rotated corner gadget that leads to the right cell, where we place a horizontal wire gadget. Another horizontal wire gadget is also placed to the right of the 'T-type' cell. The remaining two cells stay empty. The rectangle is anchored at the top-left corner of the central region of the 'T-type' cell. On disk graph level, the rectangle can, once again, be embedded to the bottom as well as to the top since we do not consider a fixed embedding. However, this time the 'anchor' and 'chain' disks are chosen such that the rectangle can also be rotated around its 'chain' disk and therefore be embedded to the right. The rectangle has the same height as the previous rectangles and its length is chosen such that it sticks out of the 'T-type' cell by $(W-2)/3$ regions if it is embedded to the right or the bottom and $(W-2)/3+1$ regions if it is embedded so the top, so for $W = 47$ it sticks out of its cell by 15 or 16 regions. Recalling the functionality of the the previously introduced gadgets, one of the three horizontal wire gadgets in the clause gadget and therefore one of the three horizontal wire gadgets next to the clause gadget has to be oriented to the right.

(c.4) The variable and stopper gadgets

Recall that if auxiliary multigraph G' is orientable, then at least one of the literal vertices of any variable has indegree 0 in any corresponding directed multigraph $\overrightarrow{G}'$. Also recall that each variable gadget consists of $1 \times (18 \cdot |C| + 1)$ cells and that the left half of a variable gadget represents the positive and the right half of the gadget represents the negative literal of the corresponding variable. We need to ensure that for either the left or the right side of each variable gadget, all connected vertical wire gadgets have to be oriented to the top, away from the variable gadget. Figure 2.7 depicts a variable gadget as well as the cells directly above it. The central cell of the variable gadget is basically a modified version of the horizontal wire gadget. The main difference is that the two line segments as well as the anchor point are moved up into the second row from the top of the mid-level grid R_W and that the distance between the two line segments is much smaller. They now enclose only the upper part of the second row of regions from the top. To the right (left) of this central cell we place another such modified horizontal wire gadget but without a rectangle. Next to this horizontal wire we place $3 \cdot (|C| - 1)$ times the following triple of cells. The first of the triple's cells is a 't-type' crossing which consists of four rigid polylines that together describe the border of the union of the regions $R_W[\lfloor W/2 \rfloor + 1][W/2 \rfloor + 1], \ldots, R_W[W][\lfloor W/2 \rfloor + 1]$ and the upper parts of the regions $R_W[W-1][1], \ldots, R_W[W-1][\lfloor W/2 \rfloor]$ and $R_W[W-1][\lfloor W/2 \rfloor + 2], \ldots, R_W[W-1][W]$, see the bottom cell in the middle of Figure 2.7. The

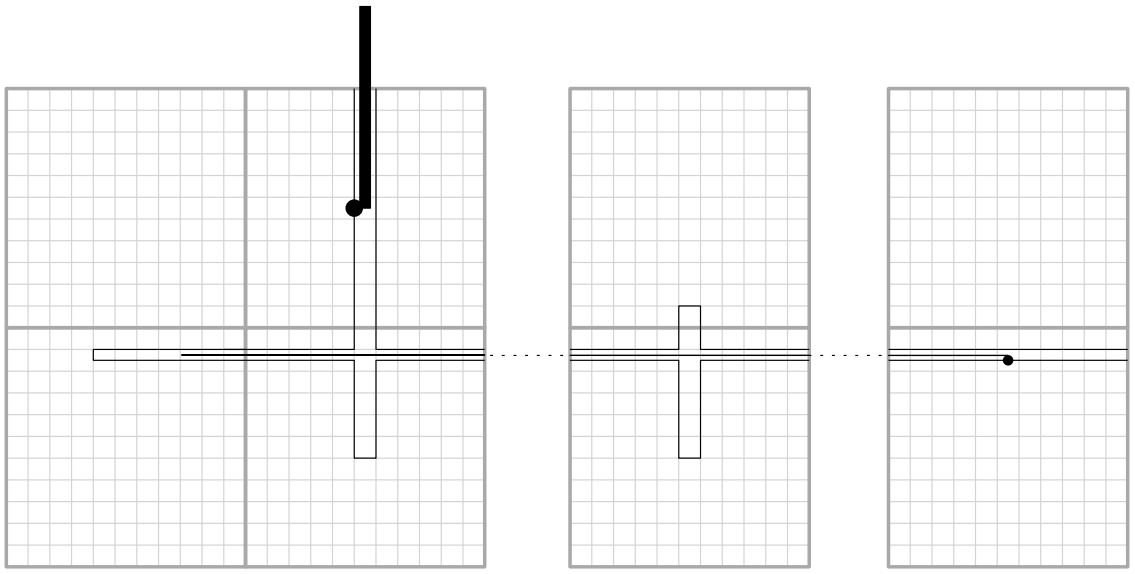
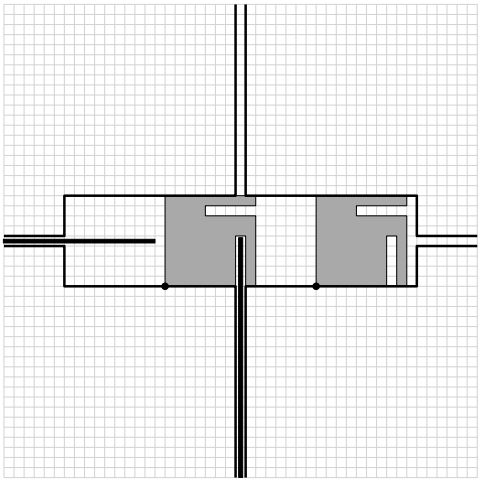


Figure 2.7.: Mid-level schematics for parts of the left side of a variable gadget together with stopper and vertical wire gadgets for $W = 11$ and $|C| = 1$.

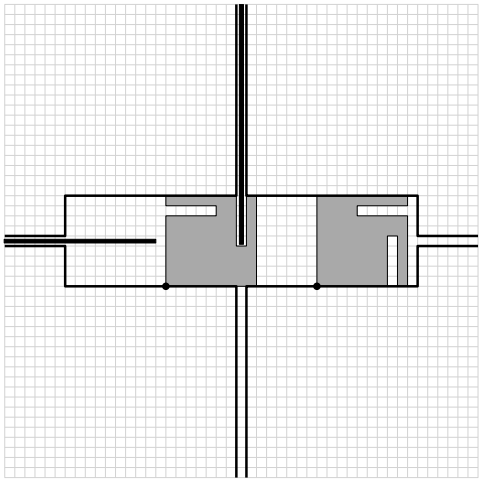
other two triple cells are once again copies of the modified version of the horizontal wire gadget without a rectangle. After all the cell triples we place one more 't-type' cell and the last cell consists of just one polyline that describes the border of the union of the upper parts of the regions $R_W[W-1][1], \ldots, R_W[W-1][\lfloor W/2 \rfloor + 1]$ on the right side and mirrored accordingly on the left side as seen in the bottom-left cell in Figure 2.7. The height of the rectangle anchored in the center of the central cell is chosen barely smaller than the distance between the two line segments (which now are very close together) and its length is chosen such that it sticks out by $(3 \cdot (|C| - 1) + 2) \cdot W + (W - 2)/3$ regions. It has to be embedded either to the right or the left and since it is located in the second row from the top of any of the 't-type' cells that it is embedded in and since rectangles of wire gadgets stick out by $(W-2)/3$, vertical wire gadgets connected to these 't-type' cells have to be oriented to the top. Stopper gadgets simply consist of a rigid polyline describing the border of the bottom-middle region and their purpose is to hold the different parts of the variable gadget together, see the upper cell in the middle of Figure 2.7.

(c.5) The crossing gadget

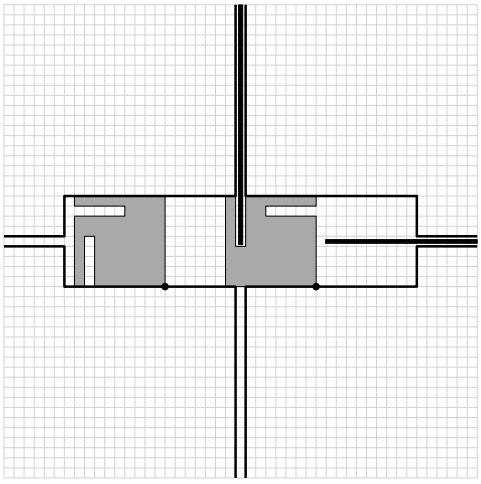
In the crossing gadget we need to propagate the orientation of the horizontal wires from left to right and vice versa and the orientation of the vertical wires from top to bottom and vice versa. Figure 2.8a, Figure 2.8b, Figure 2.8c and Figure 2.8d illustrate the schematic for this complicated gadget. The crucial components of this gadget are two identical huge, rigid squares with some notches in them. We assign $W = 47$. The side length of such a 'notched square' is barely smaller than 9 regions. The notches are created by cutting out the eighth region from the left of the five lowest rows and the five rightmost regions of the eighth row from the bottom of the square, see Figure 2.8a. The bottom-left point of the two 'notched squares' is anchored at the bottom of a huge cavity in the center of the cell. The cavity is described by four polylines and its exact dimension as well as the exact coordinates for the two anchor points can be taken from Figure 2.8a. Like the rectangle in the horizontal wire gadget, the 'notched squares' can be embedded to the left or the right since we do not consider a fixed embedding. Like the rectangle in the clause gadget, the 'notched squares' can be rotated, but in this case the cavity is chosen large enough that they can be rotated from both sides. We can therefore obtain essentially four embeddings per 'notched square'.



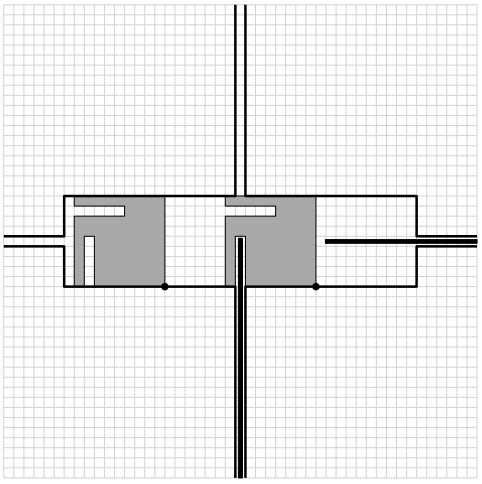
(a) Oriented to the top and the right.



(c) Oriented to the bottom and the right.



(b) Oriented to the bottom and the left.



(d) Oriented to the top and the left.

Figure 2.8.: Mid-level schematics for the crossing gadget with $W = 47$.

Recall that the rectangles of the wires stick into the crossing's cell by $(W-2)/3 = 15$ regions. For the horizontal orientation propagation, note how both 'notched squares' have to be embedded to the right (left) if the horizontal wire gadget to the left (right) is oriented to the right (left). For the vertical orientation propagation, we adjust all vertical wire gadgets that are located above or below a crossing gadget such that their rectangles stick into the crossing's cell by 24 regions. This can be done as follows. If the cell above as well as the cell below a vertical wire contain a crossing, simply increase the rectangle's length by 9 regions. If only the cell below (above) the vertical wire contains a crossing, increase the length of the rectangle by $4 + 1/2$ regions and move the anchor point down (up) by $4 + 1/2$ regions, so that if the rectangle is oriented to the top (bottom) it sticks out into the cell above (below) as usual. Note how in Figure 2.8a the rectangle of the vertical wire on the bottom fits into the notch at the bottom of the 'notched square'. It is not possible to simultaneously embed a rectangle from a vertical wire above oriented to the bottom since on top of the 'notched square' there is no notch. However, the notch on the right side of the square is chosen such that the rectangle from the vertical wire above can be embedded, if the 'notched square' is first embedded to the left and then rotated back to the right as depicted in Figure 2.8c in which case there is no notch located at the 'notched square's bottom. The same holds true if the right 'notched squares' is embedded to the middle. We say that the crossing gadget is *oriented* to the right (left) if the right (left) 'notched square' is embedded to the right (left) and we say that is is *oriented* to the top (bottom) if there is no notch on the top (bottom) side of the 'notched square' embedded in the middle of the cavity. Obviously we can not orient both adjacent horizontal wires towards the crossing since then both 'notched squares' would have to be embedded in the middle. It is also not possible to orient both adjacent vertical wires towards the crossing since both would stick into the crossing by 24 and therefore overlap in the central region $R_W[24][24]$ of the crossing gadget.

(d) Constructing the gadgets in detail

In this section we construct a unit disk graph for each of our gadgets that realizes the corresponding mid-level schematic presented in Section (b). Each of the graphs realized by these unit disk touching graphs will serve as a subgraph of G. Recall that we want to be able to construct rigid, rectangular shapes. We therefore design each of the high-level cells on a low-level grid D, composed of unit disks in a hexagonal pattern as illustrated in Figure 2.9. To make sure that two high-level cells designed on such low-level grids can be properly aligned we need to specify the exact dimensions for D. We define that D consists of an even number Y of horizontal rows each of which consists of X unit disks. Without loss of generality, we define the unit size to be 1, therefore, each of our disks has radius 1. Our low-level grid is hexagonal in the sense that each (inner) disk is surrounded by six other disks and the centers of these disks form a regular hexagon. Due to this hexagonal nature, the rows, in turn, are horizontally offset by $1/2$ and the vertical distance between the centers of two consecutive rows is $2 \cdot \sin(\pi/3) = \sqrt{3}$, while the horizontal distance between the centers of two consecutive disks in a row is simply 2. The horizontal length of one row can therefore be described as $2X$, while the height of the entire grid cell is $(Y-1)\sqrt{3}+2$. We therefore define $X = \lceil (Y-1)\sqrt{3}/2+1 \rceil$ in order to obtain grid cells that are approximately square shaped. Recall that our mid-level grid R_W divided each cell into W^2 square regions in a $W \times W$ pattern, where $W \in \mathbb{N}$ is an odd natural number with $W - 2 \equiv 0 \mod 3$. We choose $Y = 4 \cdot W \cdot S(|C|, |U|)$, where S is a scaling function in $|C|$ and $|U|$ that will be specified later. For now, assume that $S(|C|, |U|) \gg 1$ is a natural number. We refer to the j-th disk from the left in the i-th row from the bottom as $D[i][j]$. Note that $Y/2$ is an even natural number and that Y/W is a natural number. We use the (kY/W)-th and the $(kY/W + 1)$-th row of disks of D from the bottom to describe the border between the k-th and the $(k+1)$-th row of regions of R_W from the

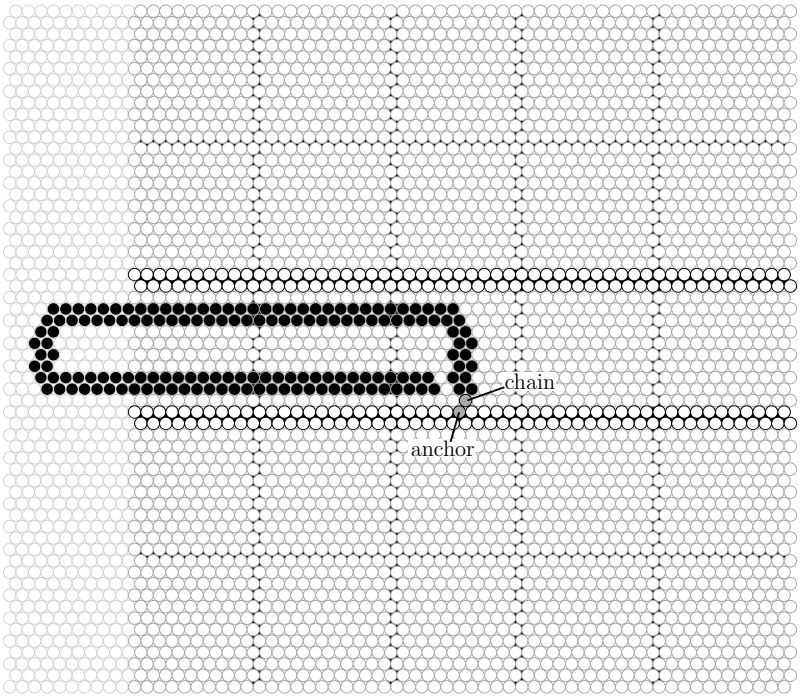


Figure 2.9.: Low-level construction of the horizontal wire gadget with $W = 5$ and $S(|C|, |U|) = 3$.

bottom as suggested in Figure 2.9. Since X/W is not necessarily a natural number there is not such a straight forward way to evenly split the low-level grid D horizontally. In Figure 2.9 we nevertheless included some vertical lines in order to make the resemblance to the mid-level grid R_W more obvious. Note, however, that the vertical slabs in this figure are not evenly sized. For our construction this does not cause any issues.

(d.1) The horizontal wire gadget in detail

Recall that the horizontal wire gadget consists of two rigid, parallel line segments and a rectangle anchored in the middle of the lower line segment that can be embedded to either the left or the right. The two line segments are realized by the disks of the rows $\lfloor W/2 \rfloor Y/W$ and $\lfloor W/2 \rfloor Y/W + 1$ as well as $\lceil W/2 \rceil Y/W$ and $\lceil W/2 \rceil Y/W + 1$, counted from the bottom, see Figure 2.9. The adjacencies are chosen according to the grid contacts such that we indeed obtain two rigid components. We define disk $D[\lfloor W/2 \rfloor Y/W + 1][\lceil X/2 \rceil]$, which is the middle disk of upper row of the bottom line segment, to be the *anchor* of the rectangle. The anchor is the only disk of the line segments touched by any of the rectangle's disks. We define disk $D[\lfloor W/2 \rfloor Y/W + 2][\lceil X/2 \rceil]$ to be the one and only disk of the rectangle that does touch the anchor and we call this vertex the *chain*.

Note that since the chain is only allowed to touch the anchor but not its right neighbor the rectangle can not actually be be embedded as depicted in Figure 2.9. Instead the chain has to be placed infinitesimally higher in any realization of the horizontal wire gadget subgraph. In Figure 2.9 the chain is simply depicted as is, since it allows the reader to easily follow the remainder of rectangle's construction, which also takes place on the low-level grid. It works as follows. Starting with the row above the chain we add two disks per row until we reach the row located three rows below the upper line segment. More precisely, we add disks $D[i][\lceil X/2 \rceil]$ and $D[i][\lceil X/2 \rceil + 1]$, where i is odd and $\lfloor W/2 \rfloor Y/W + 3 \leq i \leq \lceil W/2 \rceil Y/W - 5$, and we add disks $D[i][\lceil X/2 \rceil - 1]$ and $D[i][\lceil X/2 \rceil]$, where i is even and $\lfloor W/2 \rfloor Y/W + 4 \leq i \leq \lceil W/2 \rceil Y/W - 4$, to the rectangle.

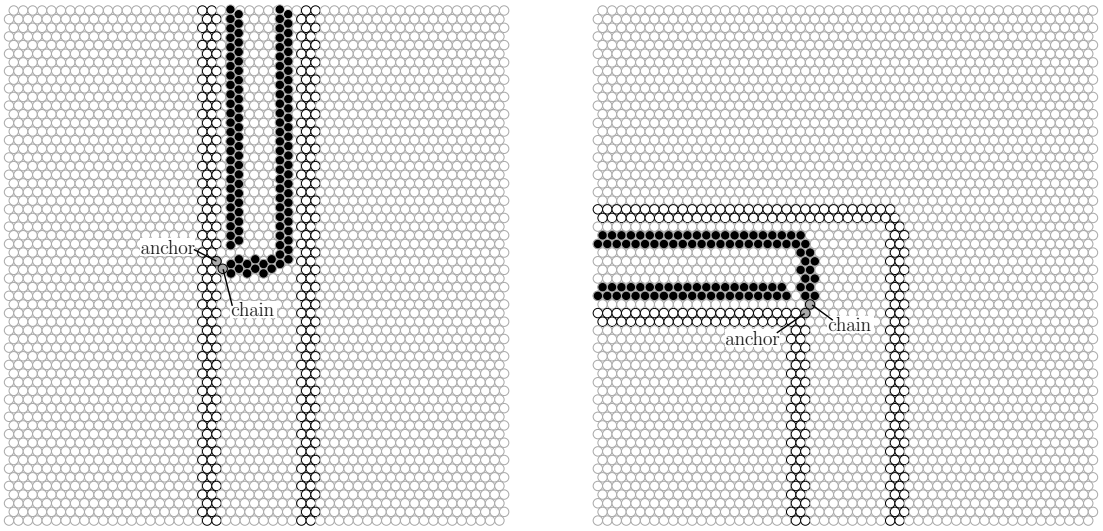
So far we have constructed the right side of the rectangle. In order to construct the top side, we actually have to leave our current cell grid. This is to be expected, as the rectangle is supposed to stick into the grid cell to the left or right in any embedding. We align the grid of another cell to the left of our current cell as suggested in Figure 2.9. We extend our notation and refer to the j-th disk in the i-th row of the left cell's grid as $D[i][j - X - 1]$. We add disks of the rows located two and three rows below the upper line segment such that they are connected to the previously created right side of the rectangle and such that the upper side sticks into the other cell by almost $(W - 2)/3$ regions. More precisely, we add the disks $D[\lceil W/2 \rceil Y/W - 3][j]$ and $D[\lceil W/2 \rceil Y/W - 2][j']$, where $3 - \lfloor ((W-2)/3)X/W \rfloor \leq j \leq \lceil X/2 \rceil$ and $3 - \lfloor ((W-2)/3)X/W \rfloor \leq j' \leq \lceil X/2 \rceil - 1$.

Constructing the left side of the rectangle works similar to the right side. We add the disks $D[i][2 - \lfloor ((W-2)/3)X/W \rfloor]$ and $D[i][3 - \lfloor ((W-2)/3)X/W \rfloor]$, for any $\lfloor W/2 \rfloor Y/W + 5 \leq i \leq \lceil W/2 \rceil Y/W - 4$. For the bottom part of the rectangle we add the disks $D[\lfloor W/2 \rfloor Y/W + 4][j]$ and $D[\lfloor W/2 \rfloor Y/W + 3][j']$ to our rectangle, where $2 - \lfloor ((W - 2)/3)X/W \rfloor \leq j \leq \lceil X/2 \rceil - 3$ and $3 - \lfloor ((W-2)/3)X/W \rfloor \leq j' \leq \lceil X/2 \rceil - 2$.

Note that we chose the rectangle to be as high as possible in the sense that only a single row of disks fits between the rectangle and the bottom line segment as well as between the rectangle and the top line segment. Of course, right now, this subgraph is not connected, therefore, there would be no need to place the two line segments parallel and this close to each other. Our high-level construction, however, will ensure that all the components are indeed placed (at least approximately) as intended. Since the corridor created by the two line segments is just barely higher than the rectangle (we will later define W and S such that the length of the corridor and the rectangle are much larger than the the height of 2 rows and, therefore, the corridor is much narrower than it seems in Figure 2.9) the rectangle's wiggle room is very small and it therefore has to be embedded either to the left or the right in the sense that the rectangle can be only be tilted negligibly. Imagine two of these unit disk touching graphs being placed next to each other. The disks realizing the line segments align and we define the cells' border disks to be touching in the obvious manner. The values for W and S will ensure that the rectangles' heights are close to the corridors' heights and, therefore, guarantee that if the right a rectangle is embedded to the left, the rectangle to its left also has to be embedded to the left and vice versa.

(d.2) The vertical wire gadget in detail

The schematics for the horizontal and the vertical wire gadgets look very similar. However, when constructing the vertical wire gadget as a unit disk touching graph, the hexagonal nature of the low-level grid causes some differences to the horizontal case. Consider the rectangle from the horizontal case and imagine rotating it by 90° to the right. In the horizontal case, the chain is located in row $\lfloor W/2 \rfloor Y/W + 2$ and the rectangle's topmost row is $\lceil W/2 \rceil Y/W - 2$, therefore, the rectangle encompasses $\lceil W/2 \rceil Y/W - 2 - (\lfloor W/2 \rfloor Y/W + 2) + 1 = (\lceil W/2 \rceil - \lfloor W/2 \rfloor) Y/W - 3 = Y/W - 3$ rows in total, which corresponds to a height of $(Y/W - 4)\sqrt{3} + 2$. The rotated rectangle now has a width of $(Y/W - 4)\sqrt{3} + 2$. A vertical corridor which contains this rectangle has to be at least $N = \lceil (Y/W - 4)\sqrt{3}/2 + 1 \rceil$ disks wide. We define the *anchor* disk for the vertical wire gadget to be $D[Y/2+1][\lceil (X-N)/2 \rceil]$, see Figure 2.10a. We attach the chain of the rectangle to the anchor. The line segments consist of the disks $D[i][\lceil (X - N)/2 \rceil - 1]$, $D[i][\lceil (X - N)/2 \rceil]$, $D[i][\lceil (X - N)/2 \rceil + N + 1]$ and $D[i][\lceil (X - N)/2 \rceil + N + 2]$ for any odd $1 \leq i \leq Y$ and disks $D[i][\lceil (X - N)/2 \rceil - 2]$, $D[i][\lceil (X - N)/2 \rceil - 1]$, $D[i][\lceil (X - N)/2 \rceil + N + 1]$ and $D[i][\lceil (X - N)/2 \rceil + N + 2]$ for any even $1 \leq i \leq Y$. The disks touch each other as dictated by the grid contacts. Note how two vertical wires placed on top of each other align since the number of rows Y is even.



(a) Vertical wire. (b) Corner.

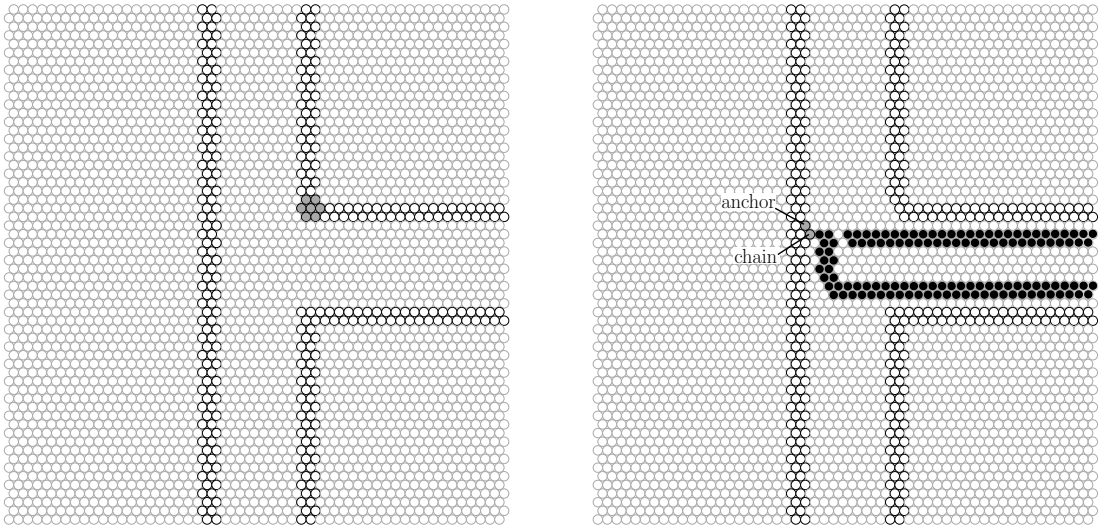
Figure 2.10.: Low-level construction of the gadgets with $W = 5$ and $S(|C|, |U|) = 3$.

(d.3) The corner gadget in detail

When designing the corner gadget, it is, once again, important that the line segments of horizontal wire gadgets placed to the left and vertical wire gadgets placed at the bottom align with the polylines of the corner gadgets. We define the *anchor* disk to be $D[\lfloor W/2 \rfloor Y/W + 1][\lceil (X - N)/2 \rceil]$, which is the bottom left disk of the central region, see Figure 2.10b. The horizontal parts of the polylines have to align with the line segments of horizontal wire gadgets and are therefore composed of the disks $D[\lfloor W/2 \rfloor Y/W][j]$ and $D[\lfloor W/2 \rfloor Y/W + 1][j']$ as well as $D[\lceil W/2 \rceil Y/W][j'']$ and $D[\lceil W/2 \rceil Y/W + 1][j'']$, for any $1 \leq j \leq \lceil (X-N)/2 \rceil - 1$, $1 \leq j \leq \lceil (X-N)/2 \rceil$ and $1 \leq j'' \leq \lceil (X-N)/2 \rceil + N + 1$. The vertical parts of the polylines consist of the disks $D[i][\lceil (X-N)/2 \rceil - 1]$, $D[i][\lceil (X-N)/2 \rceil]$, $D[i'][\lceil (X - N)/2 \rceil + N + 1]$ and $D[i'][\lceil (X - N)/2 \rceil + N + 2]$ for any odd $1 \leq i \leq \lfloor W/2 \rfloor Y/W - 1$ and any odd $1 \leq i' \leq \lceil W/2 \rceil Y/W$ and disks $D[i][\lceil (X - N)/2 \rceil - 2]$, $D[i][\lceil (X - N)/2 \rceil - 1]$, $D[i'][\lceil (X - N)/2 \rceil + N + 1]$ and $D[i'][\lceil (X - N)/2 \rceil + N + 2]$ for any even $1 \leq i \leq \lfloor W/2 \rfloor Y/W - 1$ and any even $1 \leq i' \leq \lceil W/2 \rceil Y/W$. Following the construction for the horizontal case, we construct a a rectangle attached to the anchor and sticking out of the cell by $(W - 2)/3$ regions. Note that this rectangle is approximately half a region shorter than the previous rectangles since the anchor is located not in the center of the central region, but in its bottom-left corner.

(d.4) The clause, variable and stopper gadgets in detail

Following our previous elaborations, it should, for the most part, be clear how to construct the clause gadget. We therefore omit most of the construction and explain only two new aspects. (1) Both the corner in the bottom-left cell as well as the 'T-type' cell in the middle row of the clause gadget require the construction of a subgraph that realizes top-to-right turn. As depicted in Figure 2.11a, the obvious construction method for this kind of turn results in a non-outerplanar subgraph due to Y being even and a multiple of W. We therefore 'round' the subgraphs that realizes these turns as illustrated in Figure 2.11b. As a result, choosing an anchor disk for a top-to-right corner gadget is not as straight-forward as for the left-to-bottom corner gadget introduced in (c.4). We will resolve this issue after explaining the second new aspect. (2) The 'T-type' cell in the middle row requires a construction that allows orienting the rectangle to the top, the bottom and the right. We therefore define the *anchor* disk to be $D[\lceil W/2 \rceil Y/W - 1][\lceil (X-N)/2 \rceil]$, which is located in the top-left of the central region, and we modify the rectangle as follows. So far, the chain



(a) Not outerplanar. (b) Outerplanar.

Figure 2.11.: Low-level construction of the 'T-type' cell of the clause gadget with $W = 5$ and $S(|C|, |U|) = 3$.

always touched two disks of the rectangle and therefore formed a rigid structure together with the rectangle. We remove the adjacency to the inner of these two disks (which is the left of the disks in the original construction), so that the chain is now only connected to one disk as depicted in Figure 2.11b. As before, the chain is the only disk that touches any of the polylines that describe the remainder of the structure and the only touched disk is the anchor. Clearly, we can still flip the rectangle at the chain so that it can be oriented to the bottom or the top, however, we can now also rotate the rectangle at the chain such that it can be oriented to the right. We can use the same anchor disk and the same modification to the rectangle's chain in order to solve the previously mentioned 'rounding' issue for the top-to-right corner in the bottom row of the clause gadget.

Recall that in the variable gadget the corridor in which the rectangle is embedded is much narrower and that the rectangle's height is reduced accordingly. This is the only new aspect when it comes to the construction of the variable and stopper gadgets. We therefore describe how the central cell is constructed, the remaining cells can be constructed accordingly. The bottom row of disks of the top row of regions as well as the row of disks below these disks constitute the upper line segment, see Figure 2.7. Independent of W and S, the corridor in this cell is exactly 4 rows high and the rectangle, accordingly, is 2 rows high and, therefore, is just a rigid line segment itself. It is anchored in the middle of the bottom line segment as usual.

(d.5) The crossing gadget in detail

Even though the crossing gadget is the most complex gadget, its low-level construction should be straight-forward considering our previous elaborations. The only new aspect is the anchoring for the four-way embeddable 'notched squares'. Figure 2.12 illustrates how this works.

(e) Conclusion

For each cell of the high-level grid R we have created a graph. We now merge all these graphs together by adding adjacencies between the vertices corresponding to 'border' disks of adjacent cells. Figure 2.13a depicts a schematic view of the obtained graph. The black lines represent the rigid, connected components, omitting the rectangles and 'notched squares'. Note that if there exist sets of clauses with disjoint variable sets, it is possible

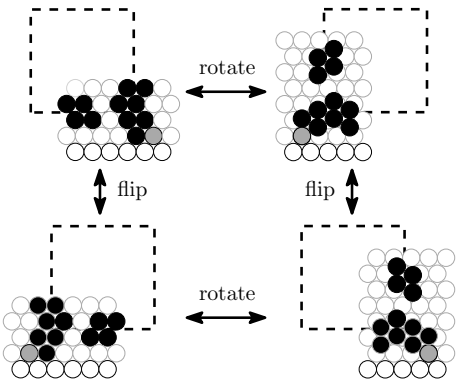


Figure 2.12.: Anchoring for the four-way embeddable 'notched squares'.

that the obtained graph actually consists of multiple such structures. However, without loss of generality, we may assume that this is not the case. Nevertheless, we immediately notice two problems. (1) Even though each of the gadget subgraphs itself is clearly outerplanar, the graph obtained by merging all of the subgraphs is not outerplanar. Right now it has an outer face (white), a 'wire' face (lightgray) and several 'other' faces (darkgray) enclosed by rigid structures. (2) There is no need to embed the rigid structures that enclose the 'other' faces where they are located in Figure 2.13a since the graph is not connected and each of the rigid structures constitutes a separate connected component.

We solve both problem (1) and problem (2) by introducing a new gadget called *maintenance section*, a low-level version of which is illustrated in Figure 2.13c. The maintenance section gadget is basically a modified horizontal wire gadget. We 'cut' the bottom line segment by removing disks $D[\lfloor W/2 \rfloor Y/W + 1][\lceil X/2 \rceil - 2]$ and $D[\lfloor W/2 \rfloor Y/W][\lceil X/2 \rceil - 2]$. Recall that N is the minimum number of disks that have to be placed next to each other such that the total length of the thereby obtained path exceeds the height of the rectangle including the chain. We connect the right part of the bottom line segment with the upper segment by adding a path consisting of $N + 2$ disks connecting anchor disk $D[\lfloor W/2 \rfloor Y/W + 1][\lceil X/2 \rceil - 2]$ and disk $D[\lceil W/2 \rceil Y/W][\lceil X/2 \rceil - 2]$. We call this path *bridge*. We delete the old chain disk. The rectangle excluding the chain is always an even number N' of disks high and on the left and right side the disks are in turn offset by $1/2$. On the right side, the outer disk of the $(N'/2 + 1)$-th row of the rectangle belongs to the rightmost disks of the rectangle. To this disk we attach a new *chain* disk. We also connect the chain disk to the $\lceil (N + 2)/2 \rceil$-th (counted from the anchor) disk of the bridge. For each of the rigid structures that enclose an 'other' face, we remove the top-left horizontal wire gadget and replace it with a maintenance section. The 'wire' face and the 'other' faces thereby collapse to one large face, see Figure 2.13b. By furthermore 'cutting' the rigid structure that separates this face from the outer face once at some arbitrary location, we obtain an outerplanar graph (1).

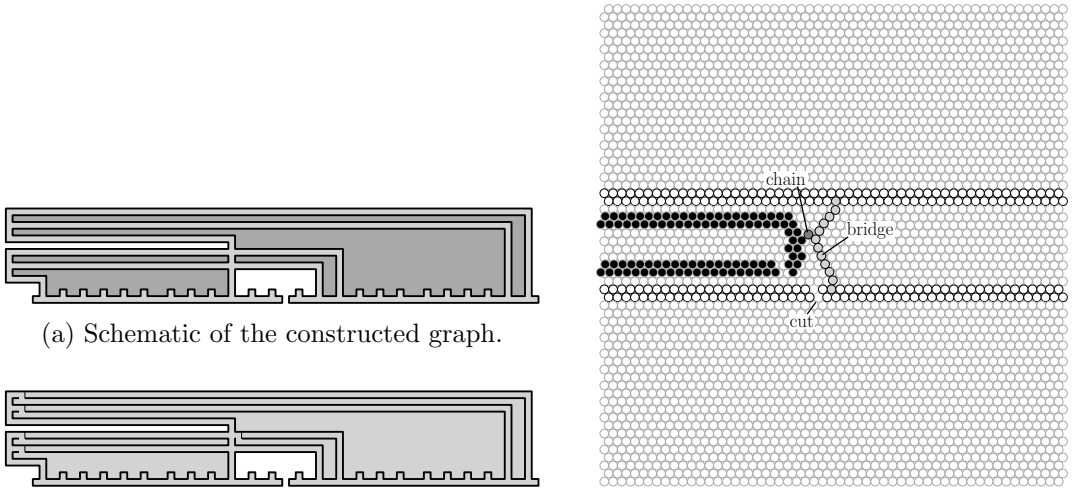
Note that since each of the rigid structures is 'cut' at exactly one location they remain rigid. In fact, now they satisfy the preconditions of Lemma 2. The bridges, however, are not rigid. It therefore is possible to move the rigid structures, which are now held together by the bridges, around, and we need to argue why the rigid structures are nevertheless placed at least approximately like in the *ideal* situation and, therefore, that it is not possible to embed the obtained graph if this is not actually supposed to be possible. First of all, consider only the rigid structure s_w that bounds the 'wire' face and the rigid structure s_{11} that bounds the top-left 'other' face. Note that the top side of s_{11} points to the top in any embedding since the length of the bridge is only a W-th of a cell's width, which limits the rotatability of s_{11}. Furthermore, the combinatorial embedding of the bridge, s_w and s_{11} (without the rectangles) has to be ideal in the sense that it is not possible to flip s_{11} such

that the side that is ideally pointing the right is pointing to the left since the maintenance section with the bridge has replaced the top-left horizontal wire gadget of s_{11} and the width of any 'other' face bounding rigid structure is at least 2 cells (note how the stoppers and vertical wire gadgets connected to the variable gadgets are separated by 2 free cells). Now consider the rectangles attached to s_{11}, s_w and the bridge connecting them. Ignoring rotation, we can move s_{11} at most roughly 2 disk widths to the left or to the top since the corridors containing the rectangles are ideally just 2 rows or columns of disks wider than the rectangles. Next, consider the rigid structure s_{21} that is ideally located directly beneath s_{11}. Again, the combinatorial embedding of s_w, s_{11}, s_{21} and the bridges has to be ideal since flipping s_{21} is not possible. Ignoring rotation, s_{21} can only be moved roughly 2 disk widths to the left. Assuming that Y and X, which are the dimensions of a cell, are much larger than 2 disks, it furthermore is only possible to move s_{21} roughly $2 + 2$ disk widths to the top from its ideal position (which requires that s_{11} is moved 2 disks to the top as well). Directly beneath s_{21} is the rigid structure s_w, therefore, ignoring rotation, the movement to the bottom for both our rigid structures is also at most roughly 2 or 4 disk widths respectively. The same holds true for all the left-most 'other' face bounding rigid structures. To the right of s_{11} we always find s_w, therefore s_{11} can also be moved at most 2 disks to the right.

Now consider the top-left rigid structure s_{ij} of the not yet considered 'other' face bounding rigid structures. The combinatorial embeddings of the rigid structures that are ideally located above and to the left of s_{ij} are ideal, therefore, the combinatorial embedding of s_{ij} and its bridge are ideal as well. It can only be moved a number of disk widths to the left or the top that is linear in the number of the previously considered 'other' faces. We can argue that s_{ij} can only be moved a number of disks width to the bottom that is linear in the number of not yet considered rigid structures below s_{ij}. A similar argument holds for the movement to the right. If the cell dimensions are large enough, we can, therefore, iteratively conclude that the position of any disk can only diverge at most a number of disk widths linear in the number of 'other' faces from the ideal position. This holds true even if rotations are allowed since in the corridors above and below each of the rigid structures each contain at least 2 rectangles and the corridors to the sides of each rigid structure contain at least 1 rectangle. It should be noted that rigid structures whose bottom parts are located in variable gadgets could be moved a little bit further to the bottom than just 2 disk widths because variable gadget utilize only one long rectangle. If the long rectangle is embedded to the right (left), the rigid structures on the left (right) of the variable gadget can be moved all the way down to the bottom of the corridor. However, this does not cause any problems since we created the long rectangle to be only 2 disks high. Therefore, the width of the entire corridor is only 4 disk widths.

All of our gadgets are designed such that if something is not supposed to be embeddable, some rigid structures overlap by at least one region of the $W \times W$ mid-level grid R_W. The dimensions of a cell are $Y = 4 \cdot W \cdot S(|C|, |U|)$ and $X = \lceil (Y-1)\sqrt{3}/2 + 1 \rceil$. The dimensions of a region of the mid-level grid R_W are therefore $4 \cdot S(|C|, |U|)$ and roughly $\lceil (4 \cdot S(|C|, |U|) - 1)\sqrt{3}/2 + 1 \rceil$. We have already established that disk positions can only diverge at most a number of disk widths linear in the number of 'other' faces from their ideal position in any direction (if the cells dimensions are large enough). Since the number of 'other' faces is clearly polynomial in $|C|$ and $|U|$, we can choose S as a polynomial in $|C|$ and $|U|$ in order to enlarge the dimensions of regions (and, therefore, cells) such that unexpected embeddings are still not possible even if disk positions are allowed to diverge from their ideal positions.

Recall that auxiliary multigraph G' is orientable if and only if C is satisfiable. We have constructed our UDT instance G by using gadgets that emulate the orientation concept. Clearly, if G' is orientable, then G is realizable as a unit disk touching graph. On the other



(a) Schematic of the constructed graph.

(b) The constructed graph after adding the maintenance sections.

(c) Low-level maintenance section with $W = 5$ and $S(|C|, |U|) = 3$.

Figure 2.13.: Replacing the top-left horizontal wire of each 'other' face makes the 'wire' and the 'other' faces collapse.

hand, if G is realizable as a unit disk touching graph, then each clause gadget has at least one horizontal wire that is oriented to the right and all vertical wires of at least one side of each variable gadget are oriented to the top. The horizontal wire (crossing) gadgets are designed such that it is not possible that both the gadgets to the right and left are oriented towards it. The vertical wire (crossing) gadgets are designed such that it is not possible that both the gadgets to the top and bottom are oriented towards it. Finally, the corner gadgets are designed such that it is not possible that both the gadgets to the left and bottom are oriented towards it. Thus, the orientation of the gadgets is propagated as intended, which induces an orientation for the corresponding edges in G'. Note that the gadget orientations do not necessarily induce an orientation for all edges of G', since there may exist an edge e in G' such that two consecutive gadgets of the succession of gadgets that correspond to e are oriented away from each other. This, however, can happen only once per edge and it does not cause a problem since we are still guaranteed to obtain at least a partial orientation for the edges in G' in which each clause vertex has outdegree at least 1 and in which at least one of the vertices of each literal pair has indegree 0. The orientation for the remaining edges can be chosen arbitrarily and, therefore, G is realizable as a unit disk touching graph if and only if G' is orientable, which on the other hand is the case if and only if C is satisfiable. The number of cells of the high-level grid as well as the number of disks used to realize each of these cells both are polynomial in the input size, which concludes our proof. □

2.2. Unit Disk Touching Graph Recognition with fixed Embedding

In this section we consider the Unit Disk Touching Graph Recognition with fixed Embedding (UDTE) problem. Recall that in this problem we need to decide whether a given graph can be realized as a unit disk touching graph that respects a given combinatorial embedding. Our main result is an extension for the $\mathcal{NP}$-hardness proof from the previous section that shows that UDTE is $\mathcal{NP}$-hard and remains so even if the input graph is outerplanar. We begin the section by describing how some of the results from the UDT problem carry over to the UDTE problem.

Corollary 3. *Any path P can be realized as a unit disk touching graph with respect to any combinatorial embedding for P.*

Corollary 4. *Let $C = (V, E)$ be a cycle and Γ be a combinatorial embedding for C. There exists a realization of C as a unit disk touching graph with respect to Γ.*

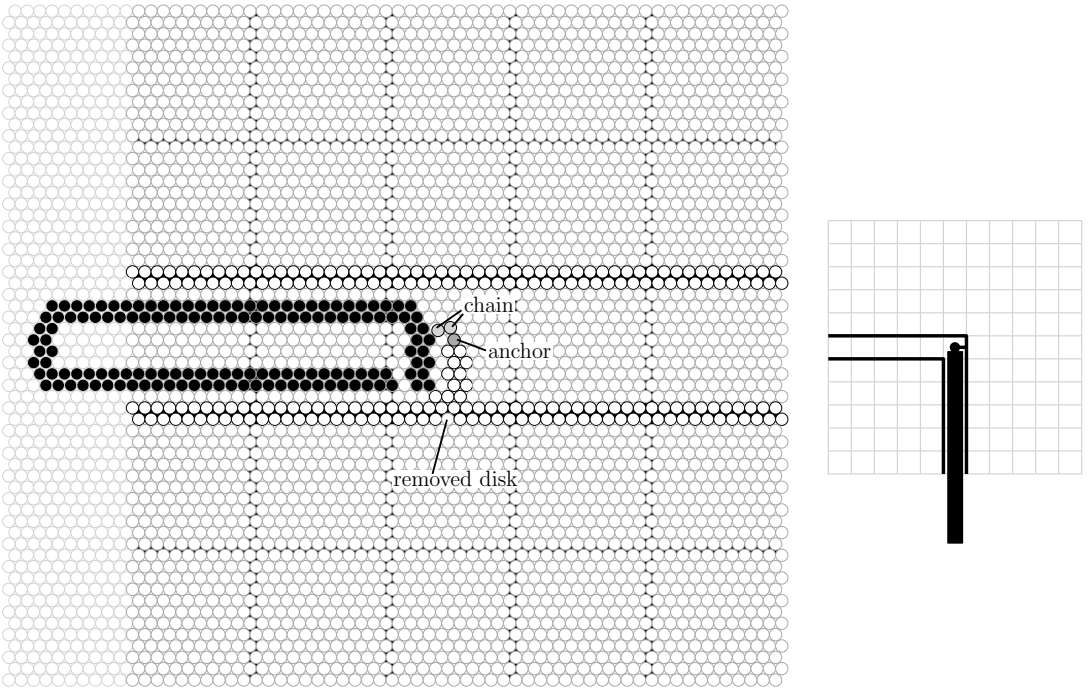
Corollary 5. *For spiders the Unit Disk Touching Graph Recognition with fixed Embedding problem can be decided in $O(n)$ time where n is the number of vertices of the given spider. In the Real RAM model a realization can be constructed in $O(n)$ time (if one exists).*

Corollary 3 and Corollary 4 follow from Observation 1 and Observation 2 since the combinatorial embedding of any path or cycle is unique. Corollary 5 follows from Observation 3. In the remainder of this section we show that UDTE is $\mathcal{NP}$-hard and remains so even if the input graph is outerplanar.

Theorem 3. *The Unit Disk Touching Graph Recognition with fixed Embedding problem is $\mathcal{NP}$-hard, even for outerplanar graphs.*

Proof. This proof is an adaption of the proof of Theorem 2. We, therefore, recycle the definitions and notations and assume that the reader is familiar with said proof. Recall that in Theorem 2 we construct a graph G that is realizable as a unit disk touching graph if and only if the auxiliary multigraph G' is orientable, which, on the other hand, is the case if and only if the collection of clauses C of the 3SAT instance is satisfiable. The graph G is constructed by placing different types of gadgets on a high-level grid R. The gadgets are schematically designed on a mid-level $W \times W$ square grid R_W and constructed in detail on a hexagonal circle grid D. The gadgets emulate the orientation concept of the auxiliary multigraph G' and the key tool for this purpose are rectangles and other rigid structures that can only be embedded in certain ways due to our anchoring concept. Many of the gadgets from Theorem 2 are designed such that different gadget orientations correspond to different combinatorial embeddings. For example, in a horizontal wire gadget that is oriented to the right, the rectangle is embedded to the right. Orienting the gadget to the left requires flipping the rectangle to the left and thereby changing the combinatorial embedding. The idea for this proof is to adapt the gadgets such the different orientations require rotating certain rigid structures rather than flipping them such that all possible realizations share one common combinatorial embedding.

We begin by adapting the horizontal wire gadget. Instead of defining one of the disks of the bottom line segments to be the anchor disk and using one chain disk to attach the rectangle, we use a rigid structure that extends from the bottom line segment in order to place the anchor disk in the center of the central region of the gadget's cell, see Figure 2.14a. Note that we need to remove one of the disks from the bottom row of the bottom line segment in order to maintain outerplanarity. We use two chain disks to connect the anchor to the middle the rectangle's side. This way, the rectangle can be rotated around the anchor



(a) Low-level construction of the horizontal wire gadget with $W = 5$ and $S(|C|, |U|) = 3$.

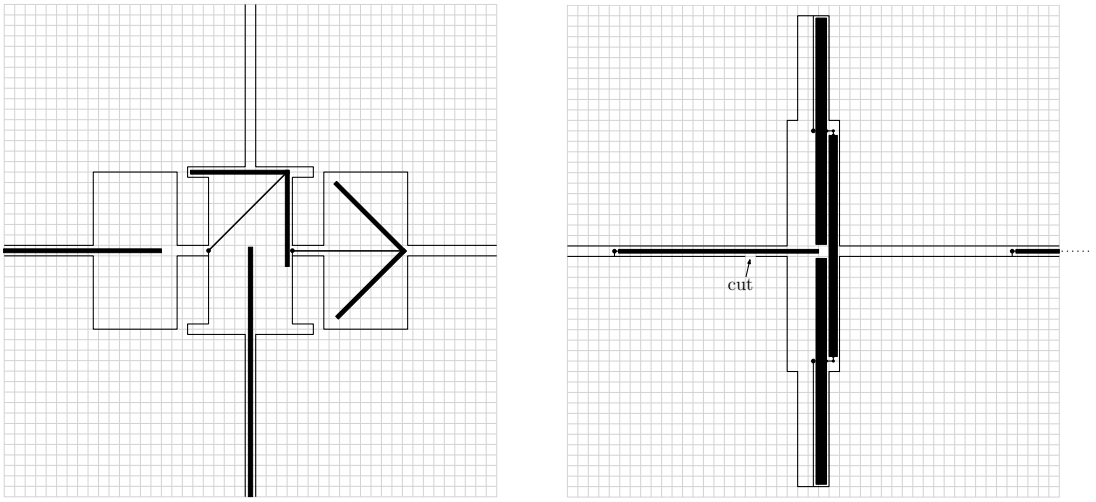
(b) Schematic of the corner gadget, $W = 11$.

Figure 2.14.: Gadget adaptations that do not require flipping the rectangles.

to either side without changing the combinatorial embedding. The same approach can be applied to adapt the vertical wire gadget. In the adaptation for the corner gadget, the extending rigid structure is attached to the right polyline, see Figure 2.14b. With this information, the adaptations for the clause, stopper and variable gadgets should be straight-forward.

The crossing gadget has to be redesigned completely since the four-way embeddable 'notched squares' require flips as well as rotations. Figure 2.15a depicts our new crossing gadget. The main ingredients are two 'arrow'-type rigid structures. On either side of the gadget there are large cavities and the 'arrow'-type structures are anchored such that the one on the right (left) can be embedded in the cavity to the right (left) if and only if the gadget to the right (left) of the crossing is oriented away from the crossing. However, if this is not the case, it has to be embedded in a cavity in the center of the crossing gadget. The 'arrow'-type structures are designed such that only one of them can be embedded in the middle cavity, therefore, it is not possible that both the gadget to the right and to the left of the crossing are oriented towards it. An 'arrow'-type structure can be embedded in the middle cavity in two different ways. It can be pointed to the top or the bottom. If the gadget below (above) the crossing gadget is oriented towards it, it has to be pointed to the top (bottom). Even if both 'arrow'-type structures are pointed to the sides, it is not possible for both the gadgets above and below the crossing gadget to be oriented towards it, since the rectangles would intersect in the central region of the cell. In order to realize the rigid 'arrow'-type structure as a unit disk touching graph, the diagonal 'shaft' of the arrow can be approximated by a zigzagging, rigid polyline.

The last remaining gadget is the maintenance section, which has to be redesigned as well since embedding the rectangle to different sides of the bridge corresponds to different combinatorial embeddings. Figure 2.15b depicts the new maintenance section gadget. We place a tall but narrow cavity in the middle of the gadget. From the top and the bottom we extend rigid structures in order to place anchor disks in the center of both the lower



(a) Crossing gadget. (b) Maintenance gadget.

Figure 2.15.: Mid-level schematics for the adapted gadgets with $W = 47$.

and the upper half of the gadget. We connect both anchor disks to the middle of two tall but narrow rectangles that fit tightly in the cavity. To the other sides of these rectangles we connect yet another rectangle using chains and two rigid structures each. Note that by rotating a rectangle around its anchor we can place it either to the left or the right side, however, since the cavity is so narrow and the rectangles are so tall, they always have to be embedded perpendicularly. Because of the third rectangle, all three rectangles have to be embedded either to the right or to the left side of the gadget. Observe that because of our anchors and chains, both possibilities share one combinatorial embedding. On the sides of the gadget we place two more rectangles that have to be embedded sticking out of the gadget if the three rectangles in the middle are embedded to the corresponding side. Note that we have to modify all clause gadgets such that their rectangles stick only 4 regions into adjacent maintenance gadgets. This can be done by moving the rectangles' anchoring to the left and by decreasing the rectangles' lengths.

We have adapted all gadgets such that they emulate the orientation concept using only rotations instead of flipping. Thus, we can provide a combinatorial embedding Γ for the generated graph G such that there exists a realization of G as a unit disk touching graph that respects Γ if and only if the auxiliary multigraph G' is orientable and therefore if and only if C is satisfiable. □

3. Recognition problems with fixed radii

In this chapter we consider the Disk Touching Graph Recognition with fixed Radii (DTR) (Section 3.1) and the Disk Touching Graph Recognition with fixed Radii and Embedding (DTRE) (Section 3.2)problems. Recall that in these problems we need to decide whether a given graph can be realized as a disk touching graph that respects a given radius assignment (and a given combinatorial embedding in case of the DTRE). Furthermore, recall that the $\mathcal{NP}$-hardness of the DTR is implied by Breu and Kirkpatricks $\mathcal{NP}$-hardness proof [BK98] for the Unit Disk Touching Graph Recognition (UDT) problem and that we strengthened their result in Chapter 2 by showing that the UDT problem as well as the Unit Disk Touching Graph Recognition with fixed Embedding (UDTE) problem are $\mathcal{NP}$-hard, even for outerplanar graphs. These results carry over to the DTR and the DTRE problems. In Section 3.1 we strengthen our result for the DTR problem further by showing that the problem remains $\mathcal{NP}$-hard even for stars. In Section 3.2, however, we devise an algorithm that solves the DTRE problem for stars in linear time in the Real RAM model. For both problems we also provide existence arguments for realizations of paths and cycles.

3.1. Disk Touching Graph Recognition with fixed Radii

In this section we consider the Disk Touching Graph Recognition with fixed Radii (DTR) problem and show that it remains $\mathcal{NP}$-hard even for stars. First, however, we provide a simple observation concerning paths and proceed with an existence argument for realizations of cycles.

Observation 4. *Any path P can be realized as a disk touching graph with respect to any radius assignment for P.*

Theorem 4. *Let $C = (V, E)$ be a cycle and $r : V \to \mathbb{R}^+$ be a radius assignment for C. There exists a realization of C as a disk touching graph with respect to r.*

Proof. We use a geometric construction to prove the existence of a disk touching graph with the property that the touching points of all pairs of consecutive disks in our cycle are located on a circle and therefore all of these touching points have a specific distance $r_c \in \mathbb{R}^+$ to some *center* point $p_c \in \mathbb{R}^2$. Let D_1, D_2 and D_3 be three consecutive disks in this construction, let $p_2 \in \mathbb{R}^2$ and $r_2 \in \mathbb{R}^+$ be the center and the radius of D_2 respectively and let $p_{1,2}, p_{2,3} \in \mathbb{R}^2$ be the touching points of D_1 and D_2 and D_2 and D_3. Using the Pythagorean Theorem we can describe the distance from p_c to p_2 as $d_{c,2} = \sqrt{r_c^2 + r_2^2}$,

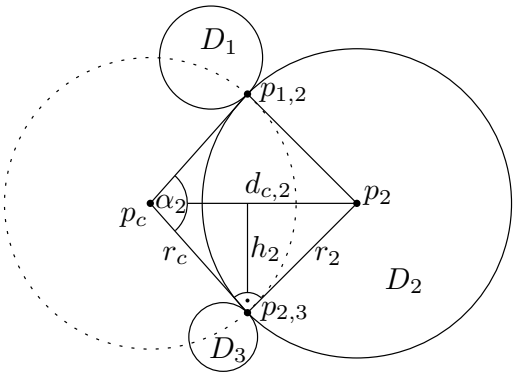


Figure 3.1.: Three consecutive disks in the geometric construction used in Theorem 4.

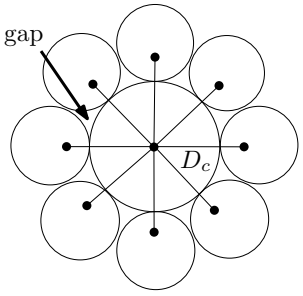


Figure 3.2.: In Theorem 5, Theorem 6 and Theorem 7 n funnel-shaped gaps of equal size are created by placing n outer disks tightly around a central disk D_c.

see Figure 3.1. The line segments $\overline{p_c p_2}$, $\overline{p_2 p_{2,3}}$ and $\overline{p_{2,3} p_c}$ constitute a rectangular triangle with height $h_2 = r_2 \cdot r_c / d_{c,2}$. We define angle $\alpha_2 = \angle(\overline{p_c, p_{1,2}}, \overline{p_c, p_{2,3}})$. By using basic trigonometry we obtain $2h_2 = 2r_c \sin(\alpha_2/2)$ and therefore $\alpha_2 = 2 \arcsin(h_2/r_c)$.

The statement above is true for any three consecutive disks of our cycle. Therefore, we know that the value r_c has to satisfy the following condition, in which h_i is defined according to our example for any $1 \leq i \leq |V|$:

$$\sum_{1 \leq i \leq |V|} 2 \arcsin(h_i / r_c) = 2\pi \tag{3.1}$$

For any $1 \leq i \leq |V|$ we also define r_i and $d_{c,i}$ according to our example and obtain $h_i/r_c = r_i/d_{c,i} = r_i/\sqrt{r_c^2 + r_i^2}$. Clearly $\lim_{r_c \to 0} r_i/\sqrt{r_c^2 + r_i^2} = 1$ and $\lim_{r_c \to +\infty} r_i/\sqrt{r_c^2 + r_i^2} = 0$. Since arcsin is continuous at any point of the interval $(0, 1)$ the intermediate value theorem states that there indeed exists a value $r_c \in (0, \infty)$ for which Condition 3.1 holds true, which concludes our proof. □

In the remainder of this section we prove that the DTR is $\mathcal{NP}$-hard even for stars. To prepare the reader, we decided to include $\mathcal{NP}$-hardness proofs for outerplanar graphs and for trees as well since they are very intuitive, follow ideas similar to those used in the proof for stars and delineate the therein occurring problems.

Theorem 5. *The Disk Touching Graph Recognition with fixed Radii (DTR) problem is $\mathcal{NP}$-hard even for outerplanar graphs.*

Proof. Note that Theorem 2 immediately implies that DTR is $\mathcal{NP}$-hard even for outerplanar graphs. As explained above, we nevertheless decided to include this particular

theorem as the proof follows roughly the same idea as the proof for Theorem 7 and, therefore, prepares the reader for the upcoming final theorem of this section. However, since our hypothesis is implied by Theorem 2 we feel justified in omitting some technical details and instead just present a proof sketch that focuses on intuition.

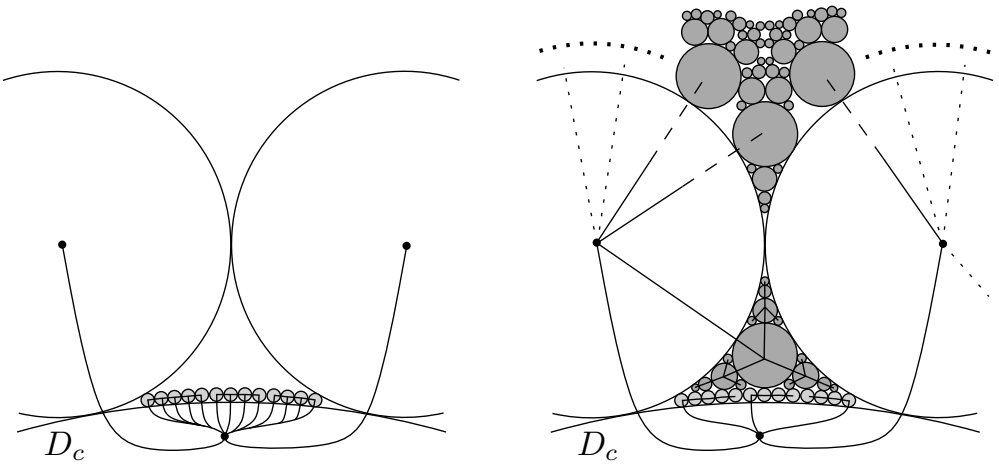
We perform a polynomial-time reduction from the 3-Partition problem. Let (A, B) be a 3-Partition instance, where A is the set of positive integers with $|A| = 3n, n \in \mathbb{N}$ and $B \in \mathbb{N}$ is the bound. We have to construct a graph $G = (V, E)$ as well as a radius assignment $r : V \to \mathbb{R}^+$ such that G has a realization as a disk touching graph with respect to r if and only if (A, B) is a yes-instance of the 3-Partition problem.

We start by creating a *central* disk D_c as well as a total of n *outer* disks adjacent to D_c. We choose the radius of the outer disks uniformly such that in any realization they have to be placed close next to each other (the last paragraph of this proof sketch explains this notion in more detail), thus, creating n funnel-shaped *gaps* of equal size, see Figure 3.2. We determine a disk radius $r_i \in \mathbb{R}^+$ such that exactly B disks with radius r_i fit tightly inside a single of the gaps while touching the central disk D_c. For each integer $a \in A$ we now create a gadget consisting of a total of a disks $D_1^a, \ldots, D_a^a$ with radius r_i such that D_i^a is adjacent to D_{i+1}^a for $1 \leq i \leq a-1$ and additionally such that each $D_i^a, 1 \leq i \leq a$ is adjacent to the central disk D_c, see Figure 3.3a. Consider a triple $t = (a_1, a_2, a_3)$ of elements of A. The triple t corresponds to a triple of disk gadgets with a total of $a_1 + a_2 + a_3$ disks. If t is feasible and, therefore, $a_1 + a_2 + a_3 = B$, then an algorithm that tries to realize our graph G with respect to r can place the three corresponding disk gadgets all together in one gap since r_i is chosen such that up to exactly B disks with radius r_i can be placed in a gap. However, if t is infeasible and, therefore, $a_1 + a_2 + a_3 > B$, only two of the gadgets can be placed together in one gap. The remaining gadget has to be placed in another gap entirely since its disks are consecutively adjacent. We can conclude that the constructed graph G can be realized as a disk touching graph with respect to r if and only if (A, B) is a yes-instance of the 3-Partition problem. Note that since 3-Partition is $\mathcal{NP}$-hard in the strong sense, this is indeed a polynomial-time reduction, even tough we create a total of $nB + n + 1 \in O(nB)$ disks. The constructed graph G is outerplanar, which concludes this proof sketch except for the precision problem described in the following paragraph.

Note that by choosing one of the three different occurring radii in this construction we uniquely determine the other two radii. They can be calculated using trigonometry and the Pythagorean Theorem, which involves computing square roots as well as sine and cosine functions and, therefore, can take a superpolynomial, even an infinite amount of time. For a complete proof, we would need to argue that either all coordinates are good-natured or that approximate values suffice. Furthermore, note that we actually do not want to construct a tight packing. Instead, the outer disks as well as the gadget are not actually supposed to touch each other. This could be achieved by ensuring that the central disk's radius is a tiny value ε larger than it would be in a tight packing. Note, however, that whether value ε is small enough depends on the input since the larger central disk allows the outer disks to move around, thereby, increasing a gaps possible size and for larger B the largest tolerable gap size decreases. □

Theorem 6. *The Disk Touching Graph Recognition with fixed Radii problem is $\mathcal{NP}$-hard even for trees.*

Proof. With Theorem 7 we provide an in-depth proof that shows that the DTR problem remains $\mathcal{NP}$-hard even for stars, which, of course, implies that this is true for trees as well. As explained above, we nevertheless decided to include this particular theorem as the proof follows roughly the same idea as the proof for Theorem 7 and, therefore, prepares



(a) A gap in Theorem 5. (b) A gap in Theorem 6.

Figure 3.3.: Comparison of the gaps in Theorem 5 and Theorem 6. The central and the outer disks are colored white, the gadgets representing the input integers are colored lightgray and the plug disk subtrees are colored darkgray.

the reader for the upcoming final theorem of this section. However, since our hypothesis is implied by Theorem 7 we feel justified in omitting some technical details and instead just present a proof sketch that focuses on intuition.

We perform a polynomial-time reduction from the 3-Partition problem. The first part of the proof is analogous to the proof of Theorem 5. We create a *central* disk D_c and n *outer* disks that are adjacent to D_c and which have to be placed close together (the last paragraph of Theorem 5 explains this notion and related problems) around D_c, thus creating n funnel-shaped equal-sized *gaps*. Again, for each integer $a \in A$ we now create a gadget consisting of a total of a disks $D_1^a, \ldots, D_a^a$ with radius r_i, which is chosen such that up to exactly B disks can be placed together in one of the gaps while touching the central disk D_c. Recall that we have to ensure that all of these disks have to be placed together in one of the gaps. In Theorem 5 we 'tied' them together by making them touch the central disk as well as each other consecutively. Since now our constructed graph has to be a tree, we have to find a different solution. We still make all disks corresponding to a consecutively adjacent, however, we only make the first disk D_1^a of the sequence adjacent to the central disk D_c. Disks $D_2^a, \ldots, D_a^a$ can now be placed anywhere in the gap, in particular, they do not have to be placed near the central disks. Therefore, it might be possible to place more than three gadgets together in a single gap.

The idea to solve this problem is to attach trees consisting of *plug* disks to the outer disks that have to be placed in the gaps and which take up almost all the remaining space in the gap and, therefore, ensure that our gadgets still have to be placed tightly next to the central disk D_c. This idea is illustrated in Figure 3.3b. In particular, the disks of the attached tree are the recursively largest possible disks that can be placed in the remainder of the gap. In a full proof, one needs to precisely argue that a total of $O(nB)$ plug disks is sufficient. Intuitively this is the case since B is also the upper bound for the number of disks that can be placed in a single gap while being placed next to the central disk D_c. In order to ensure that the attached trees with the plug disks are placed inside the gaps, we attach some additional disks (for example additional copies of the plug disk trees as illustrated in Figure 3.3b) to the outer disks. By filling the remaining space in the gaps with plug disks, we have ensured that, like in the proof of Theorem 5, only gadgets corresponding to feasible input triples can be placed together in a single gap and, therefore, that the constructed graph, which is a tree, can be realized by a disk touching graph with respect

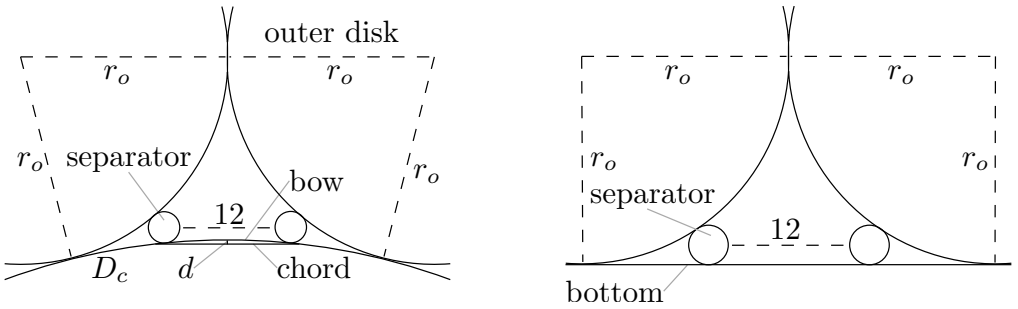
to the construction radius function if and only if the 3-Partition instance is a yes-instance. Since we only create $O(nB)$ disks and 3-Partition is $\mathcal{NP}$-hard in the strong sense, this concludes our proof sketch. □

Theorem 7. *The Disk Touching Graph Recognition with fixed Radii problem is $\mathcal{NP}$-hard even for stars.*

Proof. Like in the $\mathcal{NP}$-hardness proof sketches for trees (Theorem 6) and outerplanar graphs (Theorem 5), we perform a polynomial-time reduction from the 3-Partition problem. Note that in order to increase readability certain portions of this proof have been moved to detail Sections (a) – (d) at the end of the proof as well as Appendix A. Let (A, B) be a 3-Partition instance, where A is the set of positive integers with $|A| = 3n, n \in \mathbb{N}$ and $B \in \mathbb{N}$ is the bound. We have to construct a star S as well as a radius assignment r' for S such that G has a realization as a disk touching graph that respects r' if and only if (A, B) is a yes-instance of the 3-Partition problem. For this reduction we shall create several disks that are all adjacent to a *central* disk D_c. Since S is supposed to be a star, these are the only adjacencies in our construction. However, several of the disks adjacent to D_c are required to be placed very close together without actually touching. We shall, whenever we need to calculate distances, handle these barely not touching disks as if they are actually touching. In Section (d) at the end of this proof, we describe how to actually compute these distances using approximations and during this step the radius of the central disk increases by a suitably small amount such that no unanticipated embeddings can be created.

As in the previous proofs, we create a total of m *outer* disks with uniform radius r_o chosen such that in any embedding these disks have to be placed close together around the central disk, thus creating m funnel-shaped *gaps* of equal size. Each of the gaps is supposed to be large enough that disks that represent a feasible input triple can be placed inside it, however, the gaps should be too small to contain an infeasible integer triple's disk representation. Unlike in the previous proofs, since our disk graph is supposed to be a star, we can not represent an integer $a \in A$ by a gadget consisting of a disks with some uniform radius since we can not use adjacencies to ensure that the gadget's disks have to be placed in the same gap. Instead we will represent each integer in A by a single disk and encode its value in its disk's radius. Each of the disks that represents one of the integers will be refered to as an *input* disk. A *feasible* (*infeasible*) disk triple is a triple that represents a feasible (infeasible) integer triple.

In the *original* scenario described above, a gap's boundary belonging to the central disk D_c, which we call the gap's *bow*, is curved as illustrated in Figure 3.4a. We will, however, first consider a *simplified* scenario in which a gap is created by placing two disks of radius r_o right next to each other on a straight line as depicted in Figure 3.4b. We refer to this gap's straight boundary as the *bottom* of the gap. We call a point's vertical distance from the bottom its *height*. We also utilize the terms *left* and *right* in an obvious manner. Assume for now that we can place two *separator* disks in the gap's left and right corner, touching the bottom and such that the distance between the rightmost point p_l of the left separator and the leftmost point p_r of the right separator is exactly 12 units. Since $B \equiv 0 \mod 4$ we know that $a \in \{B/4 + 1, \ldots, B/2 - 1\}$ for any $a \in A$. Our first goal is to find a function $r : \{B/4, B/4+1, \ldots, B/2\} \to \mathbb{R}^+$ that assigns a disk radius to each input integer as well as to the values $B/4$ and $B/2$ such that a disk triple t together with two separator disks can be placed on the bottom of a gap without intersecting each other or the outer disks if and only if t is feasible. In the following, we show that $r(x) = 2 - (4 - 12x/B)/B$ will satisfy our needs.



(a) A gap in the original scenario.

(b) A gap in the simplified scenario.

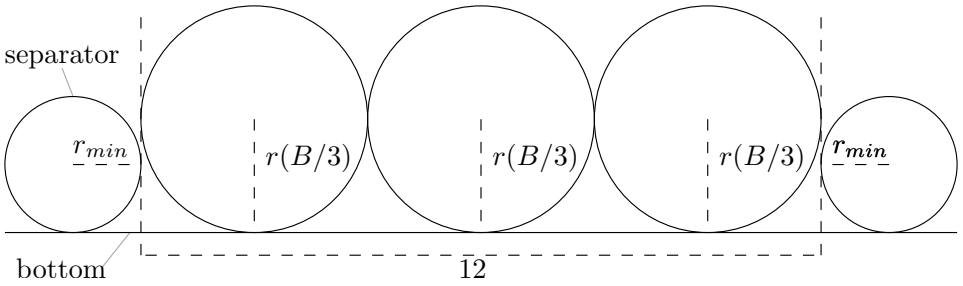
Figure 3.4.: Comparison of the two scenarios. In the original scenario the gap is bounded by two disks and its bow. In the simplified scenario the gap's bottom replaces its bow. The distance between the separators is 12 in both scenarios.

Assume for now that for any $a \in A$ it is not possible that a disk with radius $r(a)$ intersects one or even both of the outer disks that bound the gap when placed between the two separators (we prove this later in Section (c)). Observe that for $B > 12$ the function r is linear and that a triple of disks with uniform radius $r(B/3) = 2$ requires a total horizontal space of $2 \cdot 2 \cdot 3 = 12$ if placed tightly next to each other on a straight line. Therefore, if we choose the radius of the separators to be $r_{min} = r(B/4 + 1) = 2 - (1 - 12/B)/B$, it follows that every feasible disk triple fits in the gap since (1) a triple of disks with uniform radius $r(B/3)$ yields an upper bound for the amount of horizontal space required by any feasible disk triple and since (2) $r(B/3) > r_{min}$, which implies that if the three disks are placed next to each other on the bottom, the height of the leftmost point of the disk triple is greater than the height of the rightmost point p_l of the left separator and, therefore, these disks do not touch (and the same holds true for the right side respectively), see Figure 3.5a.

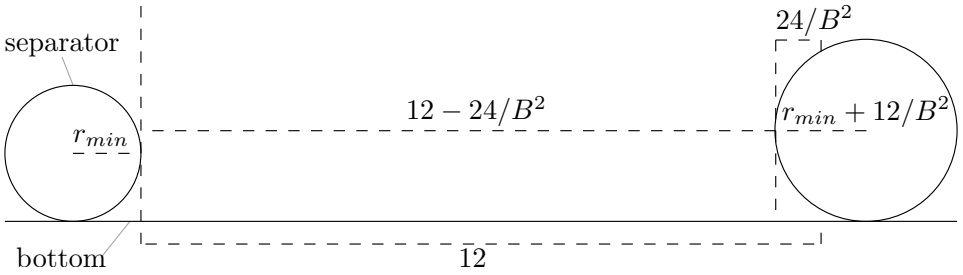
Note that r_{min} is the smallest possible input disk radius. Analogously we define r_{max}, the largest possible input disk radius with $r_{max} = r(B/2 - 1) = 2 + (2 - 12/B)/B$. Consider the disk triple $t_i = (r_{min}, r_{max}, r_{min})$. The sum of the integers corresponding to the disks of t_i is $B/4 + 1 + B/2 - 1 + B/4 + 1 = B + 1$ and, therefore, t_i is infeasible. When placing the three disks next to each other on a straight line, placing the disk with radius r_{max} in the middle maximizes the difference between the radii of adjacent disks and, therefore, minimizes the overall horizontal space required, which, using the Pythagorean Theorem, can be described as $s_i = 2r_{min} + 2\sqrt{(r_{max} + r_{min})^2 - (r_{max} - r_{min})^2}$, see Figure 3.6a. Since r is linear and $B + 1$ is the smallest possible sum of any infeasible integer triple, s_i is the least possible amount of horizontal space required by any infeasible disk triple. In order to prove that no infeasible disk triple triple can be placed in the gap together with two separators, we later show the existence of values $\varepsilon > 0$ and $\varepsilon_1, \varepsilon_2, \phi \geq 0$ with $\varepsilon = \varepsilon_1 + \varepsilon_2$ which satisfy the following two conditions (an explanation of these conditions as well as a definition for distance $d(\varepsilon_1, x)$, which occurs in these conditions, is given in the next paragraph).

$$12 + \varepsilon \leq s_i \tag{3.2}$$

$$d(\varepsilon_1, x) \leq r(x) - \phi, \forall x \in \{B/4 + 1, \ldots, B/2 - 1\} \tag{3.3}$$



(a) Depiction of a feasible input triple's disk representation. This particular representation requires the largest possible amount of horizontal space out of all representations for feasible input triples with sum B.

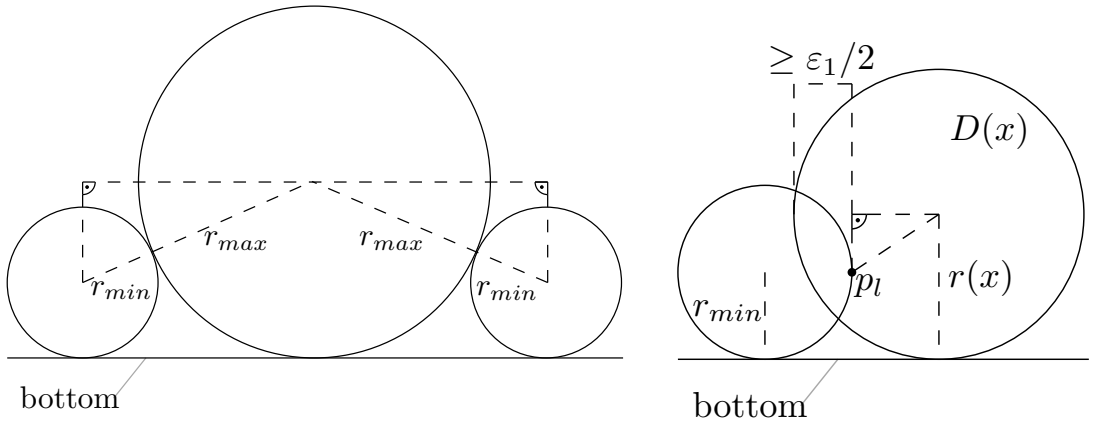


(b) An upper bound for the amount of horizontal space between the two disks placed in a gap's corner.

Figure 3.5.: Two illustrations regarding the proof of Theorem 7.

Recall that the distance between rightmost point p_l of the left separator and the leftmost point p_r of the right separator, which are located at height r_{min}, is exactly 12 units. Condition 3.2 ensures that all infeasible disk triples take up at least $12+\varepsilon$ units of horizontal space, however, this condition is not sufficient to guarantee that infeasible disk triples can not be placed between the separators since we do not know yet at what height the leftmost and the rightmost point of the disk triple are located. However, it is guaranteed that either the leftmost point of the disk triple is located at least $\varepsilon_1/2$ units to the left of p_l or the rightmost point of the triple is located at least $\varepsilon_1/2$ units to the right of p_r. Let now $x \in \{B/4+1, \ldots, B/2-1\}$ be an input integer. The Pythagorean Theorem tells us that the distance between p_l (p_r) and the center of a disk $D(x)$ with radius $r(x)$ whose center is located between the two separator disks and whose leftmost (rightmost) point is located at least $\varepsilon_1/2$ units to the left (right) of p_l (p_r) is at most $d(\varepsilon_1, x) = \sqrt{(r(x)-\varepsilon_1/2)^2 + (r(x)-r_{min})^2}$, as illustrated in Figure 3.6b. Condition 3.3 ensures that this distance is at most $r(x) - \phi$, implying that $D(x)$ intersects the left (right) separator. Therefore, Condition 3.2 and Condition 3.3 together guarantee that infeasible disk triples together with two seperators can not be placed inside a gap in the simplified scenario. The significance of ε_2 is discussed later in the proof, where we tailor our conditions to apply to the original scenario as well.

So far we assumed that the separators are always placed in the corners of the gap. In reality separators can be placed in different location, moreover, there could even be gaps with multiple separators and gaps with zero or one separator. Since the radius of the separators is r_{min}, which is the radius of the smallest possible input disk, it seems natural to place them in the gaps' corners to efficiently utilize the horizontal space. However, all feasible disk triples (except $(B/3, B/3, B/3)$) require less than 12 units of horizontal space. It might therefore be possible to place a feasible disk triple inside a gap together with two disks that are not necessarily separators but input disks with a radius greater than r_{min}. To account for this problem, we additionally show the existence of values $\xi > 0$



(a) The smallest possible infeasible disk triple. (b) Disk $D(x)$ is intersecting the separator.

Figure 3.6.: It is not possible to place an infeasible disk triple inside a simplified gap.

and $\xi_1, \xi_2, \psi \geq 0$ with $\xi = \xi_1 + \xi_2$ which satisfy the following two conditions (an explanation for these conditions as well as a definition for distance $d(\xi_1, x)$ and for length s_f, which occur in these conditions, is given in the next paragraph).

$$12 - 24/B^2 + \xi \leq s_f \tag{3.4}$$

$$d(\xi_1, x) \leq r(x) - \psi, \forall x \in \{B/4 + 1, \ldots, B/2 - 1\} \tag{3.5}$$

The second smallest possible input disk radius is $r(B/4 + 2) = 2 - (1 - 24/B)/B = 2 - (1 - 12/B)/B + 12/B^2 = r_{min} + 12/B^2$ and, therefore, $12 - 2 \cdot 12/B^2 = 12 - 24/B^2$ is an upper bound for the remaining horizontal space in a gap in which two disks have been placed such that one of the disks has radius greater than r_{min}, see Figure 3.5b. The input integers' values are at least $B/4 + 1$ and at most $B/2 - 1$, therefore, the horizontal space consumption of the disk triple $t_f = (r(B/4), r(B/2), r(B/4))$ is a lower bound for the space consumption of any feasible disk triple since the total difference between the radii of adjacent disks in t_f is larger than that of any feasible disk triple. Yet again we utilize the Pythagorean Theorem to describe t_f 's required horizontal space as $s_f = 2r(B/4) + 2\sqrt{(r(B/2) + r(B/4))^2 - (r(B/2) - r(B/4))^2}$. Condition 3.4 therefore ensures that any feasible disk triple consumes at least $12 - 24/B^2 + \xi$ and, analog to Condition 3.3, Condition 3.5 together with $d(\xi_1, x) = \sqrt{(r(x) - \xi_1/2)^2 + (r(x) - r_{min})^2}$ ensures that the disks of t_f intersect the disk(s) that have replaced the seperator(s). Like with ε_2 the significance of ξ_2 will become apparent later in the proof when we describe how to apply our conditions to the original scenario.

We evaluated that suitable values that satisfy our four conditions are $\varepsilon_1, \xi_1 = 16/B^2$ and $0 \leq \varepsilon_2, \phi, \xi_2, \psi \leq 1/B^2$ (recall that $\varepsilon = \varepsilon_1 + \varepsilon_2$ and $\xi = \xi_1 + \xi_2$). The calculations for these values are lengthy and have been moved to Appendix A. Specifically, Appendix A.1 shows that $\varepsilon_1, \xi_1 = 16/B^2$ and $0 \leq \phi, \psi \leq 1/B^2$ satisfy Condition 3.3 and Condition 3.5. Appendix A.2 states that $\varepsilon \leq 17/B^2$ is a sufficient choice to satisfy Condition 3.2, which implies that we need to ensure that $\varepsilon_2 \leq 1/B^2$. Similarly, Appendix A.3 states that $\xi \leq 17/B^2$ is a sufficient value to satisfy Condition 3.4, which implies that we need to ensure that $\xi_2 \leq 1/B^2$.

In the simplified scenario the separators are placed 12 units apart from each other on a straight line. In the original scenario we also want separators to be placed in the gap's corners and the distance between rightmost point p_l of the left separator and the leftmost

point p_r of the right separator is still supposed to be 12 units. In Section (c) of this proof, we use a geometric construction to show that in the original scenario even a disk with radius r_{max} placed right next to a separator can not intersect an outer disk, which implies that input disks of all sizes can actually be placed inside the gap together with the two separators. Together with the upper bound of 12 units for the horizontal space consumption of any feasible disk triple placed next to each other on a straight line, this ensures that any feasible disk triple together with a pair of separators can be placed inside a gap in the original scenario.

We now describe how Conditions 3.2–3.5 translate to the original scenario. By virtue of the previous calculations, Condition 3.2 and Condition 3.4 ensure that every infeasible disk triple requires at least $12+\varepsilon = 12+16/B^2+\varepsilon_2, 0 \leq \varepsilon_2 \leq 1/B^2$ units of horizontal space and that every feasible disk triple requires at least $12-24/B^2+\xi = 12-8/B^2+\xi_2, 0 \leq \xi_2 \leq 1/B^2$ units of horizontal space if placed on a straight line. Note that the horizontal space consumption of a disk triple placed on the bow is lower compared to the disk triple being placed on the bottom. We define $\varepsilon_2, \xi_2 = 1/B^2$. If we make sure that ε_2 (ξ_2) is an upper bound for the amount of horizontal space that can be saved in the original scenario compared to the simplified scenario when placing any infeasible (feasible) disk triple, we know that Condition 3.2 (Condition 3.4) still holds true for the original scenario. Therefore, we know that, without loss of generality, the leftmost point of the leftmost disk $D(x)$ of a disk triple placed in the gap is located at least $\varepsilon_1/2, \xi_1/2 = 8/B^2$ units to the left of the rightmost point p_l of the left separator or the disk that has replaced it respectively. Let $r(x)$ be the radius of $D(x)$. For the simplified scenario Condition 3.3 and Condition 3.5 ensure that the distance between p_l and the center of $D(x)$ is at most $r(x) - \psi, 0 \leq \psi \leq 1/B^2$ or $r(x) - \phi, 0 \leq \phi \leq 1/B^2$, indicating that $D(x)$ intersects the left separator or replacing disk respectively. In the simplified scenario the conditions hold true even for $\phi, \psi = 0$. We define $\phi, \psi = 1/B^2$. If we make sure that ϕ and ψ are upper bounds for the increase of distance between p_l and the center of $D(x)$ in the original scenario compared to the simplified scenario, we know that Condition 3.3 and Condition 3.5 still hold true in the original scenario.

In the original scenario, consider a straight line directly below the two separators. We call this straight line the gap's *chord*, see Figure 3.4a. The gap's chord has a function similar to the bottom in the simplified scenario. The required upper bounds for Condition 3.3 and Condition 3.5 are certainly achieved if the maximum distance d between the gap's bow and its chord is at most $1/B^2 = \psi, \phi$, since this implies that in the orignial scenario $D(x)$ is located at most $1/B^2$ units above the chord. Interestingly enough, this requirement for d is also sufficient to satisfy the required upper bounds for Condition 3.2 and Condition 3.4 as we shall elaborate in remainder of this paragraph. Consider the disk triple $t = (r(B/4), r(B/2), r(B/4))$. Yet again utilizing the Pythagorean Theorem, we calculate an upper bound for the amount of horizontal space that can be saved when placing t on a gap's bow instead of its string to be $s = 2(\sqrt{(r(B/2)+r(B/4))^2 - (r(B/2)-r(B/4))^2} - \sqrt{(r(B/2)+r(B/4))^2 - (r(B/2)+d-r(B/4))^2})$ by simply moving the left and right disk down by d units and as far to the center disk as possible, see Figure 3.7. Since the sum of the integer triple corresponding to t is B and $B/4$ is smaller and $B/2$ is greater than any input integer, the value s is also an upper bound for the amount of horizontal space that can be saved in the original scenario compared to the simplified scenario when placing any infeasible or feasible disk triple since the differences between the radii of adjacent disks in t are larger than in any actual input disk triple. In Appendix A.4 we show that for $d \leq 1/B^2$ an upper bound for s is $1/B^2 = \varepsilon_2, \xi_2$. In order to conclude the proof, it therefore remains to describe how to choose the radii for the central and outer disks and how to create the gaps such that $d \leq 1/B^2$.

Recall that we have a central disk D_c with radius r_c and m outer disks with radius r_o

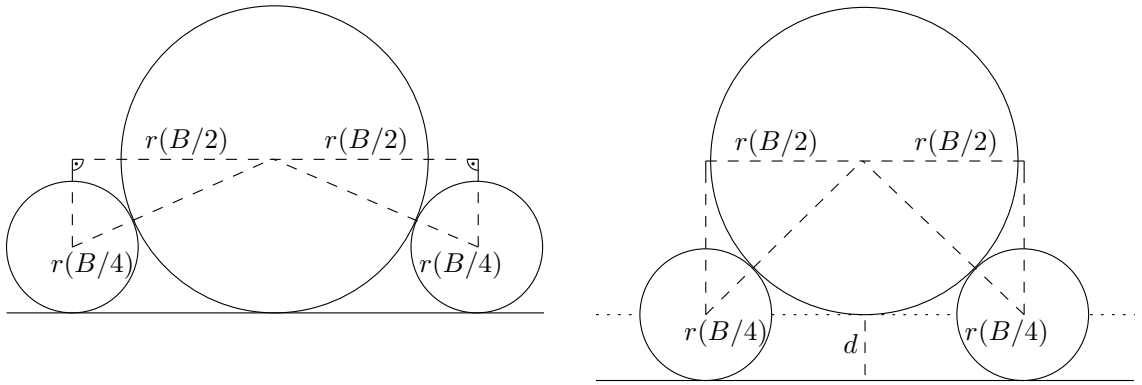


Figure 3.7.: An upper bound for the amount of horizontal space that can be saved by placing a disk triple on the gap's bow instead of its chord can be calculated by replacing the bow by a straight line d units above the chord and comparing the required space to the space required when placing the two outer disks directly on the chord.

which are tightly packed around D_c such that m equal-sized gaps are created. With basic trigonometry we see that $r_c + r_o = r_o/\sin(\pi/m)$ (see Figure 3.10) and, therefore, $r_c = r_o/\sin(\pi/m) - r_o$. Clearly, there always exists a value r_o such that the two separator disks can be placed in each gap's corners and such the distance between each pair of separators is exactly 12 units. The value of particular importance is the maximum distance between a gap's bow and its chord, which, utilizing the Pythagorean Theorem, can be calculated to be $d = r_c - (\sqrt{(r_c + r_{min})^2 - (6 + r_{min})^2} - r_{min})$, see Figure 3.4a. The crucial observation is that we do not necessarily need to choose m to be n. Instead we may choose any $m \geq n$ and thereby decrease d, as long as we make sure that m is still a polynomial in the input's size or numeric values (see Section (a)) and that none of the input disks or separators can be placed in one of the $m - n$ additional gaps (see Section (b)).

(a) Determining a suitable number of gaps

In order to choose an $m \geq n$ such that $d \leq 1/B^2$, we require some information about the radius r_o of the outer disks. A precise calculation of this value yields a complicated formula, however, a lower as well as an upper bound for r_o are sufficient to conclude our argument. Clearly, $r_o^l = 6$ is a lower bound for r_o. In Section (c) we designate a minimum value of $m_{min} = 6$ for m and use it to show that $r_o^u = 38$ is an upper bound for r_o. Recalling that r_{min} is a polynomial in B, that $m \geq m_{min} = 6$ and utilizing that $\sin(x) \leq x, \forall x \geq 0$, we can now prove that m can be chosen as a polynomial in B such that $d \leq 1/B^2$:

$$
\begin{aligned}
d \leq 1/B^2 &\Leftrightarrow \\
r_c - (\sqrt{(r_c + r_{min})^2 - (6 + r_{min})^2} - r_{min}) \leq 1/B^2 &\Leftrightarrow \\
(r_c + r_{min}) - 1/B^2 \leq \sqrt{(r_c + r_{min})^2 - (6 + r_{min})^2} &\Leftarrow \\
1/B^4 - 2(r_c + r_{min})/B^2 \leq -(6 + r_{min})^2 &\Leftrightarrow \\
1/(2B^2) + B^2(6 + r_{min})^2/2 - r_{min} \leq r_c &\Leftrightarrow \\
1/(2B^2) + B^2(6 + r_{min})^2/2 - r_{min} + r_o \leq r_o/\sin(\pi/m) &\Leftrightarrow \\
\sin(\pi/m) \leq r_o/(1/(2B^2) + B^2(6 + r_{min})^2/2 - r_{min} + r_o) &\Leftarrow \\
\pi/m \leq r_o/(1/(2B^2) + B^2(6 + r_{min})^2/2 - r_{min} + r_o) &\Leftrightarrow \\
m \geq (\pi/r_o) \cdot (1/(2B^2) + B^2(6 + r_{min})^2/2 - r_{min} + r_o)) &\Leftarrow \\
m \geq (\pi/r_o^l) \cdot (1/(2B^2) + B^2(6 + r_{min})^2/2 - r_{min} + r_o^u))
\end{aligned}
$$

Therefore, we define $m = B^{c_m}$ where c_m is a sufficiently large constant. Note that we need to ensure that $m \geq n$, which is however no problem since we can, without loss of generality, assume that B is a multiple of n since we could simply multiply each input integer as well as the bound by n to obtain a problem instance that is a yes-instance if and only if the original instance was a yes instance and whose size is polynomial in the size of the original input.

(b) Coping with additional gaps

Our construction contains $m > n$ gaps. We need to ensure that it is not possible to use the additional gaps to solve an infeasible problem instance. We therefore attach another $m-n$ *huge plug* disks to the central disk. The uniform radius of the huge plugs is chosen so that each one fits tightly inside a gap. Embedding a huge plug in a gap creates two new, equally sized *small* gaps next to the central disk D_c. We distinguish three cases.

First, if a disk with radius r_{min} is too large to be embedded in a small gap, then no input disk can be embedded in the small gaps. Otherwise, we create two new maximum sized *small* plug disks per gap that tightly fit inside the small gaps. If the radius of the small plug disks is at least r_{min} and at most r_{max}, then the small gaps do not make an infeasible instance feasible since embedding one of the input disks D in a small gap requires embedding a small plug disk in a regular gap, which is only possible if the instance is feasible anyways. Note that since disks with radius r_{max} can be embedded inside the gaps, the radius of the huge plugs is at least r_{max}. Therefore, it is also not possible to embed two (or more) input disks in a single small gap since it is not even possible to embed four disks with radius r_{min} together with a huge plug inside a single gap since these five disks constitute an infeasible disk triple together, as well as two separators. Finally, if the radius of the small plugs is greater than r_{max}, the two small plugs together with the huge plug constitute an infeasible disk triple, it therefore is not possible to place any more input or separator disks inside the same gap.

(c) An upper bound for r_o and placing maximum sized disks

In this section, we first utilize a geometric construction to show that $r_o^u = 38$ is an upper bound for the outer disks' radius r_o and then use this result to prove that even disks with radius r_{max} placed in a gap right next to a separator do not intersect an outer disk in the original scenario, implying that the input disks can actually be placed inside the gaps.

Observe that the outer disk radius r_o increases when m decreases and when r_{min} increases. We designate a minimum value of $m_{min} = 6$ to m and observe that $r_{min} = r(B/4 + 1) =$

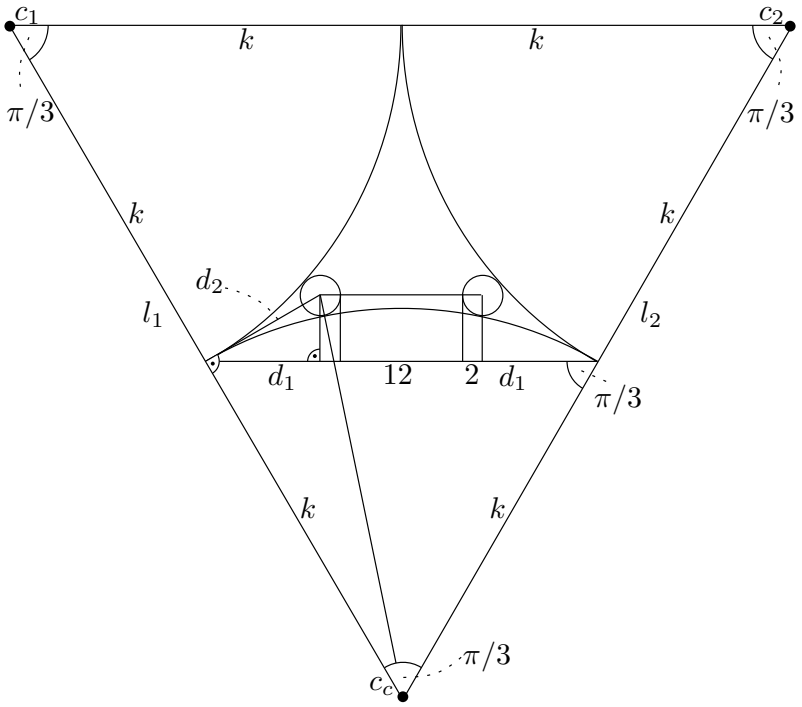


Figure 3.8.: The geometric construction for part (c) in the proof of Theorem 7.

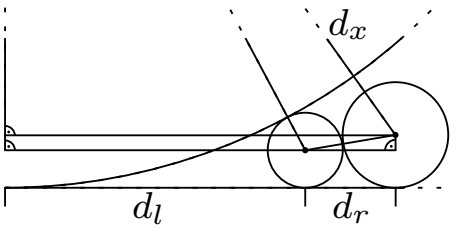


Figure 3.9.: A disk with radius r_{max} can always be placed right next to a separator in a corner of a gap.

$2-(1-12/B)/B < 2$ for any $B > 12$. The angle between two line segments l_1, l_2 bounded by the centers c_1, c_2 of two adjacent outer disks and the center c_c of of the central disk D_c when our construction contains m_{min} outer disks is $2\pi/m_{min} = \pi/3$ and therefore l_1 and l_2 together with the line segment $\overline{c_1c_2}$ constitute an equilateral triangle. This implies that the outer disk radius is equal to the radius of the central disk, we denote this radius with k. As illustrated in Figure 3.8, by using basic trigonometry as well as the Pythagorean Theorem, we obtain that k has to satisfy the equality $k = d_1 + 2 + 12 + 2 + d_1 = 2d_1 + 16$, where $d_1 = \cos(\pi/6) \cdot d_2 = \sqrt{3} \cdot d_2/2$ and $d_2 = \sqrt{(k+2)^2 - k^2} = \sqrt{4k+4} = 2\sqrt{k+1}$. This set of equalities solves for $k = 22 + 4\sqrt{5}\sqrt{3} = 37.4919... < 38 = r_o^u$.

We now show that a disk with radius r_{max} placed in a gap right next to a separator does not intersect an outer disk. Observe that placing a disk with radius r_{max} next to a separator without intersecting outer disks becomes more complicated as r_o, r_c and r_{max} increase and as r_{min} decreases (Figure 3.11 from Section (d) illustrates why this is the case). In order to show that placing a disk with radius r_{max} can always be placed as intended, we consider a gap created by using upper bounds for r_o, r_c and r_{max} and a lower bound for r_{min} as depicted in Figure 3.9. We show that the disk whose radius is the upper bound for r_{max} does not intersect the outer disk by determining the distance d_x between the centers of the two disks. In the previous paragraph, we have established an upper bound $r_o^u = 38$ for r_o. A lower bound for r_{min} is $2 - 1/13 < 2 - 1/B + 12/B^2$ and an upper bound for r_{max} is $2 + 1/6 > 2 + 2/B - 12/B^2$ since $B > 12$. As depicted in Figure 3.9 we can think of a maximum sized central disk as a straight line since $\lim_{m\to\infty} r_c = \infty$. By utilizing the Pythagorean Theorem, the distance in question can now be calculated as $d_x = \sqrt{(38 - (2 + 1/6))^2 + (d_l + d_r)^2}$ with $d_l = \sqrt{(38 + (2 - 1/12))^2 - (38 - (2 - 1/12))^2}$ and $d_r = \sqrt{((2 - 1/12) + (2 + 1/6))^2 - ((2 + 1/6) - (2 - 1/12))^2}$. This set of equalities solves for $d_x = 41.606... > 41 > 38 + (2 + 1/6)$, which concludes our proof.

(d) Precision

In this section we describe how to compute the disks radii in polynomial time. Note that for the input and separator disks the radius computation does not cause any complications since the output of our radius function r is always a rational number. Computing the radius of the outer disks and the central disk, however, is more complicated. Recall that the central radius can be described as $r_c = r_o/\sin(\pi/m) - r_o$. By using the Pythagorean Theorem, we obtain the following(see Figure 3.10).

$$\sqrt{(r_o/\sin(\pi/m))^2 - r_o^2} = d_3 = d_1 + d_2 = \sqrt{(r_o + r_{min})^2 - (r_o - 6 - r_{min})^2} + \sqrt{(r_o/\sin(\pi/m) - r_o + r_{min})^2 - (6 + r_{min})^2}$$

By keeping in mind that a lower bound for r_o is $r_o^u = 6$ we can determine a unique solution:

$$r_o = \frac{r_{min}\sin(\pi/m) - 3r_{min} - 6 - 2\sqrt{2r_{min}^2 + 6r_{min} - 2r_{min}^2\sin^2(\pi/m) - 6r_{min}\sin^2(\pi/m)}}{\sin(\pi/m) - 1}$$

Precise computation of this formula can take a superpolynomial amount of time. In the remainder of this section we shall, however, prove that we do not require precisely computed radii for the outer disks and the central disk. Instead, we shall approximate radii $r_o^\varepsilon \le r_o + \varepsilon_3$ and $r_c^\varepsilon \le r_o^\varepsilon/\sin(\pi/m) - r_o^\varepsilon + \varepsilon_4$ for the outer disks' radii and the central disk's radius respectively, where $\varepsilon_3 = 1/B^{c_3}$ and $\varepsilon_4 = 1/B^{c_4}$ are small, positive error terms.

Observe that for the calculations of the values $\varepsilon_1, \varepsilon_2, \xi_1, \xi_2, \psi, \phi$ (the existence of which implied that our radius function r satisfies Condition 3.2 - 3.5) we did not encounter or use any tight bounds. The same holds true for the statement with which we argued that

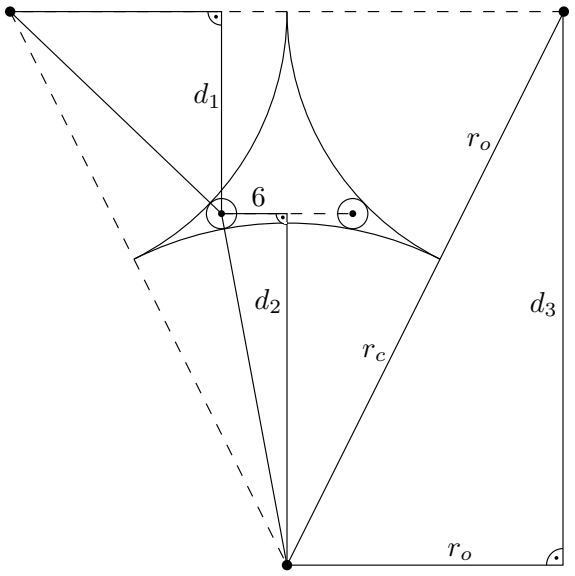


Figure 3.10.: Determining the outer disk radius by utilizing the Pythagorean Theorem.

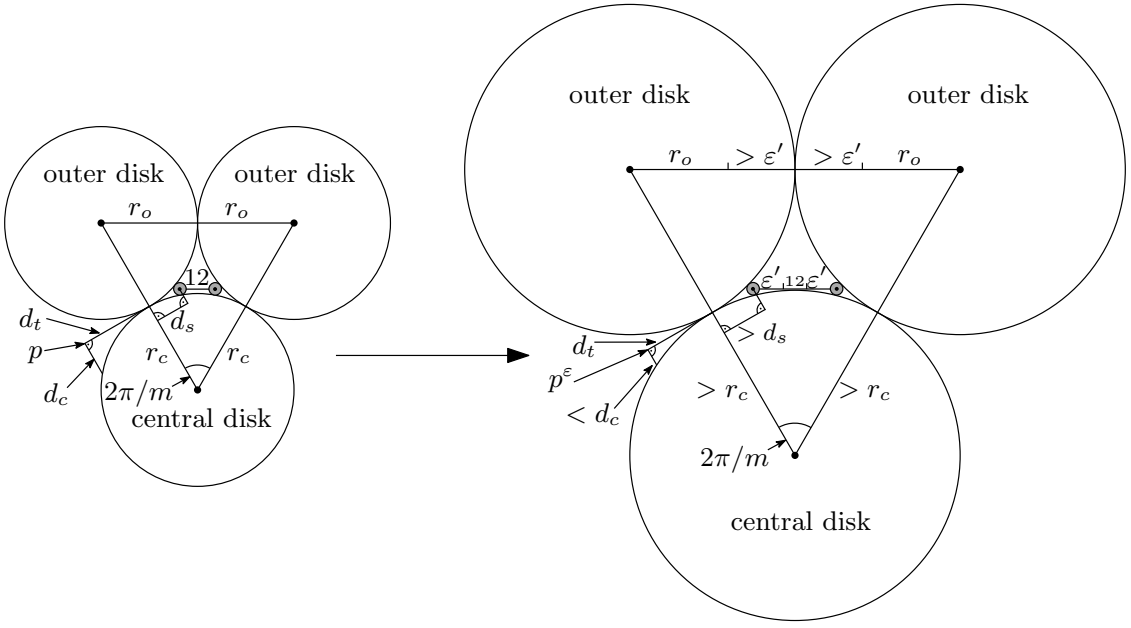


Figure 3.11.: Increasing the separator distance by $2\varepsilon'$ increases the outer disks' radii by at least ε'.

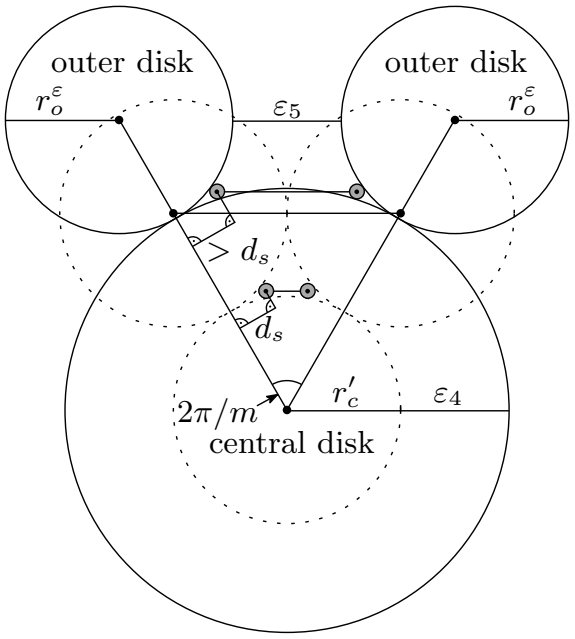


Figure 3.12.: Increasing the central radius by ε_4 creates a distance of ε_5 between the outer disks. The distance between the separators increases by at most ε_5.

we can choose m as a polynomial in B such that the distance d is sufficiently small. As a consequence, it is possible to adjust these values such that our conditions also hold true if we allow a distance of $12 + \varepsilon_s$ (instead of just 12) between the separator disks in a gap where $\varepsilon_s = 1/B^c > \varepsilon_3, \varepsilon_4$ for some suitable constant c. We now examine how this larger distance between the separators changes the outer and central disks' radii in a corresponding tight packing compared to the original tight packings with radii r_o and r_c.

The left part of Figure 3.11 illustrates two outer disks bounding a gap with the original radii and the original separator distance of 12. Changing this distance to $12 + 2\varepsilon'$ (as depicted in the right part of Figure 3.11) increases the required outer and central radii for a tight packing since we do not change m and, therefore, maintain the angle between the two outer disks. In both packings, consider the tangent line between the respective left outer disk and the central disk. We travel a distance d_t along these tangent lines and arrive at points p and p^ε (see Figure 3.11). From these points we travel orthogonally (to the tangents) until we reach the central disk. Let d_c be the distance traveled in the original packing and observe that the traveled distance in the modified packing is smaller than d_c since the radius of the central disk is larger. On an intuitive level, this means that the funnel-shaped regions next to the tangent points become more narrow as the radii of the outer and central disks increase. This phenomenon causes separator disks (which maintain their original size) in the modified packing to be pushed farther away from the lines that connect the centers of the central and outer disks than in the original packing (the distance d_s in Figure 3.11 increases). For this reason, increasing the separator distance from 12 to $12 + 2\varepsilon'$ pushes the centers of the outer disks at least distance ε' to the sides since this is also the distance that the separators move to the left or right respectively. We can conclude that increasing the separator distance by $2\varepsilon'$ increases the radius of the outer disks in a tight packing by at least ε'. The implication is that in a tight packing with outer disk radius $r_o^\varepsilon \leq r_o + \varepsilon_3$ and a corresponding (precise) central radius $r'_c = r_o^\varepsilon / \sin(\pi/m) - r_o^\varepsilon$ the distance between the separators is at most $12 + 2\varepsilon_3$.

Once again, we can not necessarily compute the central radius r'_c precisely. Instead, we approximate it as r_c^ε with $r'_c < r_c^\varepsilon \leq r'_c + \varepsilon_4 = r_o^\varepsilon / \sin(\pi/m) - r_o^\varepsilon + \varepsilon_4$, which basically pushes

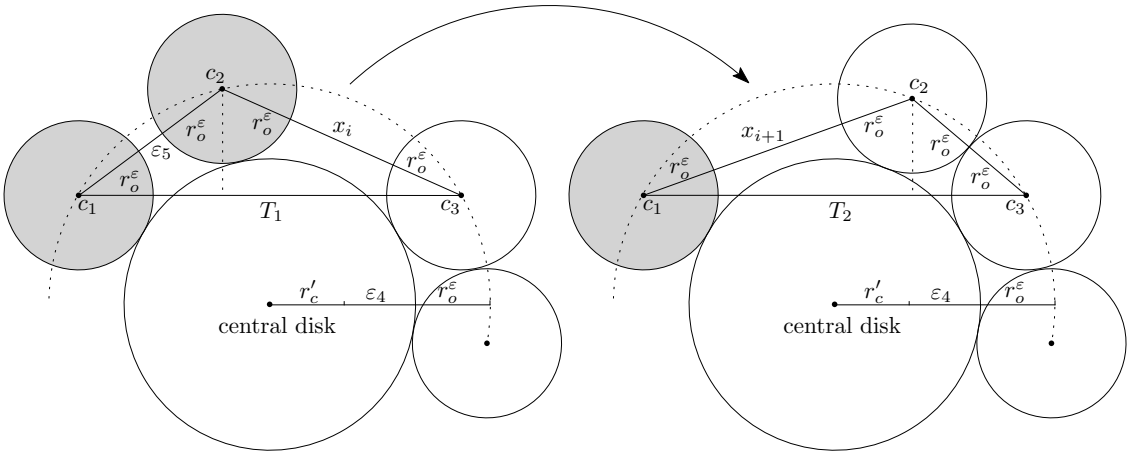


Figure 3.13.: When allowing $i+1$ instead of i outer disks to move (non-movable outer disks in gray), the maximum distance increases at most linear in ε_5.

the outer disks to the outside as depicted in Figure 3.12. Assuming that the outer disks can not deviate from these positions, this creates some distance ε_5 between the outer disks in each gap. The outer disk radius remains r_c^ε but the central disk radius is larger than in a tight packing with radius r'_c. Like in the argument in the previous paragraph, this causes the separator disks to be pushed away from the lines that connect the outer and central disks' centers. For this reason, the distance between the separators increases by at most ε_5 from at most $12 + 2\varepsilon_3$ to at most $12 + 2\varepsilon_3 + \varepsilon_5$.

So far, we have assumed that the outer disks can not deviate from their positions even though they are placed distance ε_5 apart from each other. In reality, however, the outer disks can rotate around the central disk and, therefore, the distance between two outer disks can increase to some value $\varepsilon_6 > \varepsilon_5$. We prove that $\varepsilon_6 < 2m\varepsilon_5$ by showing by induction that if we allow i of the outer disk to move, the maximum distance x_i between two outer disks is smaller than $2(i+1)\varepsilon_5$ for any $0 \leq i \leq m-1$. Clearly this holds true for $x_0 = \varepsilon_5 < 2\varepsilon_5$. Now assume that our hypothesis is true for some fixed $i \leq m-2$. Clearly, the distance x_i is maximized when we place all of the i movable disks close together and thereby create one large gap. One of the neighboring gaps is bounded by two non-movable (in step i) disks such that the distance between these disks is ε_5 as depicted in the left part of Figure 3.13. The distance x_{i+1} gets maximized by now allowing the previously non-movable disk next to the x_i gap to move such that the two gaps merge as illustrated in the right part of Figure 3.13. Consider the triangles T_1 (left) and T_2 (right) formed by the centers c_1, c_2, c_3 in Figure 3.13. The bottom side of these triangles is identical, the height of T_2 is smaller than the height of T_1 and the circumcircle of both triangles has radius $r'_c + \varepsilon_4 + r_o^\varepsilon$. We can conclude that the area of T_2 is smaller than the area of T_1. Recalling that the area of a triangle T with sides a, b, c can be described as $abc/(4r)$ where r is the radius of the circumcircle of T, we obtain $2r_o^\varepsilon(2r_o^\varepsilon + x_{i+1}) < (2r_o^\varepsilon + \varepsilon_5)(2r_o^\varepsilon + x_i)$ and, hence, $x_{i+1} < x_i + \varepsilon_5 + \varepsilon_5 x_i/(2r_o^\varepsilon) < 2(i+1)\varepsilon_5 + \varepsilon_5 + \varepsilon_5(2(i+1)\varepsilon_5)/(2r_o^\varepsilon)$ by our induction hypothesis. By choosing ε_4 and, therefore, ε_5 such that $2m\varepsilon_5 < 1$ we obtain that $x_{i+1} < 2(i+1)\varepsilon_5 + \varepsilon_5 + \varepsilon_5/(2r_o^\varepsilon) < 2(i+1)\varepsilon_5 + \varepsilon_5 + \varepsilon_5 = 2(i+2)\varepsilon_5$, which concludes our induction proof. Thus, the maximum distance between two outer disks increases to at most $\varepsilon_6 < 2m\varepsilon_5$. This increases the maximum distance between the separators to at most $12 + 2\varepsilon_3 + 2m\varepsilon_5$ as illustrated in Figure 3.14.

With basic trigonometry, we determine that $2r_o^\varepsilon + \varepsilon_5 = 2(r'_c + \varepsilon_4 + r_o^\varepsilon)\sin(\pi/m)$ and $2r_o^\varepsilon = 2(r'_c + r_o^\varepsilon)\sin(\pi/m)$, see Figure 3.12. We combine these two equalities and obtain $\varepsilon_5 = 2\varepsilon_4 \sin(\pi/m) < 2\varepsilon_4\pi/m < 2\varepsilon_4 4/m$. The maximum distance between two separators is, therefore, at most $12 + 2\varepsilon_3 + 16\varepsilon_4$. Recall that Condition 3.2 - 3.5 hold true for our radius

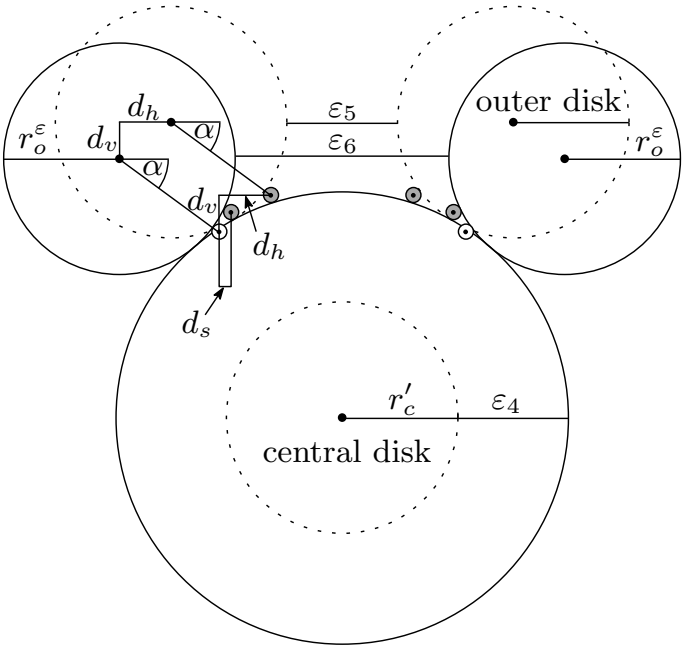


Figure 3.14.: Increasing the distance between two outer disks from ε_5 to ε_6 increases the distance between the separators by at most $\varepsilon_6 - \varepsilon_5$ since $d_s \geq 0$.

function r as long as the maximum distance between two separators is at most $12 + \varepsilon_s$ with $\varepsilon_s = 1/B^c$. We can now simply choose $\varepsilon_3 = 1/B^{c_3}$ and $\varepsilon_4 = 1/B^{c_4}$ such that $2\varepsilon_3 + 16\varepsilon_4 \leq \varepsilon_s$. Therefore, we can conclude that the approximate radii for the outer and central disks suffice.

It remains to argue that we can approximate our radii as required. The formulas for r_o and r'_c contain a constant number of square root and sine operations. Recall that $m = B^{c_m}$. Redefining and increasing m such that $m = 2^p$ with $2^{p-1} < B^{c_m} \leq 2^p = m$ causes no issues for our construction. Therefore, by using half-angle formulas, we can replace each sine operation in our formulas by $p = \log_2 m$ nested square root operations. In total, we therefore perform $O(\log m) = O(\log B^{c_m})$ square root operations. Individually, each square root approximation can be performed in polynomial time using Heron's quadratically converging method since we can easily determine constant upper and lower bounds for each square root term and use these as the initiation values. In order to approximate the nested square roots, we need to increase the approximation accuracy by an according polynomial amount. □

3.2. Disk Touching Graph Recognition with fixed Radii and Embedding

In this section we consider the Disk Touching Graph Recognition with fixed Radii and Embedding (DTRE) problem. We begin by summarizing implications of previous chapters and then provide a linear time algorithm that solves the DTRE for stars in the Real RAM model, in spite of the DTR being $\mathcal{NP}$-hard for stars.

Since the combinatorial embedding of any path or cycle is unique, Observation 4 and Theorem 4 yield the following:

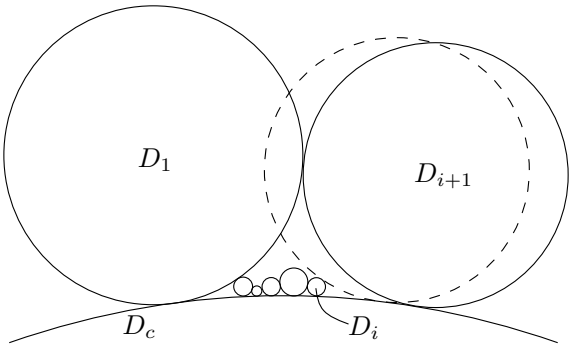


Figure 3.15.: Placing D_{i+1} right next to D_i would cause a collision (dashed circle) with D_1.

Corollary 6. *Any path P can be realized as a disk touching graph with respect to any radius assignment for P and any combinatorial embedding for P.*

Corollary 7. *Let $C = (V, E)$ be a cycle, let $r : V \to \mathbb{R}^+$ be a radius assignment for C and let Γ be a combinatorial embedding for C. There exists a realization of C as a disk touching graph with respect to r and Γ.*

Theorem 3 from the Unit Disk Touching Graph Recognition with fixed Embedding section carries over to the more general DTRE as follows:

Corollary 8. *The Disk Touching Graph Recognition with fixed Radii and Embedding problem is $\mathcal{NP}$-hard even for outerplanar graphs.*

Theorem 8. *For stars the Disk Touching Graph Recognition with fixed Radii and Embedding problem can be decided in $O(n)$ time in the Real RAM model, where n is the number of vertices of the given star.*

Proof. Let $S = (V, E)$ be a star, Γ be a combinatorial embedding for S and r be a radius assignment for S. Let $v_c \in V$ be the central vertex of S and let $(v_1, \ldots, v_{|V|-1})$ a sequence of the remaining vertices which is clockwise-ordered according to Γ. Let $D_c, D_1, \ldots, D_{|V|-1}$ be the disks that correspond to $v_c, v_1, \ldots, v_{|V|-1}$ and let $r_c, r_1, \ldots, r_{|V|-1}$ be the disks' radii respectively. We can, without loss of generality, assume that $r_1 \geq r_i$ for any $2 \leq i \leq |V|-1$.

We provide a constructive algorithm that can recognize no-instances. Suppose for now, however, that we are dealing with a yes-instance. We start by placing D_1 adjacent to D_c and add the remaining disks in clockwise order around D_c. Adding D_2 tightly next to D_1 and adjacent to D_c is always possible. Now suppose we have already added $D_3, \ldots, D_i$ where $i < |V|$. If $r_{i+1} \leq r_i$, we can simply place D_{i+1} tightly next to D_i while touching D_c. If, however, $r_{i+1} > r_i$, we have to test whether placing D_{i+1} tightly next to D_i would cause D_{i+1} to intersect one (or more) of the previously added disks and if this is the case, find a new suitable location for D_{i+1}. As depicted in Figure 3.15, it is even possible that our initial location for D_{i+1} causes D_{i+1} to intersect the first disk D_1. Simply testing for collisions with all previously added disks would yield a runtime of $O(i)$ for this step and, therefore, a total runtime of $O(|V|^2)$. However, we can improve this runtime to amortized $O(|V|)$ by maintining a list L in which we add each disk as we place it in our disk touching graph. Now, if $r_{i+1} > r_i$, we traverse the list backwards until we find a disk D_j with $r_{i+1} \leq r_j$ and test for collision with any of the traversed disks. A crucial observation is that none of the remaining disks $D_{i+2}, \ldots, D_{|V|-1}$ can have a collision with any of the disks $D_j, \ldots, D_i$ since all of these disks have radius at most r_{i+1}, thus we can delete these disks from L and therefore, in total, each disk is traversed at most once.

If we are indeed dealing with a yes-instance, our approach clearly generates disk touching graph that realizes S with respect to r and Γ. To cope with no-instances, every time we add a disk, we additionally test whether it intersects D_1. This is sufficient, since $r_1 \geq r_i$ for any $2 \leq i \leq |V| - 1$, which implies that the new disk can not intersect any disk that is located clockwise after D_1. Technically, even in the Real RAM model we can not simply place disks barely not touching next to each other if we do not know how to determine a sufficiently small distance that guarantees that we recognize every yes-instance. However, we can simply compute the disk locations such that tightly placed disks do actually touch. This way, if the last added disk does not intersect D_1, since the disk centers have real coordinates there clearly exists a realization. Note that with the modification, a collision only occurs if disks do intersect in more than one point. □

4. Recognition problems with fixed seeds

In this chapter we consider the Disk Touching Graph Recognition with fixed Seeds (DTS) and the Unit Disk Touching Graph Recognition with fixed Seeds (and Embedding) (UDTS and UDTSE) problems. Recall that in these problems we need to decide whether a realization of a given graph $G = (V, E)$ exists that respects a given seed assignment $\sigma : V \to \mathbb{R}^2$, meaning that $\sigma(v) \in D_v$ for every $v \in V$ where $D_v \in \mathcal{V}$ is the disk that corresponds to v. In Section 4.1 we strengthen an $\mathcal{NP}$-hardness result of Atienza et al. [AdCC+12] by showing that DTS remains $\mathcal{NP}$-hard even for trees. In Section 4.2 we combine assigning seeds with assigning radii, specifically uniform radii. We show that UDTS (and UDTSE) are $\mathcal{NP}$-hard, even if the input graph is a path.

4.1. Disk Touching Graph Recognition with fixed Seeds

Atienza et al. [AdCC+12] show that the Disk Touching Graph Recognition with fixed Seeds (DTS) problem is $\mathcal{NP}$-hard by performing a reduction from Planar 3-Satisfiability (P3SAT). In this reduction, for any P3SAT instance (U, C), they construct a graph G and a seed assignment σ_G for G, such that G can be realized as a disk touching graph respecting σ_G if and only if C is satisfiable. In this section we first recapitulate their reduction and observe that the constructed graph G is outerplanar (and not connected). We then extend their seed assignment σ_G and transform the graph G into a tree T that can be realized as a disk touching graph respecting the extended seed assignment σ if and only if C is satisfiable. We thereby strengthen the result of Atienza et al. by showing that the DTS problem is $\mathcal{NP}$-hard even for trees.

Similar to Cabello et al. [CDR07], Atienza et al. make use of the fact that each P3SAT instance can be represented by a planar (multi-) graph that be drawn on a slanted grid of polynomial size such that all variable vertices are placed on a horizontal line and such that the clause vertices are connected from above and below in a comb-shaped manner [KR92] (see Section 1.1.2). They construct G and σ_G on a triangular grid. As basic subgraphs in their construction serve the so-called *chains* of adjacent points/vertices. Each chain has one of excatly two possible states in any realization and they are used to propagate truth states between their variable and clause gadgets. A small part of a chain is depicted in Figure 4.1. We refer to the seeds s and s' as *stopper elements* for the *inner* seed p. Note how the seeds s, t_1, t_2, r_1, r_2, q and their counterparts $s', t'_1, t'_2, r'_1, r'_2, q'$ and the adjacencies are chosen such that the center of the disk D_p that covers p has to be located either in the region I_p^{true} or in the region I_p^{false} and that these regions can made arbitrarily small by

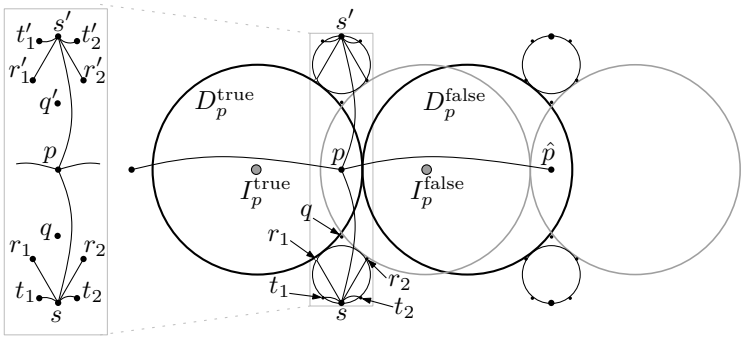


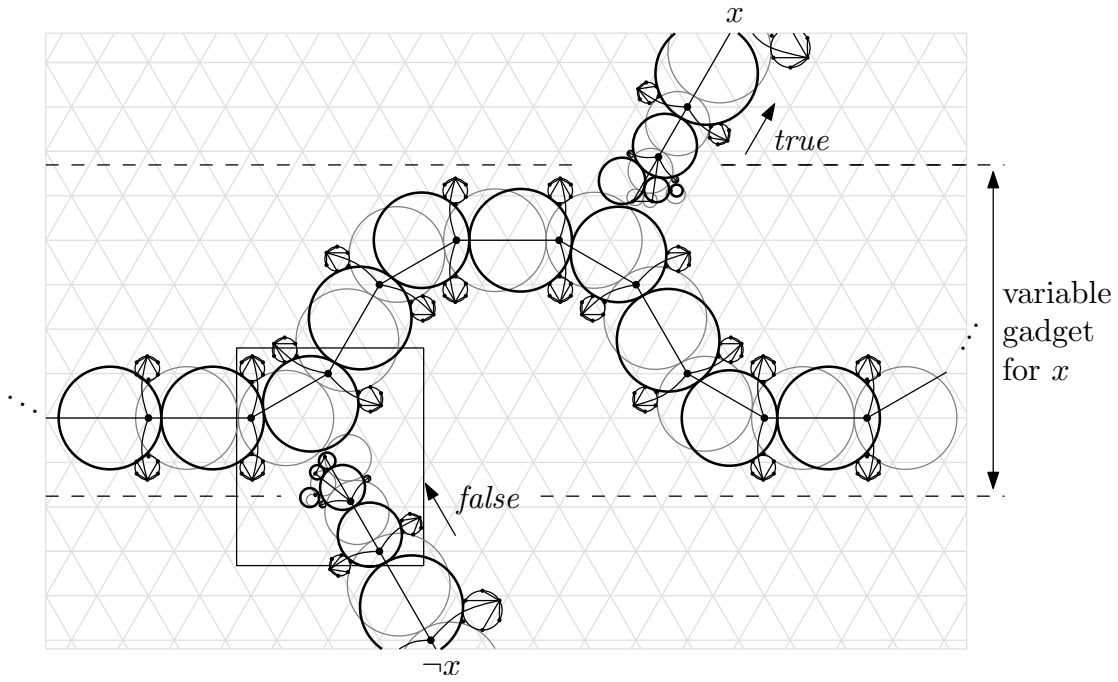
Figure 4.1.: A small part of a chain in the construction by Atienza et al. [AdCC+12]. This figure is a modified version of a figure provided by Atienza et al. that occured in [AdCC+12].

choosing the aforementioned seeds appropriately. Accordingly, D_p has to be located either to the left (like the black disk D_p^{true}) or to the right (like gray disk D_p^{false}). Choosing the left (right) embedding forces the disk covering the other inner seed $\hat{p}$ to also be embedded to the left (right), the depicted small chain part therefore only has two states. Clearly, this holds true for longer chains with more inner seeds as well, and it is not necessary that all the inner seeds are collinear, which can be seen in Figure 4.2a. The wavelike chain in the middle of this figure is part of a *variable* gadget in true (false) state illustrated by the black (gray) disks. Additional chains connect the variable gadget for $x \in U$ to all clause gadgets whose corresponding clauses contain a literal of x. The seeds at the end of such a chain are chosen such that the disks are pulled towards the variable gadget if the clause's literal for x evaluates to false for the respective state of the variable gadget. This is illustrated in Figure 4.2b and Figure 4.2c. Note how in Figure 4.2b the two topmost disks of the bottom chain can be embedded since the disks of the chain are embedded to the top and, therefore, towards the variable gadget. However, in Figure 4.2c we see that if the disks are embedded to the bottom and, therefore, towards the clause gadget, one of the topmost disks intersects the variable gadget. If a chain's literal evaluates to true, its disks can be embedded in either direction, see the top chain in Figure 4.2a. Figure 4.3a depicts a valid realization of a *clause* gadget. Note, how the last two disks of a chain oriented towards the clause gadget can have a very small radius. The last two disks of a chain oriented away from the clause gadget, however, are forced to have a much larger radius. In fact, if all three chains are oriented away from the gadget, it is impossible to find a valid realization, see Figure 4.3b.

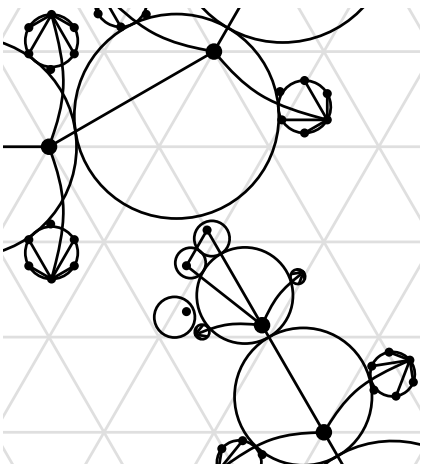
Let t be a truth assignment for U. If C is not satisfied by t then at least one of its clauses is not satisfied by t. All chains of the corresponding clause gadget are therefore pulled towards the variable gadgets and there exists no realization of G with respect to σ_G. However, if C is satisfiable, than there exists a truth assignment t for U for which all clauses are satisfied. Therefore, one of the literals of each clause evaluates to true for t and the corresponding chains can be oriented towards their respective clause gadget thereby allowing a valid realization. Atienza et al. conclude their reduction by arguing that the time required to construct G and σ_G is polynomial in the size of (U, C).

Clearly each chain in this construction corresponds to a non-connected (the seeds q and q' in Figure 4.1 have no incident edges), outerplanar subgraph of G, which therefore also is non-connected and outerplanar. This yields the following.

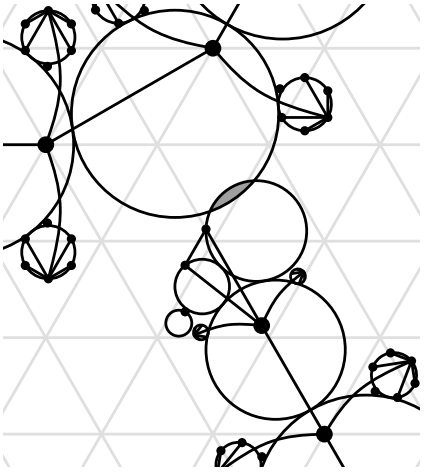
Observation 5. *The Disk Touching Graph Recognition with fixed Seeds problem is $\mathcal{NP}$-hard even for outerplanar graphs.*



(a) Variable gadget for x in true (false) state illustrated by the black (gray) disks.

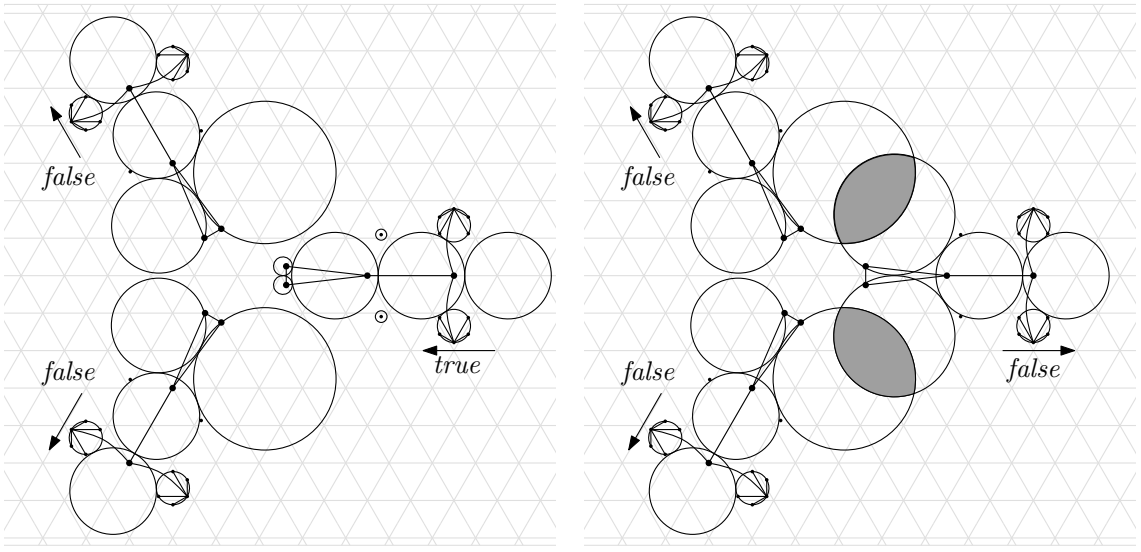


(b) Valid realization.



(c) Invalid realization.

Figure 4.2.: A variable gadget in the construction by Atienza et al. [AdCC+12]. These figures were provided by Atienza et al. and occured in [AdCC+12].



(a) Valid realization.

(b) Invalid realization.

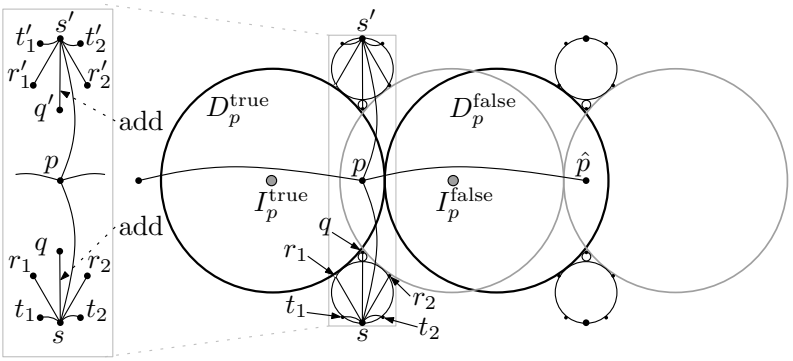
Figure 4.3.: A clause gadget in the construction by Atienza et al. [AdCC+12]. These figures were provided by Atienza et al. and occured in [AdCC+12].

By slightly modifying G and σ_G, however, we can strengthen this result.

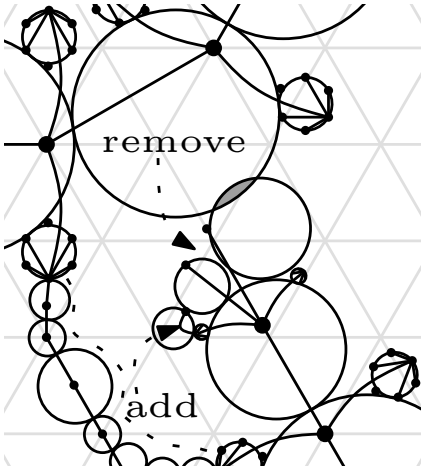
Theorem 9. *The Disk Touching Graph Recognition with fixed Seeds problem is $\mathcal{NP}$-hard even for trees.*

Proof. For this proof, we modify the previously presented construction by Atienza et al. [AdCC+12]. In particular, we transform the constructed outerplanar, non-connected graph G and its seed assignment σ_G into a tree T (by adding/removing some edges and vertices) and a seed assignment σ such that T can be realized as a disk touching graph with respect to σ if and only if G can be realized as a disk touching graph with respect to σ_G.

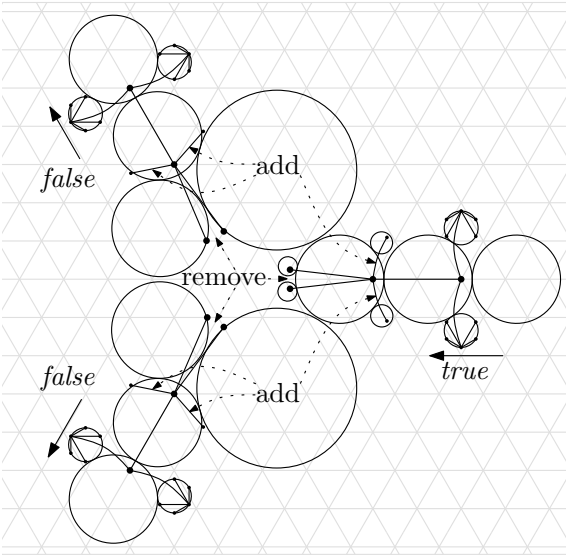
First, we reconsider the construction of the inner parts of the chains as depicted in Figure 4.1. We add an edge between seed q and s and q' and s' for any inner seed p and thereby transform the subgraph of the inner part of each chain into a tree as depicted in Figure 4.4a. We additionally shift the seeds q (q') a little bit towards s (s') such that the additional edges do not change the original purpose of s, s', q and q' and, therefore, such that all previously (in-)valid realizations are still (in-)valid. Next, we consider the endings of the chains that connect variable and clause gadgets. We add an edge between the leftmost singleton seed and the nearest stopper element of the bottom chain as depicted in Figure 4.4b. Note, that the edge between the two topmost seeds of the bottom chain is not necessary to ensure that the disk of the right seed intersects the variable gadget if the chain's disks are embedded away from the variable gadget. We can therefore remove it without creating new possible valid realizations. Similarly, we can remove the edge between the two seeds at the other end of the chain, which are (for example) the two leftmost seeds of the right chain as depicted in Figure 4.4c. Even if this edge is removed, the last two disks of the chain are forced to have large radii if the chain is pulled away from the clause gadget which can not be the case for all three chains. We additionally connect the two singleton seeds at this end of the chain to the common neighbor of the chain's last two seeds, see Figure 4.4c. The subgraphs of each of these chains, as well as the subgraphs of all variable gadgets are now trees. It remains to connect these trees, which can be done by adding a simple path of seeds between each literal chain and its corresponding variable



(a) A modified chain.



(b) A modified variable gadget.



(c) A modified clause gadget.

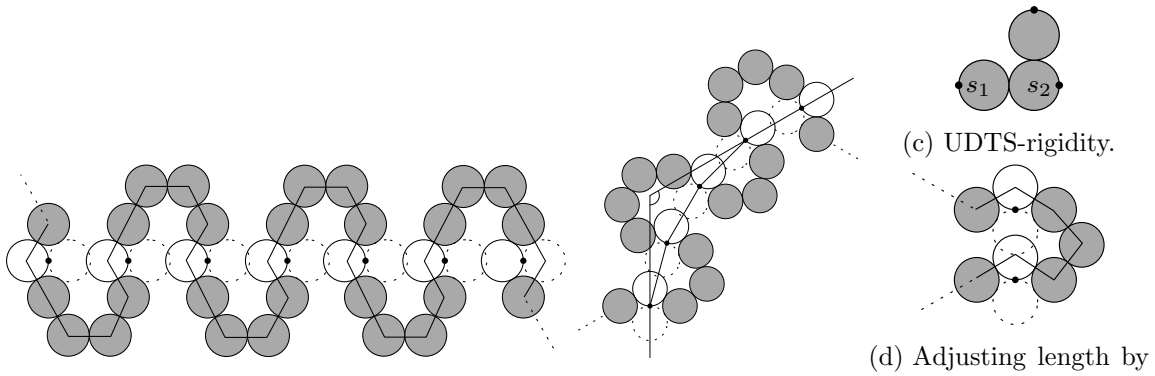
Figure 4.4.: The construction of Atienza et al. [AdCC+12] can be modified such that the resulting graph is a tree. These figures are modified versions of figures provided by Atienza et al. that occured in [AdCC+12].

gadget (see the bottom-left corner of Figure 4.4b) and by connecting all variable gadgets by adding additional simple paths of seeds between them. These additional paths should connect to stopper elements such that no previously valid realization becomes invalid.

For any P3SAT instance (U, C) we have created a tree T and a seed assignment σ such that T can be realized as a disk touching graph with respect to σ if and only if G can be realized as a disk touching graph with respect to σ_G, which by the construction of Atienza et al. is the case if and only if C is satisfiable. Atienza et al. argue that amount of time to construct G and σ_G is polynomial in the size of (U, C) . The time required to transform G and σ_G into T and σ is polynomial in the size of G and σ_G and, therefore, polynomial in the size of (U, C), which concludes our proof. □

4.2. Unit Disk Touching Graph Recognition with fixed Seeds (and Embedding)

In this section we consider the Unit Disk Touching Graph Recognition with fixed Seeds (UDTS) problem, which combines assigned seeds and assigned radii. We begin by defining a rigidity concept for the UDTS context and then prove that it is $\mathcal{NP}$-hard to decide whether there exists a unit disk touching graph realization of a given graph that respects



(a) A chain and the corresponding path. (b) A 60° turn. (c) UDTS-rigidity. (d) Adjusting length by increasing overlap.

Figure 4.5.: The elemental components for the reduction in Theorem 10.

a given seed assignment even if the graph is a path. This result carries over to the Unit Disk Touching Graph Recognition with fixed Seeds and Embedding (UDTSE) problem.

Let G be a graph, σ be a seed assignment for G and $\mathcal{G}$ be a realization of G as a unit disk touching graph that respects σ. We say that G is *UDTS-rigid* with respect to σ if and only if $\mathcal{G}$ is the only possible realization of G as a unit disk touching graph that respects σ.

Theorem 10. *The Unit Disk Touching Graph Recognition with fixed Seeds (UDTS) problem is $\mathcal{NP}$-hard, even for paths.*

Proof. We perform a polynomial-time reduction from Planar 3-Satisfiability (P3SAT). Recall that any P3SAT instance (U, C) can be represented as a planar (multi-) graph H that can be drawn on a slanted grid of polynomial size such that all variable vertices are placed on a horizontal line and such that the clause vertices are connected from above and below in a comb-shaped manner (see Section 1.1.2). Due to formatting reasons, however, we rotate this slanted layout 90° clockwise such that the variable vertices now are organized on a vertical line. We construct a path G and a seed assignment σ for G, which resemble the planar, slanted and rotated drawing of H such that G can be realized as a unit disk touching graph that respects σ if and only if C is satisfiable.

Figure 4.5c illustrates that it is very easy to construct UDTS-rigid subgraphs containing at least two adjacent vertices v_1 and v_2. This can be done by simply assigning seeds s_1 and s_2 to v_1 and v_2 such that the distance between s_1 and s_2 is exactly twice the unit diameter. Further vertices/disks can be attached in a UDTS-rigid manner by choosing their seeds exactly one unit diameter away from the desired touching point. In the figures in the remainder of this section, we represent UDTS-rigid structures by gray disks and omit representing the corresponding seeds.

The most basic components in our construction are structures called *chains*, as depicted in Figure 4.5a. Note that the depicted structure is a path and observe that any disk that represents any of the vertices belonging to one of the seeds in the middle can be placed in one of exactly two location as illustrated by the straight and dashed white disks. We say that these disks/vertices/seeds belong to the chains *inner path*. The potential locations for disks belonging to two consecutive vertices of the inner path overlap each other. Therefore, if the left location is chosen for the seed on the very right, the left location has to be chosen for any other disk of the inner path as well (and vice versa). Chains are not restricted to being straight. They can form turns as depicted in Figure 4.5b, in which a 60° turn is realized by a succession of four 15° turns. We can fine-tune the length of a straight chain by shortening the distance between two consecutive seeds from the chains inner path as depicted in Figure 4.5d.

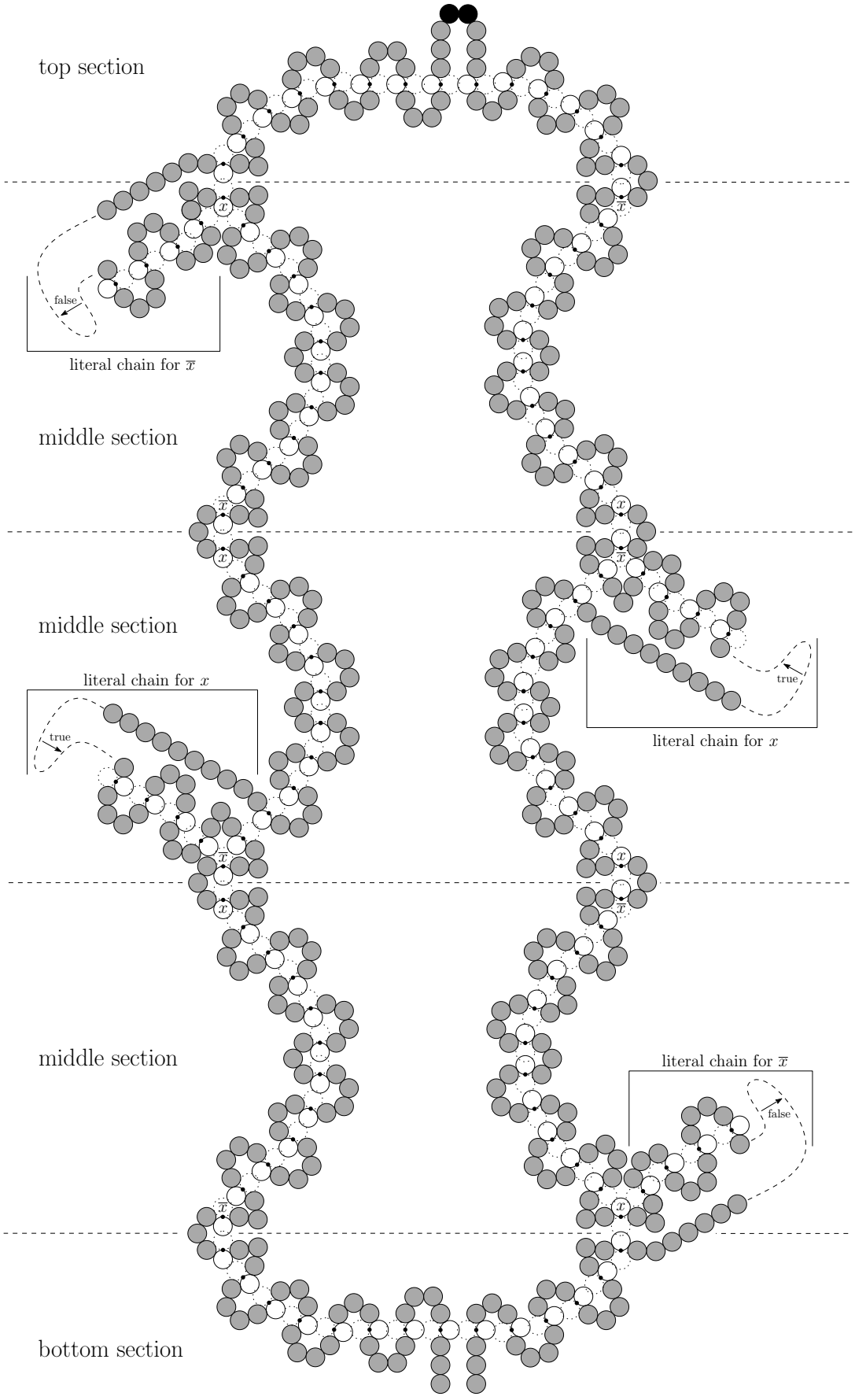
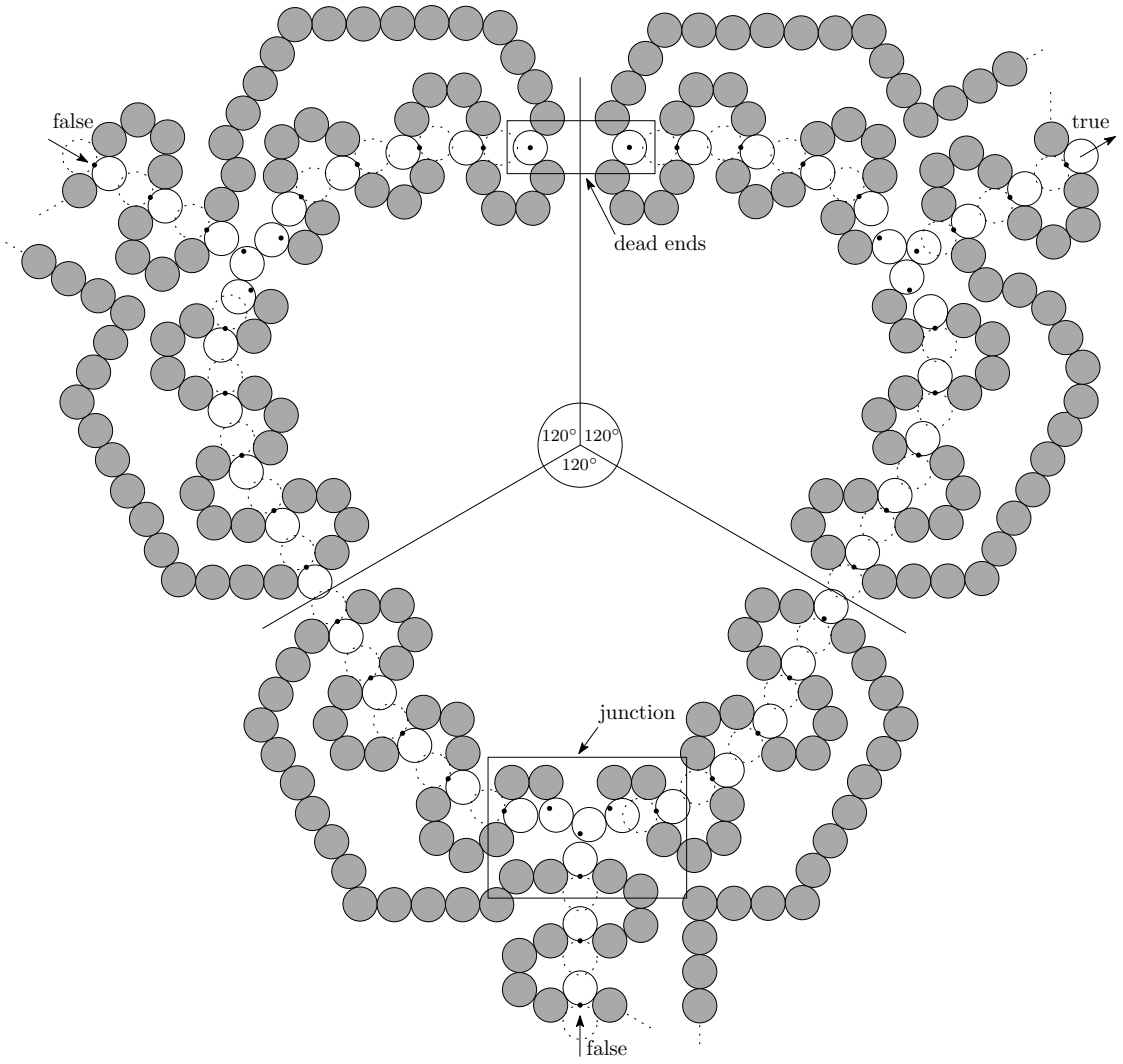


Figure 4.6.: A variable gadget for x in true state.

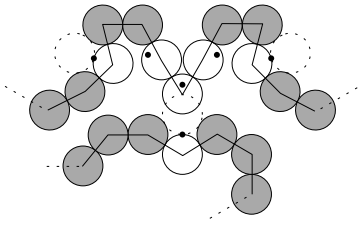
For each variable in U we construct a *variable* gadget which basically is one huge cyclic chain, see Figure 4.6. The cyclic structure of this chain ensures that choosing a location for any of the disks of the inner path dictates the locations for all other disks of the variable gadget. It therefore has exactly two possible states, which are used to reflect the two possible truth states of the corresponding variable. Each variable gadget has a *top* and a *bottom* section and a number of *middle* sections that is linear in the number $|C|$ of clauses (we require $\max\{\deg_l(v), \deg_r(v)\}$ middle sections for the variable gadget corresponding to vertex v in H where $\deg_l$ and $\deg_r$ denote the number of edges connected from the left or right in the drawing of H respectively). Recall that in the drawing of H the variable vertices are organized on a vertical line. Accordingly our variable gadgets are placed on a vertical line and the bottom and top sections are designed such that consecutive gadgets connect to each other. To the top-most variable gadget we add the two disks depicted in black, which ensure that our constructed graph, in the end, is indeed a single path. Recall the each clause of C contains exactly three literals. Accordingly, in the drawing of H each clause vertex is incident to exactly three edges that connect the clause vertex to the corresponding variable vertices. We realize each of these edges as chains which we call *literal* chains. The literal chains connect the clause gadgets to the variable gadgets according to the drawing of H. We attach the literal chains belonging to some variable $x \in U$ to the the sides of the middle sections of the variable gadget for x. More specifically, each side of a middle section contains two designated disk locations, which are labeled x and $\overline{x}$ in Figure 4.6, exactly one of which is occupied by a disk of the variable gadget's inner path in both of the two states. We connect a literal chain such that its disks are pushed away from the variable gadget if the corresponding literal is not satisfied by the truth state that is represented by the variable gadget. For example, the literal chain in the bottom-right corner of Figure 4.6 represents the literal $\overline{x}$ and it is connected such that the left disk location for the inner path's left-most vertex overlaps with a disk location of the variable gadget that is occupied by a disk if the variable gadget for x is in true state (this disk location is marked with x in Figure 4.6). Because of this overlap, the disks of the literal chain are pushed away from the variable gadget and towards its clause gadget, which is explained in the next paragraph. Note that disks of literal chains whose literal is satisfied can be embedded in either direction.

For each clause vertex of H we construct a *clause* gadget at which its three literal chains meet symmetrically at 120° angles, see Figure 4.7a. Each literal chain is connected to a *junction* which again is connected with *junction* chains to the other two junctions. The junctions are designed such that if its literal chain is oriented away from the clause gadget, both adjacent junction chains can be oriented towards the junction, see Figure 4.7b. However, if the literal chain is oriented towards the clause gadget, at least one of the adjacent junction chains has to be oriented away from the junction and therefore towards another junction, see Figure 4.7c and Figure 4.7d. By shifting two seeds of one of the junction chains we generate two dead ends in the sense that the disks of this modified chain are pushed towards both its adjacent junctions, see the top chain in Figure 4.7a. If the literal chain connected to one of these junctions is oriented towards it, the other adjacent chain therefore has to be oriented towards the third junction. Observe that this construction allows at most two literal chains to be oriented towards the clause gadget, otherwise its subgraph can not be realized as a unit disk touching graph that respects our seed assignment.

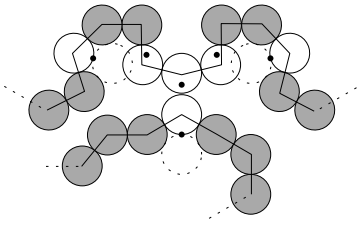
In any realization, each variable gadget has one of two states, corresponding to the two possible truth states. The literal chains are connected such that they are pushed towards their respective clause gadgets if the literal evaluates to false. A clause gadget can be realized if and only if at least one literal chain can be oriented away from it, which corresponds to at least one satisfied literal. The entire construction can be realized if and only if all of



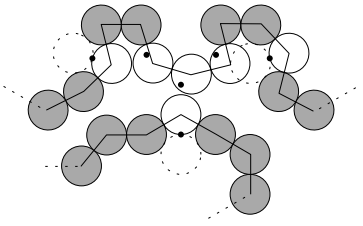
(a) A clause gadget. Two of its literal chains are embedded towards it.



(b) A junction whose literal chain is embedded away from it. Both its junction chains can be embedded towards it.



(c) A junction whose literal chain is embedded towards it. It is not possible to embed both its junction chains towards it.



(d) A junction whose literal chain is embedded towards it. One of its junction chains can be embedded towards it.

Figure 4.7.: Clause gadgets and junctions.

its clause gadgets can be realized, therefore, if at least one of the literals of each clause is satisfied. In conclusion, the construction can be realized if and only if C is satisfiable.

Observe that our construction is indeed a single path (intuitively, we remove the points representing the clause vertices from the drawing of H and traverse along the side of the resulting tree-like structure). The number of disks in a variable gadget is linear in the number of clauses. Each clause gadget contains a constant number of disks. The drawing of H can be realized on a polynomial sized grid and the literal chains resemble lines on this grid. Each literal chain contains a number of disks that is linear in the lengths of the corresponding line (and the length of a chain can be fine-tuned by modifying a constant number of disks). Due to the overlap of the disk locations of the chains' inner paths we do not need infinitely precise coordinates for the seeds and can instead perturb and shift the seeds such that all of them are located on a Cartesian grid (note that the required density of this grid does not depend on the input). When determining the seeds of the gray disks we technically loose the precise UDTS-rigidity property, however, as long as we make sure that we 'round' to the inside such that realizing the gray disks is still possible such that locations of the inner paths' disks still overlap consecutively, the construction with the shifted seeds serves the same purpose as with the precise seeds. Due to the seeds being located on a Cartesian grid of size polynomial in (U, C), the computation of the seeds can be accomplished in polynomial time as well, which concludes our proof. □

Since a path's combinatorial embedding is unique, we furthermore obtain the following.

Corollary 9. *The Unit Disk Touching Graph Recognition with fixed Seeds and Embedding (UDTSE) problem is $\mathcal{NP}$-hard, even for paths.*

5. Conclusion

In this thesis, we explored disk touching graph related recognition problems. It was known that the basic Disk Touching Graph Recognition problem can be solved in linear time due to Koebe's Theorem [Koe36], which implies that the set of planar graphs coincides with the set of graphs that are realizable as disk touching graphs. However, once one tries to dictate the radii or provides seeds that need to be covered, the related recognition problems become $\mathcal{NP}$-hard [BK98, AdCC$^+$12]. We considered several variations and combinations of these basic scenarios and showed that our problems remain $\mathcal{NP}$-hard even for input graph classes that are (much) more basic than planar graphs. We thereby strengthened several old results and provided findings for new interesting problem variations. We summarize our results in the following list. To provide a more complete view, we included very easy, but not necessarily new observations in []-brackets.

- Unit Disk Touching Graph Recognition is $\mathcal{NP}$-hard even for outerplanar graphs (which strengthens a result by [BK98]). For caterpillars, the problem can be solved in linear time. [There exist realizations for any path or cycle. For spiders, the problem can be solved in linear time.]
- Unit Disk Touching Graph Recognition with fixed Embedding is $\mathcal{NP}$-hard, even for outerplanar graphs. [There exist realizations for any path or cycle. For spiders, the problem can be solved in linear time.]
- For spiders the ρ-bounded Disk Touching Graph Recognition problem can be solved in linear time in the Real RAM model for any fixed $\rho \geq 1$.
- Disk Touching Graph Recognition with fixed Radii is $\mathcal{NP}$-hard even for stars (which strengthens a result by [BK98]). There exists a realization for any cycle regardless of the radius assignment. [There exists a realization for any path regardless of the radius assignment.]
- Disk Touching Graph Recognition with fixed Radii and Embedding is $\mathcal{NP}$-hard, even for outerplanar graphs. For stars, the problem can be solved in linear time in the Real RAM model. There exists a realization for any cycle regardless of the radius assignment. [There exists a realization for any path regardless of the radius assignment.]
- Disk Touching Graph Recognition with fixed Seeds is $\mathcal{NP}$-hard even for trees (which strengthens a result by [AdCC$^+$12]).

- Unit Disk Touching Graph Recognition with fixed Seeds is $\mathcal{NP}$-hard, even for paths.
- Unit Disk Touching Graph Recognition with fixed Seeds and Embedding is $\mathcal{NP}$-hard, even for paths.

We achieved closure for some of our problem variations, however, a few open problems remain. A particularly interesting one is the following:

- Is it possible to solve the Unit Disk Touching Graph Recognition problem for trees in polynomial time / Does the Unit Disk Touching Graph Recognition problem remain $\mathcal{NP}$-hard even for trees?

In spite of this open problem, we can conclude our work by stating that disk touching graph related recognition problems are in many cases $\mathcal{NP}$-hard even for basic graph classes, especially when a lot of information is supposed to be encoded in the disks radii or positions. This motivates researching heuristics and approximation algorithms. We have listed some heuristics in Section 1.2, however, little seems to be known about approximation algorithms with guaranteed quality which generate realizations while, for example, allowing a certain percentage of overlap between touching disks or ignoring a certain number of adjacencies.

Bibliography

[AdCC+12] N. Atienza, N. de Castro, C. Cortés, M. Á. Garrido, C. I. Grima, G. Hernández, A. Márquez, A. Moreno-González, M. Nöllenburg, J. R. Portillo, P. Reyes, J. Valenzuela, M. T. Villar, and A. Wolff, "Cover Contact Graphs," *Journal of Computational Geometry*, vol. 3, no. 1, pp. 102–131, 2012.

[AEG+14] M. J. Alam, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and S. Pupyrev, "Balanced Circle Packings for Planar Graphs," *CoRR*, vol. abs/1408.4902, 2014, to appear in Proceedings of Graph Drawing 2014. [Online]. Available: http://arxiv.org/abs/1408.4902

[BK96] H. Breu and D. G. Kirkpatrick, "On the Complexity of Recognizing Intersection and Touching Graphs of Disks," in *Graph Drawing*, ser. Lecture Notes in Computer Science, F. Brandenburg, Ed. Springer Berlin Heidelberg, 1996, vol. 1027, pp. 88–98. [Online]. Available: http://dx.doi.org/10.1007/BFb0021793

[BK98] ——, "Unit Disk Graph Recognition is $\mathcal{NP}$-hard," *Computational Geometry*, vol. 9, no. 1-2, pp. 3–24, Jan. 1998. [Online]. Available: http://dx.doi.org/10.1016/S0925-7721(97)00014-X

[CCJ90] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs," *Discrete Mathematics*, vol. 86, no. 1–3, pp. 165–177, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0012365X9090358O

[CDR07] C. Cabello, E. D. Demaine, and G. Rote, "Planar Embeddings of Graphs with Specified Edge Lengths," *Journal of Graph Algorithms and Applications*, vol. 11, no. 1, pp. 259–276, 2007.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[Coo71] S. A. Cook, "The Complexity of Theorem-proving Procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71. New York, NY, USA: ACM, 1971, pp. 151–158. [Online]. Available: http://doi.acm.org/10.1145/800157.805047

[Dor96] D. Dorling, "Area Cartograms: Their Use and Creation," in *Concepts and techniques in modern geography*. University of East Anglia: Environmental Publications, 1996.

[GJ75] M. R. Garey and D. S. Johnson, "Complexity Results for Multiprocessor Scheduling under Resource Constraints," *SIAM Journal on Computing*, vol. 4, no. 4, pp. 397–411, 1975. [Online]. Available: http://dx.doi.org/10.1137/0204035

[GJ79] ——, *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, 1979.

[Hal80] W. Hale, "Frequency Assignment: Theory and Applications," *Proceedings of the IEEE*, vol. 68, no. 12, pp. 1497–1514, Dec 1980.

[HK01] P. Hliněný and J. Kratochvíl, "Representing graphs by disks and balls (a survey of recognition-complexity results)," *Discrete Mathematics*, vol. 229, no. 1–3, pp. 101–124, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0012365X00002041

[HT74] J. Hopcroft and R. Tarjan, "Efficient Planarity Testing," *Journal of the ACM (JACM)*, vol. 21, no. 4, pp. 549–568, 1974.

[Ino11] R. Inoue, "A New Construction Method for Circle Cartograms," *Cartography and Geographic Information Science*, vol. 38, no. 2, pp. 146–152, 2011. [Online]. Available: http://dx.doi.org/10.1559/15230406382146

[Koe36] P. Koebe, "Kontaktprobleme der konformen Abbildung," in *Berichte über die Verhandlungen der Sächsischen Akadmie der Wissenschaften zu Leipzig. Math.-Phas. Klasse*, vol. 88, 1936, pp. 141–164.

[KR92] D. E. Knuth and A. Raghunathan, "The Problem of Compatible Representatives," *SIAM Journal on Discrete Mathematics*, vol. 5, no. 3, pp. 422–427, 1992.

[Lic82] D. Lichtenstein, "Planar Formulae and their Uses," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 329–343, 1982.

[LW70] D. R. Lick and A. T. White, "k-Degenerate Graphs," *Canadian J. of Mathematics*, vol. 22, pp. 1082–1096, 1970.

[Men27] K. Menger, "Zur allgemeinen Kurventheorie," *Fundamenta Mathematicae*, vol. 10, no. 1, pp. 96–115, 1927. [Online]. Available: http://eudml.org/doc/211191

[PS85] F. P. Preparata and M. I. Shamos, "Computational Geometry, An Introduction," 1985.

[RT90] J.-M. Robert and G. Toussaint, "Computational Geometry and Facility Location," in *Proc. International Conference on Operations Research and Management Science*, 1990, pp. 11–15.

[Wel91] E. Welzl, "Smallest Enclosing Disks (Balls and Ellipsoids)," in *New Results and New Trends in Computer Science*, ser. Lecture Notes in Computer Science, H. Maurer, Ed. Springer Berlin Heidelberg, 1991, vol. 555, pp. 359–370. [Online]. Available: http://dx.doi.org/10.1007/BFb0038202

Appendix

A. Theorem 7

A.1. Calculations for Condition 3.3 and Condition 3.5

In order to show that $\varepsilon_1, \xi_1 = 16/B^2$ and $0 \leq \phi, \psi \leq 1/B^2$ satisfy Condition 3.3 and Condition 3.5, we substitute $y = x \cdot 12/B$ and show that

$$d((16/B^2), (y \cdot B/12)) \leq r(y \cdot B/12) - c/B^2$$

for any $y \in \{3 + 1 \cdot 12/B, 3 + 2 \cdot 12/B, \ldots, 6 - 12/B\}$, any $0 \leq c \leq 1$ and any $B > 12$.

$$\begin{aligned}
d((16/B^2), (y \cdot B/12)) &\leq r(y \cdot B/12) - c/B^2 \Leftrightarrow \\
\sqrt{(r(y \cdot B/12) - 8/B^2)^2 + (r(y \cdot B/12) - r_{min})^2} &\leq r(y \cdot B/12) - c/B^2 \Leftarrow \\
2 \cdot r(y \cdot B/12)^2 - 16 \cdot r(y \cdot B/12)/B^2 + 64/B^4 - 2r(y \cdot B/12)r_{min} + (r_{min})^2 &\leq \\
r(y \cdot B/12)^2 - 2c \cdot r(y \cdot B/12)/B^2 + c^2/B^4 &\Leftrightarrow \\
r(y \cdot B/12)^2 - 16 \cdot r(y \cdot B/12)/B^2 + 64/B^4 - 2r(y \cdot B/12)r_{min} + (r_{min})^2 &\leq \\
-2c \cdot r(y \cdot B/12)/B^2 + c^2/B^4 &\Leftrightarrow \\
(4 + 16/B^2 + y^2/B^2 - 16/B + 4y/B - 8y/B^2) + (-32/B^2 + 64/B^3 - 16y/B^3) + & \\
64/B^4 + (-8 + 16/B - 4y/B + 4/B - 8/B^2 + 2y/B^2 - 48/B^2 + 96/B^3 - 24y/B^3) + & \\
(4 + 1/B^2 + 144/B^4 - 4/B + 48/B^2 - 24/B^3) + (4c/B^2 - 8c/B^3 + 2cy/B^3) - c^2/B^4 \leq 0 &\Leftrightarrow \\
(208 - c^2)/B^4 + (64 - 16y + 96 - 24y - 24 - 8c + 2cy)/B^3 + & \\
(16 + y^2 - 8y - 32 - 8 + 2y - 48 + 1 + 48 + 4c)/B^2 \leq 0 &\Leftrightarrow \\
(208 - c^2)/B^4 + (136 - 40y + 2cy - 8c)/B^3 + (-23 + y^2 - 6y + 4c)/B^2 \leq 0 &\Leftarrow \\
208 - c^2 + (136 - 40y + 2cy - 8c)B + (-23 + y^2 - 6y + 4c)B^2 \leq 0 &\Leftarrow \\
208 - 0^2 + (136 - 40 \cdot 3 + 2 \cdot 1 \cdot 6 - 8 \cdot 0)B + (-23 + 0 + 4 \cdot 1)B^2 \leq 0 &\Leftrightarrow \\
208 + 28B - 19B^2 \leq 0 &
\end{aligned}$$

The last inequality clearly holds true for any $B > 12$, which concludes the proof.

A.2. Calculations for Condition 3.2

In order to show that $0 < \varepsilon \leq 17/B^2$ is a sufficient choice to satisfy Condition 3.2, we assign $\varepsilon = c/B^2$ and show that the condition holds for any $0 < c \leq 17$ and for any $B > 12$.

$$
\begin{aligned}
12 + \varepsilon \leq s_i \Leftrightarrow \\
12 + c/B^2 \leq 2r_{min} + 2\sqrt{(r_{max} + r_{min})^2 - (r_{max} - r_{min})^2} \Leftrightarrow \\
3 + (c/4)/B^2 - (1/2)r_{min} \leq \sqrt{r_{max} r_{min}} \Leftarrow \\
(3 + (c/4)/B^2 - (1/2)(2 - 1/B + 12/B^2))^2 \leq (2 - 1/B + 12/B^2)(2 + 2/B - 12/B^2) \Leftrightarrow \\
9 + (c^2/16)/B^4 + 1 + 1/(4B^2) + 36/B^4 - 1/B + 12/B^2 - 6/B^3 + 3c/(2B^2) - 6 + \\
3/B - 36/B^2 - c/(2B^2) + c/(4B^3) - 3c/B^4 \leq 4 + 2/B - 2/B^2 + 36/B^3 - 144/B^4 \Leftrightarrow \\
(c^2/16 - 3c + 180)/B^4 + (c/4 - 42)/B^3 + (c - 87/4)/B^2 \leq 0 \Leftarrow \\
c^2/16 - 3c + 180 + (c/4 - 42)B + (c - 87/4)B^2 \leq 0 \Leftarrow \\
17/16 - 3 \cdot 0 + 180 + (17/4 - 42)B + (17 - 87/4)B^2 \leq 0 \Leftrightarrow \\
17/16 + 180 - (151/4)B - (19/4)B^2 \leq 0
\end{aligned}
$$

The last inequality clearly holds true for any $B > 12$, which concludes the proof.

A.3. Calculations for Condition 3.4

In order to show that $0 < \xi \leq 17/B^2$ is a sufficient choice to satisfy Condition 3.4, we assign $\xi = c/B^2$ and show that the condition holds for any $0 < c \leq 17$ and for any $B > 12$.

$$
\begin{aligned}
12 - 24/B^2 + \xi \leq s_f \Leftrightarrow \\
12 - 24/B^2 + c/B^2 \leq 2r(B/4) + 2\sqrt{(r(B/2) + r(B/4))^2 - (r(B/2) - r(B/4))^2} \Leftrightarrow \\
12 + (c - 24)/B^2 - 2r(B/4) \leq 4\sqrt{r(B/2) \cdot r(B/4)} \Leftrightarrow \\
2 + (c/4 - 6)/B^2 + (1/2)/B \leq \sqrt{(2 + 2/B) \cdot (2 - 1/B)} \Leftarrow \\
4 + (c^2/16 - 3c + 36)/B^4 + (1/4)/B^2 + (c - 24)/B^2 + 2/B + (c/4 - 6)/B^3 \leq \\
4 - 2/B + 4/B - 2/B^2 \Leftrightarrow \\
(c^2/16 - 3c + 36)/B^4 + (c/4 - 6)/B^3 + (c - 24 + 1/4 + 2)/B^2 \leq 0 \Leftarrow \\
c^2/16 - 3c + 36 + (c/4 - 6)B + (c - 22 + 1/4)B^2 \leq 0 \Leftarrow \\
17^2/16 - 3 \cdot 0 + 36 + (17/4 - 6)B + (17 - 22 + 1/4)B^2 \Leftrightarrow \\
289/16 + 36 - (7/4)B - (19/4)B^2 \leq 0
\end{aligned}
$$

The last inequality can easily be verified to be true for any $B > 12$, which concludes the proof.

A.4. Upper bounds for Condition 3.2 and Condition 3.4

We show that $s \leq 1/B^2$ for any $d \leq 1/B^2$ and for any $B > 12$.

$$
\begin{aligned}
s \leq 1/B^2 \Leftrightarrow \\
2(\sqrt{(r(B/2)+r(B/4))^2-(r(B/2)-r(B/4))^2} \\
-\sqrt{(r(B/2)+r(B/4))^2-(r(B/2)+d-r(B/4))^2}) \leq 1/B^2 \Leftrightarrow \\
2(\sqrt{4\cdot r(B/2)\cdot r(B/4)} \\
-\sqrt{4\cdot r(B/2)\cdot r(B/4)-2d\cdot r(B/2)+2d\cdot r(B/4)-d^2}) \leq 1/B^2 \Leftrightarrow \\
\sqrt{4\cdot r(B/2)\cdot r(B/4)}-(1/2)/B^2 \\
\leq \sqrt{4\cdot r(B/2)\cdot r(B/4)-2d\cdot r(B/2)+2d\cdot r(B/4)-d^2} \Leftarrow \\
(1/4)/B^4-\sqrt{4\cdot r(B/2)\cdot r(B/4)}/B^2 \leq 2d\cdot r(B/4)-2d\cdot r(B/2)-d^2 \Leftrightarrow \\
(1/4)/B^2+2dB^2\cdot(2+2/B)-2dB^2\cdot(2-1/B)+d^2B^2 \leq \sqrt{4\cdot(2+2/B)\cdot(2-1/B)} \Leftarrow \\
((1/8)/B^2+dB^2(2+2/B-2+1/B)+(1/2)\cdot d^2B^2)^2 \leq (2+2/B)\cdot(2-1/B) \Leftarrow \\
((5/8)/B^2+1\cdot(3/B))^2 \leq 4-2/B+4/B-2/B^2 \Leftrightarrow \\
(25/64)/B^4+(30/8)/B^3+9/B^2 \leq 4-2/B+4/B-2/B^2 \Leftrightarrow \\
(25/64)/B^4+(15/4)/B^3+11/B^2-2/B-4 \leq 0 \Leftarrow \\
25/64+(15/4)B+11B^2-2B^3-4B^4 \leq 0
\end{aligned}
$$

The last inequality clearly holds true for any $B > 12$, which concludes the proof.