DISCRETIZATION BASED SOLUTION APPROACHES FOR THE CIRCLE PACKING PROBLEM

by
RABIA TAŞPINAR

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Sciences

Sabancı University
July 2021

DISCRETIZATION BASED SOLUTION APPROACHES FOR THE CIRCLE PACKING PROBLEM

Approved by:

Approval: July 14, 2021

ABSTRACT

DISCRETIZATION BASED SOLUTION APPROACHES FOR THE CIRCLE PACKING PROBLEM

RABIA TAŞPINAR

INDUSTRIAL ENGINEERING M.S. THESIS, 2021

Thesis Supervisor: Assist. Prof. BURAK KOCUK

Keywords: circle packing problem, discretization, global optimization

In this thesis, we study the Circle Packing Problem (CPP), which involves packing a set of circles into the smallest surrounding container. This problem can be cast as a nonconvex quadratically constrained quadratic program with reverse convex constraints, which is difficult to solve in general. The CPP arises in different application areas such as automobile, textile, food, and chemical industries. For this problem, we propose an iterative solution approach based on a bisection-type algorithm on the radius of the surrounding container. At each iteration, the algorithm solves two different mixed-integer linear programming formulations proposed for a restricted and a relaxed version of the original problem over the discretized surrounding container. In the restricted version, the centers of the circles can only be located at the candidate points in each cell. By changing some properties of the restricted version, we also construct a relaxed version of the original problem. We also proposed an enhancement to the algorithm that exploits the special structure of CPP. We perform a computational study in order to evaluate the performance of our algorithm. We compare the results from our algorithm with those from commercial global optimization solvers BARON and Gurobi that solve the original nonlinear formulation, and also with the results of other methods from the literature. In addition, we examine a case study from a real life problem from the automobile industry, which is solved to a small optimality gap.

ÖZET

ÇEMBER PAKETLEME PROBLEMİ İÇİN
AYRIKLAŞTIRMA TEMELLİ ÇÖZÜM YAKLAŞIMLARI

RABIA TAŞPINAR

ENDÜSTRİ MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, 2021

Tez Danışmanı: Dr. BURAK KOCUK

Anahtar Kelimeler: çember paketleme problemi, ayrıklaştırma, küresel eniyileme

Bu tezde, verilen bir grup çemberi içeren en küçük konteynerin bulunmasını amaçlayan çember paketleme problemi ele alınmıştır. Çember paketleme problemi konveks olmayan ikinci dereceden kısıtlı karesel programlama problemi olarak ele alınabilir ve zor bir problem olarak bilinir. Bu problem gıda, otomotiv, tekstil ve kimya sektörleri gibi çok farklı alanlarda uygulamalara sahiptir. İlgili problem için konteynerin yarıçapı üzerinden yarılama metoduna dayalı yinelemeli bir çözüm yöntemi geliştirilmiştir. Algoritmanın her adımında problemin kısıtlanmış ve gevşetilmiş versiyonu için ayrıklaştırılmış konteyner üzerinde tanımlanmış iki farklı karma tamsayılı doğrusal programlama gösterimi oluşturulmuştur. Kısıtlanmış versiyonda, orijinal problemde verilen çemberlerin merkezlerinin yerleştirilebileceği sürekli alan ayrıklaştırma sonucu elde edilen karelerin köşe noktalarına kısıtlanmıştır. Benzer şekilde, çember merkezlerinin aday kareler tarafından içerildiği varsayılarak gevşetilmiş versiyon inşa edilmiştir. Ayrıca, çember paketleme probleminin kendine özgü özellikleri kullanılarak bazı analitik çıkarımlar yapılmış ve geliştirilen yönteminin elde edilen bilgilerle beslenmesi sağlanmıştır. Geliştirdiğimiz algoritmanın başarımının belirlenmesi için, literatürdeki örnekler kullanılarak bir deneysel çalışma gerçekleştirilmiş ve elde edilen sonuçlar raporlanmıştır. Deneysel çalışmada, geliştirilen çözüm yönteminin başarım analizi için orijinal problem için geliştirilen doğrusal olmayan programlama gösterimi BARON ve Gurobi çözücüleri kullanılarak çözdürülmüş ve algoritmayla karşılaştırılmıştır. Ayrıca, otomotiv endüstrisinden bir gerçek hayat problemi için gerçekleştirilen çalışma detaylandırılmıştır.

ACKNOWLEDGEMENTS

First and foremost I am extremely grateful to my supervisor, Dr. Burak Kocuk for his invaluable advice, continuous support, and patience during my master study. I would like to express my sincere gratitude to Assoc. Prof. Dr. Kadir Ertoğral to be a part of my thesis committee and for his insightful comments and questions on my thesis. I would also like to thank my other committee member Prof. Dr. Ali Rana Atılgan for his insightful comments.

This thesis is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) 1002 Program under grant #120M345, as well as I received graduate scholarship from 2210-E Program of TÜBİTAK during my master study. Hence, I would like to thank to TÜBİTAK for this support to me and to my thesis.

Besides, I thank my friend Büşra for her encouragement, for her friendship, and for all the fun we had in last few years in Sabanci University. Also, I thank Seher, Pelin, Furkan and Onur for their friendship, for their help, for the stimulating discusses and for all the time we spent together at TOBB University of Economics and Technology. I am also grateful to my friends Sevde Nur, Ayşe, Sinem and Melike for their support, for their encouragement and for all the time we had fun together.

Finally, I would like to express my gratitude to my friends, my parents, my sisters and my nephews and my niece. Without their tremendous understanding and encouragement, it would be impossible for me to complete my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$\mathcal{C}$	set of circles to be packed
n	the number of circles to be packed
$\overline{\mathcal{C}}$	set of circles to be packed and the surrounding container, $\overline{\mathcal{C}} = \mathcal{C} \cup \{0\}$
R	the radius of the surrounding circle
(x_c, y_c)	the coordinates of the center of circle c
r_c	the radius of circle c
δ	the side length of each cell in discretization
(θ, θ)	the representation of the guiding point in discretization corresponding to the origin
$\mathcal{I}$	set of corner points of the cells in discretization, $\lvert\mathcal{I}\rvert = 2\theta + 1$
$\mathcal{L}_c$	set of candidate points for locating circle c
$z_{i,j,c}$	$z_{i,j,c} = 1$ if the center of circle c is located at a candidate point denoted by (i, j), 0 otherwise
$(a_{i,c}, b_{j,c})$	$a_{i,c} = 1$ and $b_{j,c} = 1$ if the center of circle c is located at a candidate point denoted by (i, j)
$\mathcal{N}_{c,k}$	set of the pairs of the minimum required number of cells on x and y axes between the centers of the circles c and k to avoid overlapping of these circles
$\pi_{u_1,u_2,c,k}$	$\pi_{u_1,u_2,c,k} = 1$ if the centers of circles c and k are located at two points which have at least u_1 cells on x-axis, and u_2 cells on y-axis between them; 0 otherwise
$\mathrm{w}_{i,j,c}$	$\mathrm{w}_{i,j,c} = 1$ if the center of circle c is included by the corresponding region of the candidate point denoted by (i, j), 0 otherwise
$\mathcal{S}_c$	set of the candidate points whose corresponding cells include at least one point where circle c can be located
$(q_{i,c}, p_{j,c})$	$q_{i,c} = 1$ and $p_{j,c} = 1$ if the center of circle c is included by the corresponding region of the candidate point denoted by (i, j), 0 otherwise

$\mathcal{O}_{c,k}$	set of the pairs of the minimum required number of cells on x and y axes to locate the circles c and k without overlapping if circles c and k are located to the farthest points of the corresponding two cells
$\Pi_{u_1,u_2,c,k}$	$\Pi_{u_1,u_2,c,k}=1$ if the centers of the circles c and k are located at two points included by two cells which have at least u_1 cells on x-axis, and u_2 cells on y-axis between the farthest two points of the corresponding cells; 0 otherwise
$\mathcal{D}_{(3.1)}$	set of all feasible solutions of the problem (3.1)
$\mathcal{D}_{(3.5)}$	set of all feasible solutions of the formulation (3.5)
$\mathcal{D}_{(4.2)}$	set of all feasible solutions of the problem (4.2)
Θ	$\Theta=\lceil\log_2(2\theta)\rceil$
α_c	vector of binary decision variables corresponding to the $x-$axis position of the candidate point where the center of the circle c is located
β_c	vector of binary decision variables corresponding to the $y-$axis position of the candidate point where the center of the circle c is located
$\mathcal{F}_c$	subset of $\mathcal{L}_c$ used for determining all of the points in $\mathcal{L}_c$, $\mathcal{F}_c=\{(g,h)\in\mathcal{L}_c : g,h\geq 0,(g,h+1)\notin\mathcal{L}_c,(g+1,h)\notin\mathcal{L}_c\}$
$\psi_{i,j,c}$	$\psi_{i,j,c}=1$ if the center of the circle c is located at the point (g,h) such that $2\theta-i\leq g\leq i$ and $2\theta-j\leq h\leq j$ where $(g,h)\in\mathcal{L}_c$, 0 otherwise
γ_c	vector of binary decision variables corresponding to the $x-$axis position of the left-lower corner point of the region including the center of the circle c
ω_c	vector of binary decision variables corresponding to the $y-$axis position of the left-lower corner point of the region including the center of circle c
$\mathcal{M}_c$	subset of $\mathcal{S}_c$ used for determining all of the points in $\mathcal{S}_c$, $\mathcal{M}_c=\{(g,h)\in\mathcal{S}_c : g,h\geq 0,(g,h+1)\notin\mathcal{S}_c,(g+1,h)\notin\mathcal{S}_c\}$
$\eta_{i,j,c}$	$\eta_{i,j,c}=1$ if the center of the circle c is located at the point (g,h) such that $2\theta-i\leq g\leq i$ and $2\theta-j\leq h\leq j$ where $(g,h)\in\mathcal{S}_c$, 0 otherwise
U	upper bound for the radius of the surrounding container
L	lower bound for the radius of the surrounding container
$d_{c,k,l}$	$d_{c,k,l}=1$ if the circles c,k, and l are adjacent to each other, 0 otherwise for $c,k,l\in\overline{\mathcal{C}} : c<k<l$
$f_{c,k,l}$	$f_{c,k,l}=1$ if one of the three rays passing through the center of circle c (or k or l) and the center of surrounding circle is included by the region between the other two rays, 0 otherwise for $c,k,l\in\mathcal{C} : c<k<l$
$\Delta_{c,k,l}$	a parameter corresponding to the idle area between the circles c,k, and l

$\rho_{c,k,l}$	the amount of reduction in the idle area if $f_{c,k,l} = 1$, i.e., $\rho_{c,k,l} = \Delta_{0,c,k} + \Delta_{0,k,l} + \Delta_{c,k,l} + \pi(r_k)^2$
$\min_c$	the minimum number of triples circle c can have as neighbours
$\max_c$	the maximum number of triples circle c can have as neighbours

1. INTRODUCTION

The circle packing problem (CPP) is a well-known NP-Hard problem with a wide range of applications in different areas such as automobile, material cutting, nanotechnology, wireless communication, and food industries. CPP is concerned with packing a given number of different circles into a larger container. The shape of the container can be a square, a rectangle, or a circle. The circles should be packed into the container in such a way that circles do not overlap and each circle is entirely in the container. Different objectives are considered depending on the application setting in the literature including minimizing the area of the surrounding container, maximizing the number of circles packed into a fixed-size container, or maximizing the minimum distance between any two circles.

A well-known application area of CPP is packing circular shaped objects such as bottles, cans, reels, or sheet roles into the smallest box (Castillo, Kampas & Pintér, 2008). Another example can be planting trees in a specific area while trying to maximize the forest density within this area (Hifi & M'Hallah, 2009). Also, one can consider facilities as circular areas in a facility location problem (Castillo et al., 2008). Locating undesirable facilities (e.g., missile silos or nuclear facilities) or fast-food franchises in a fixed-area while maximizing the minimum distance is also a specific version of CPP. Another example can be detecting the optimal placement of modules in a facility such that the cost of interactions is minimized (Castillo & Sim, 2010). Also, cutting circular objects from a larger plate while trying to increase material efficiency can be considered as a version of circle packing (Dowsland, Gilbert & Kendall, 2007; Hifi & M'Hallah, 2004).

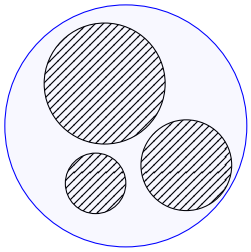
Moreover, determining the dimensions of cables, layers or pipes containing smaller ones with different diameters used by telecommunication/electrical/oil companies is another application of CPP (Wang, Huang, Zhang & Xu, 2002). A similar example from the automobile industry is estimating the size of the hole on the body of the car from which a bundle of wires passes to connect the car's sensors to the display board (Sugihara, Sawai, Sano, Kim & Kim, 2004). There are also some example applications in nanotechnology, for instance, incorporating silver nanoparticles into

the 3D silica colloidal film while trying to maximize the silver density (Li & Sun, 2009).

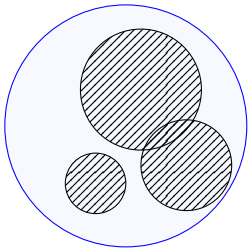
In this thesis, we consider CPP with the objective of minimizing the radius of the surrounding container. Under the feasibility rules of CPP, Figure 1.1a shows a feasible suboptimal solution for an instance of the problem consisting of three circles with radii 0.5, 0.75 and 1.0 units where the surrounding container's radius is 2 units. Two possible infeasible configurations are also given in Figure 1.1: Figure 1.1b includes two overlapping circles violating the non-overlapping feasibility rule; and Figure 1.1c is an example where a circle is not fully contained in the container.

Figure 1.1 Example feasible and infeasible placements of circles for CPP.

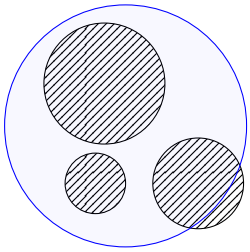
(a) A feasible suboptimal solution for CPP.



(b) An infeasible solution where two circles overlap.



(c) An infeasible solution with a partially included circle.



Within this framework, we propose an iterative solution approach based on a bisection-type algorithm to solve the CPP considering the minimum radius objective. Our solution procedure relies on discretizing the container into smaller cells, and iteratively solves two mixed integer linear programming (MILP) formulations designed for a restricted and a relaxed versions of the original problem. This allows us to utilize commercial MILP solvers in order to certify upper and lower bounds for the minimum radius of the surrounding container efficiently.

The remainder of the thesis is organized as follows. In Chapter 2, we review the literature on different classifications of CPP. In Chapter 3, we give the precise non-convex quadratically constrained quadratic programming formulation of the CPP as well as the mixed integer linear programming formulations proposed for the restricted and relaxed versions of the original problem. In Chapter 4, we explain the proposed iterative solution approach based on a bisection-type algorithm considering the minimum radius objective as well as the algorithmic enhancements designed for this algorithm. Then, Chapter 5 presents a computational study analyzing the performance of our algorithm and the proposed MILP formulations. Also, we introduce a solution method to solve a real-life instance with 162 circles from automobile industry as a case study (Sugihara et al., 2004). Finally, Chapter 6

concludes this thesis.

2. LITERATURE REVIEW

As stated before, the objective of CPP depends on the application setting and is typically one of the following: "minimizing the radius of the surrounding container", "maximizing the number of circles packed into a fixed-size container", or "maximizing the minimum distance between any two circles". In Section 2.1, we will briefly summarize the circle packing literature considering the objectives mentioned above, except for minimizing the radius of the surrounding container objective or its equivalents. Then, we will summarize the studies considering the minimization of the surrounding container's radius objective, or equivalent ones in Section 2.2. The vast majority of the CPP literature focuses on developing heuristic methods. In particular, most of the studies include a heuristic algorithm designed for first constructing a feasible packing of the circles, and then improving it with a search algorithm, see e.g. Francesco, Cerrone & Cerulli (2014); Huang, Li, Jurkowiak, Li & Xu (2003); Huang, Li, Li & Xu (2006); Zeng, Yu, He, Huang & Fu (2016).

2.1 Circle Packing Literature Considering Other Objectives

In this section, we will review the related literature considering different objectives such as maximizing the minimum pairwise distance, maximizing the number of circles placed in a fixed-size container. A common objective in CPP is maximizing the minimum pairwise distance between circles (Drezner & Erkut, 1995; Locatelli & Raber, 2002; Stoyan & Yaskov, 2012; Szabó, 2000). Drezner & Erkut (1995) compare the problem of packing p circles into a fixed-size container by maximizing pairwise distances between circles and the p-dispersion problem, and proves that these problems are equivalent (p-dispersion problem is the problem of locating p-points such that pairwise distance between them is maximized). Maranas, Floudas & Pardalos (1995) formulate this problem as a max-min optimization problem, and

generates initial solutions to span a large part of parameter space since their solver (MINUS 5.3) does not guarantee global optimality. For selecting the initial points, the unit square is divided into a number of equal rectangles whose sides are equal or almost equal, and then, uniformly distributed random points are generated inside every rectangle. These points are initial candidate points for placing the centers of the circles to construct a solution. Then, the proposed formulation for the max-min optimization problem is solved for locating n circles through the initial candidate points to obtain possible configurations. Locatelli & Raber (2002) examine the structure of optimal solutions of the same problem.

For the same problem, Szabó (2000) uses a grid structure in the stochastic algorithm they propose. In the algorithm, the unit square is divided into sub-rectangles. Then, the center of the first circle is placed at one of corners of the rectangles, and others are placed to the cross corners of the corresponding rectangles one by one. This study also proves that locating n equal circles into a square while maximizing the circles' radius are equivalent to putting n circles in a square such that the minimum distance between any pair of them is maximized.

Maximizing the number of circles placed in a fixed-size container is another objective considered in CPP. Galiev & Lisafina (2013) try to pack the maximum number of equal circles into a given circle. In this study, a set of candidate points is constructed where the centers of the circles to be placed at, after a construction procedure; then, a maximal number of circles is determined by considering the candidate points. López & Beasley (2016) consider the CPP where the objective is maximizing the value of packed circles. If the values of circles are equal; then, the objective is equivalent to maximizing the number of circles. The proposed solution method starts with a number of candidate points where the centers of circles are located, and tries to identify the subset of circles to be packed. This algorithm iteratively solves a formulation packing the circles to the candidate points and a relaxed version which is a modified linear programming relaxation with some additional constraints. It is also proved that an optimal solution for the maximizing the number of circles consists of the first K circles (for some K) if the circles are listed according to their radii in an increasing order.

Litvinchev & Ozuna (2014) study the same problem by using an integer programming formulation. In their work, the surrounding container is divided into cells, and the centers of cells are considered as candidate points to locate circles' centers. Two valid inequalities are given for the introduced integer programming formulation, and five linear programming relaxations are constructed involving different subsets of constraints. Then, both the integer programming and linear programming relaxations

are solved by using CPLEX 12.5. This study considers both the original CPP and the nested CPP in which packing a circle into another packed circle is possible. In a follow-up study, Litvinchev, Infante & Ozuna Espinosa (2015b) present a new format for the constraints to eliminate some of the constraints proposed in a previous study (Litvinchev & Ozuna, 2014). Another set of valid inequalities are also introduced in Litvinchev, Infante & Ozuna Espinosa (2015a). These valid inequalities are a constraint set that ensures each grid point is covered by only one circle and similarly a new set of constraints to eliminate the nested ones proposed by Litvinchev & Ozuna (2014).

In the literature, another objective consideered is minimizing the total weighted pairwise distance between circles. An example of such studies is (Castillo & Sim, 2010). In this study, the authors of the paper formulate the facility layout problem as a circle packing problem where the departments are circular shaped, and this problem is formulated as an unconstrained optimization problem by using the augmented Lagrangian Multiplier method. The proposed method starts with a penalty multiplier and an initial multiplier vector, and updates these parameters by using a procedure within the solution method. The solution is found by iteratively solving the unconstrained problem with different penalty multipliers and multiplier vectors. Layouts for different instances including up to 30 departments are given in this paper.

Torres, Marmolejo & Litvinchev (2020) consider CPP where the objective is maximizing the weighted profit of packed circles; and the authors formulate this problem as an integer programming. Then, a binary monkey algorithm, an algorithm designed for the knapsack problems, is presented to solve this problem. The binary monkey algorithm is a population-based algorithm. It starts with an initial population, and then the algorithm tries to improve the initial solution by using different methods: climbing process (in which two random vectors are used to generate two new populations from the initial population), watch-jump process (in which the better solutions are detected within the neighborhood of the current population). For the proposed algorithm, the surrounding container is divided into grids, and the algorithm tries to locate the centers of circles to the centers of grids. This study proposes that the final solutions obtained from the algorithm can be used as initial solutions for the exact solution methods to eliminate some of the initial steps of optimization process.

2.2 Literature on Minimizing the Radius of the Surrounding Container

In the literature, there are different studies considering the same problem as ours, or the equivalent maximum density problem which is the problem of maximizing the density of circles in a surrounding circle (Hifi & M'Hallah, 2009). Most studies include an algorithm designed for first constructing a feasible packing of the circles, and then improving the feasible solution with a search algorithm. Huang et al. (2003) proposes such an algorithm including two-level search strategy for packing unequal circles. In this study, the authors assume that m of n circles are already placed into the circular container, and then the remaining $(n-m)$ circles are packed. Any feasible packing of an unpacked circle is defined as a "legal action", and the legal action with maximum benefit is selected to pack the corresponding unpacked circle accordingly into the last configuration.

Moreover, Huang, Li, Akeb & Li (2005) and Huang et al. (2006) use the same method to pack unequal circles into rectangular and circular containers, respectively. The common method in these two studies consists of two algorithms: a basic heuristic based on the maximal hole degree rule, and a self-look-ahead strategy. Similar to a previous study (Huang et al., 2003), the method starts with an initial configuration with a subset of circles, and the first part of this method is also the same. However, the rest of the method is different. There is a new term, "corner placement", which is the term used for the circle with at least two neighbor circles; and the method selects the configuration for an unpacked circle which corresponds to the corner placement with the maximum benefit. Sugihara et al. (2004) study a circle packing problem arising in the automotive industry where design engineers should drill holes on the body of the automobile through which a bundle of wires passes to connect the sensors to the display board. These holes should be large enough for the bundle of wires to pass, however, unnecessary largeness of these holes will weaken the body of the automobile unnecessarily. In this study, the problem of estimating the smallest size of the hole is examined.

There are also some studies performing Tabu search to solve CPP. For instance, Huang, Zeng, Xu & Fu (2012) introduce a population based solution approach. The algorithm starts with a member of the initial population including the best k-solution, and perturbs it by adaptive Tabu search, and updates the population accordingly. As another study, Francesco et al. (2014) propose another algorithm based on a multi-start technique in which the starting solutions are obtained by a Tabu search algorithm. In this paper, different instances containing from 5 to 50 circles are solved,

and the results are compared with the best known solutions which show that the gaps are less than 1% for all instances.

Zeng et al. (2016) perform a Tabu Search and Variable Neighborhood Descent procedure. The procedure assigns a value for each solution, named as the potential energy, and compares solutions by using the energy value and the radius of the surrounding circle for any solution. There are three phases of this approach: continuous optimization, local search, and diversification. Instances with up to 50 circles are solved, and the results are compared with the best known solutions. The authors manage to improve the results in some of them.

Another study uses a reduced gradient method, the concept of active inequalities and the Newton method to pack circles into a strip by starting with different extreme points (Stoyan & Yas'kov, 2004). For a square container, an action-space-based global optimization method is introduced in another study, (He, Huang & Yang, 2015). Huang & Ye (2011) propose a quasi-physical global optimization method for the problem of packing a given set of equal circles into a larger circle.

Among all the discussed literature, none of the algorithms verifies the optimality of the solutions obtained. The quality of the solutions is determined by only comparing them against the best-known results from the literature since none of the algorithms proposes a lower bound for the radius of the surrounding container. According to our preliminary research, it is hard to determine a tight lower bound for the CPP. In this study, we introduce a lower bound for the radius of the container as well as improve it during the iterations of our proposed algorithm.

3. PROBLEM FORMULATION

In this chapter, we discuss the formulations introduced for the CPP. Within this chapter, we assume that the surrounding container is circular-shaped; however, our approach can be applied to other container shapes as well. First, we introduce the precise mathematical formulation of CPP. We denote the set of circles by $\mathcal{C} = \{1, \cdots, n\}$, and the radius of circle $c \in \mathcal{C}$ by r_c. We assume, without loss of generality, that the center of the surrounding circle is located at the origin, and R is a decision variable denoting its radius. The center of circle c is represented by the decision variables (x_c, y_c). Then, CPP can be modelled as the following non-convex quadratically constrained program:

$$\begin{aligned}
&\text{(3.1a)} && \min\ R && \\
&\text{(3.1b)} && \text{s.t. } (x_c - x_k)^2 + (y_c - y_k)^2 \geq (r_c + r_k)^2 && c, k \in \mathcal{C} :\ c \neq k \\
&\text{(3.1c)} && x_c^2 + y_c^2 \leq (R - r_c)^2 && c \in \mathcal{C} \\
&\text{(3.1d)} && x_c,\, y_c,\, R \in \mathbb{R} && c \in \mathcal{C}.
\end{aligned}$$

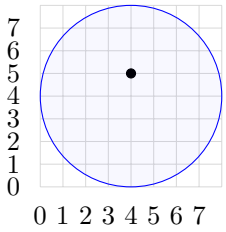
The objective of the problem (3.1) is the minimization of the size of the surrounding circle. Constraint (3.1b) ensures that circles c and k do not overlap. Constraint (3.1c) satisfies that circle c is totally included by the surrounding circle where Constraint (3.1d) is the domain constraint for the decision variables.

In an optimal solution of the problem (3.1), the centers of circles can be located in anywhere in $\mathbb{R}^2$ space. However, this problem is hard to solve in the continuous space by using the global solvers, such as BARON and Gurobi. Hence, we discretized the continuous solution space into smaller squares in order to design our algorithm. Then, we constructed a restricted and a relaxed versions of the problem (3.1). MILP formulations are introduced based on the discretization of the surrounding circle into smaller squares for the restricted and relaxed versions whose details are given in the following sections.

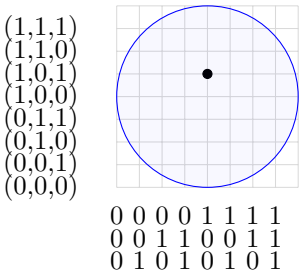
The corner points of the squares are the guiding points within the MILP formulations to assign the center of each circle c, and these assignments can be represented by utilizing linear or logarithmic number of binary decision variables in the size of the problem.

Figure 3.1 The representation of the guiding points with different number of decision variables.

(a) Linear number of variables.



(b) Logarithmic number of variables.



To clarify the definition of these variables, consider the example given in Figure 3.1. If these points are represented by using linear number of binary decision variables like given in Figure 3.1a, then, two decision variables are defined for each row and each column number, i.e., totally 14 decision variables are defined to represent these points for each circle. For instance, if a circle is located at the black point in Figure 3.1a, then, the decision variables defined for row 5 and column 4 are assigned as one where the others are assigned as zero. On the other hand, totally 6 binary decision variables are defined for each circle for the example given in Figure 3.1b by using logarithmic number of decision variables. According to this representation, the center of the corresponding circle is assigned to the black point if the corresponding six binary decision variables are assigned as ((1,0,1);(1,0,0)).

According to the representation of these points, we will provide two types of MILP formulations for CPP based on the representation of the points resulting from discretizing the surrounding circle into small square cells: i) linear-sized discretization formulations given in Section 3.1 (including linear number of decision variables and constraints in the number of cells), and ii) logarithmic-sized discretization formulations introduced in Section 3.2 (including logarithmic number of decision variables in the number of cells and linear number of constraints).

Each discretization scheme is accompanied by a restricted and a relaxed version under the assumption that a candidate radius for the surrounding circle is given as R. In the restricted version, we will consider the corners of each cell as candidate points to locate the centers of circles. If feasible, this solution will give a feasible solution for CPP, and an upper bound of at most R for the radius of the surrounding

circle in problem (3.1). In the relaxed version, we will let the centers of circles to be located at any point in a cell. In this version, we will allow any pair of circles to overlap in a well-defined and limited manner so that the resulting model gives a systematic way to construct a relaxation for CPP. If infeasible, the relaxed model will give a lower bound of R for the radius of the surrounding circle in problem (3.1).

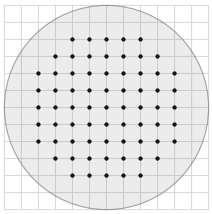
We will now give the details of each formulation in the following sections.

3.1 Linear-Sized Discretization Formulations

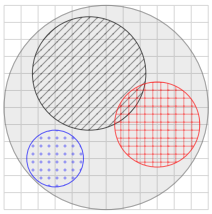
In this formulation, we will first divide the smallest square containing the surrounding circle into small square-shaped cells. The corners of these cells are "guiding points" to construct the formulations. The side length of each cell is given by the parameter δ, and we assume that the diagonal size of each cell is smaller than the radius of the smallest circle, i.e., $\delta\sqrt{2} < \min\{r_1, \cdots, r_n\}$. If a corner of any cell is out of the surrounding circle, we can eliminate this point from our set of candidate points since the size of each cell is smaller than each circle. We give an example discretization for a circular container in Figure 3.2a.

Figure 3.2 Example discretizations of the container and example placements of circles.

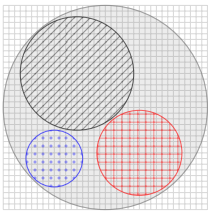
(a) Candidate points in discretization.



(b) An infeasible configuration.



(c) A feasible configuration with smaller cells.



Remark 3.1. *There are two possible definitions of binary variables to represent the guiding points: (i) introducing a binary variable for each guiding point, (ii) defining a pair of binary variables to represent x-axis and y-axis positions of each guiding point. Our preliminary experiments have revealed that the formulations constructed by the second approach are better in terms of solution times than the naïve formulations designed by the first approach.*

According to the interpretation of the guiding points stated in Remark 3.1, we

will introduce linear-sized formulations for the restricted and relaxed versions of problem (3.1).

3.1.1 Restricted Version

In this formulation, we treat the guiding points as candidate points to locate the centers of circles. We model the cells as square-shaped objects. We define the set $\mathcal{I}$ to represent the corners of the cells on the x-axis (as well as on the y-axis) where $\mathcal{I} = \{0, 1, \cdots, 2\theta\}$ and $|\mathcal{I}| = 2\theta + 1$. The lower left corner of the discretized region is denoted by $(0, 0)$. To simplify the notation, the center of the surrounding circle is assumed to be at the origin of the coordinate system which is represented by (θ, θ). Hence, the coordinates of the candidate point represented by (i, j) are given with $((i-\theta)\delta, (j-\theta)\delta)$. Also, we select the parameters θ and δ such that $\theta\delta = R$. Then, we have a total of $(2\theta+1)^2$ guiding points within the surrounding circle with radius R, and a subset of these points are candidate points to locate the center of circle $c \in \mathcal{C}$. An example illustration of the introduced notations is demonstrated in Figure 3.2c, where $\delta = 0.1$, $\theta = 18$, and $\mathcal{I} = \{0, 1, \cdots, 36\}$ with corresponding axis coordinates $\{-1.8, \cdots, 1.7, 1.8\}$, and the smallest circle is located at the candidate point represented by $(9, 9)$ at $(-0.9, -0.9)$.

Let us illustrate the main features of the formulation with an example. By using the candidate points, subsets of the bullets in Figure 3.2a, we can obtain a restricted formulation which correctly models all constraints of the original problem. In Figures 3.2b and 3.2c, we will try to pack three circles with radii 1 unit, 0.75 units, 0.5 units, respectively, into a circle with radius 1.8 units. The cell sizes in these figures are 0.3 and 0.1 units, respectively. Depending on the granularity of the discretization, it may not be possible to find any feasible configuration for a given radius for the surrounding circle as given in Figure 3.2b. On the other hand, a finer grid with smaller cells must enable to find a feasible solution for the original problem, as in Figure 3.2c, as long as the original problem is feasible on $\mathbb{R}^2$.

As we stated in Remark 3.1, we introduce two different perspectives on defining the binary decision variables to represent each candidate point. In the first perspective, a binary variable $z_{i,j,c}$ is defined where it is assigned one if the center of circle c is located at the candidate point (i, j), $i, j \in \mathcal{I}$ for any circle $c \in \mathcal{C}$. It is clear that the number of decision variables increases linearly with the number of cells, hence, the name of the formulation.

Any two circles cannot be placed at two candidate points if the corresponding circles overlap, i.e., the corresponding binary decision variables cannot be one at the same time which will be banned in the proposed formulation. Moreover, to conclude with a feasible configuration for a given circle c, we know that the smallest distance of any candidate point to the boundary of the surrounding circle should be at least r_c units. Then, the corresponding subset of such candidate points for circle c are denoted by set $\mathcal{L}_c$, i.e., $\mathcal{L}_c = \{(i,j) \in \mathcal{I}^2 : ((\theta - i)\delta)^2 + ((\theta - j)\delta)^2 \leq (R - r_c)^2\}$. Hence, we can eliminate the candidate points not included by $\mathcal{L}_c$ in this formulation. Then, the naïve formulation including the binary variables $z_{i,j,c}$ is given as follows:

$$\text{(3.2a)} \qquad \sum_{(i,j)\in\mathcal{L}_c} z_{i,j,c} = 1 \quad c \in \mathcal{C}$$

$$\text{(3.2b)} \qquad z_{i,j,c} + z_{g,h,k} \leq 1 \quad c,k \in \mathcal{C}; (i,j) \in \mathcal{L}_c, (g,h) \in \mathcal{L}_k : \left\| \begin{pmatrix} i-g \\ j-h \end{pmatrix} \right\|_2 < \frac{r_c + r_k}{\delta}$$

$$\text{(3.2c)} \qquad z_{i,j,c} \in \{0,1\} \qquad c \in \mathcal{C}, (i,j) \in \mathcal{L}_c.$$

All circles are packed into the surrounding circle with the help of Constraint (3.2a). Constraint (3.2b) ensures the non-overlapping packing of the circles. Constraint (3.2c) is the domain constraint for the decision variables $z_{i,j,c}$.

In the second perspective mentioned in Remark 3.1, we define a pair of binary decision variables, $a_{i,c}$ and $b_{j,c}$, to denote the x-axis and the y-axis positions to represent the candidate points. The decision variable $a_{i,c}$ ($b_{j,c}$) is assigned one if the center of circle c is located at a candidate point whose x-axis (y-axis) position is denoted by i (j). Then, if $a_{i,c}$ and $b_{j,c}$ take value one, it means that the circle c's center is located at the candidate point (i,j). In this formulation, the number of decision variables increases linearly with the sum of the number of cells on x and y axes. However, the number of decision variables decreases with this perspective. For instance, the number of decision variables is 18 instead of 69 for the discretization given in Figure 3.2a.

For ensuring the feasibility of the original problem, the overlap of two circles should be banned. The candidate points to place the centers of the circles c and k such that $c < k$ should be far enough to avoid the overlap of these circles. We can focus on the pairs of the minimum required number of cells on x and y axes between the centers of the circles c and k to avoid overlapping of these circles. Hence, we will define another set, $\mathcal{N}_{c,k}$, defined as follows:

$$\mathcal{N}_{c,k} = \Big\{(u_1,u_2) : u_1,u_2 \in \mathcal{I},\, u_1,u_2 \geq 0,\, (u_1\,\delta)^2 + (u_2\,\delta)^2 \geq (r_c + r_k)^2,$$
$$\Big(u_1\,\delta - \delta\Big)^2 + (u_2\,\delta)^2 < (r_c + r_k)^2,\, (u_1\,\delta)^2 + \Big(u_2\,\delta - \delta\Big)^2 < (r_c + r_k)^2\Big\}.$$

In other words, the set $\mathcal{N}_{c,k}$ includes the pairs (u_1, u_2) such that the diagonal of the rectangle with side lengths u_1 and u_2 is larger than the number $(r_c + r_k)/\delta$. In addition to this, there is no other rectangles which is totally included by the rectangle with side lengths u_1 and u_2 with a diagonal size larger than $(r_c + r_k)/\delta$.

In Figure 3.2b, assume that the largest circle is denoted by index 1, the second largest circle is denoted by index 2, and the smallest one is denoted by index 3. Example illustration of sets $\mathcal{N}_{1,2}, \mathcal{N}_{1,3}, \mathcal{N}_{2,3}$ can be given as follows for Figure 3.2b: $\mathcal{N}_{1,2} = \{(0,6),(4,5),(5,4),(6,0)\}, \mathcal{N}_{1,3} = \{(0,5),(3,4),(4,3),(5,0)\}, \mathcal{N}_{2,3} = \{(0,5),(2,4),(3,3),(4,2),(5,0)\}$. With the help of the set $\mathcal{N}_{c,k}$, we can eliminate the set of pairs of the guiding points where the centers of circles c and k cannot be assigned at the same time.

Before introducing the corresponding formulation, we need one more binary variable: $\pi_{u_1,u_2,c,k}$. With the help of $\pi_{u_1,u_2,c,k}$ and the set $\mathcal{N}_{c,k}$, we can ensure that any two circles are non-overlapping by stating the sum of all such variables is equal to 1 for each $c,k \in \mathcal{C} : c < k$, and the decision variable $\pi_{u_1,u_2,c,k}$ can be assigned one if the centers of circles c,k are located at two points which have at least u_1 cells on x-axis, and u_2 cells on y-axis between them, i.e., $(u_1,u_2) \in \mathcal{N}_{c,k}$

Then, formulation (3.3) including the binary decision variables $a_{i,c}$, $b_{j,c}$ and $\pi_{u_1,u_2,c,k}$ is given as follows:

$$\text{(3.3a)} \qquad \sum_{i \in \mathcal{I} : \exists j, (i,j) \in \mathcal{L}_c} a_{i,c} = 1 \qquad c \in \mathcal{C}$$

$$\text{(3.3b)} \qquad \sum_{j \in \mathcal{I} : \exists i, (i,j) \in \mathcal{L}_c} b_{j,c} = 1 \qquad c \in \mathcal{C}$$

$$\text{(3.3c)} \qquad \sum_{i \in \mathcal{I}} i\, a_{i,c} - \sum_{g \in \mathcal{I}} g\, a_{g,k} \geq \sum_{(u_1,u_2) \in \mathcal{N}_{c,k}} u_1\, \pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c < k$$

$$\text{(3.3d)} \qquad \sum_{j \in \mathcal{I}} j\, b_{j,c} - \sum_{h \in \mathcal{I}} h\, b_{h,k} \geq \sum_{(u_1,u_2) \in \mathcal{N}_{c,k}} u_2\, \pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c < k$$

$$\text{(3.3e)} \qquad \sum_{(u_1,u_2) \in \mathcal{N}_{c,k}} \pi_{u_1,u_2,c,k} = 1 \qquad c,k \in \mathcal{C}, c < k$$

$$\text{(3.3f)} \qquad a_{i,c} \in \{0,1\} \qquad c \in \mathcal{C}, i \in \mathcal{I}$$

$$\text{(3.3g)} \qquad b_{j,c} \in \{0,1\} \qquad c \in \mathcal{C}, j \in \mathcal{I}$$

$$\text{(3.3h)} \qquad \pi_{u_1,u_2,c,k} \in \{0,1\} \qquad (u_1,u_2) \in \mathcal{N}_{c,k}.$$

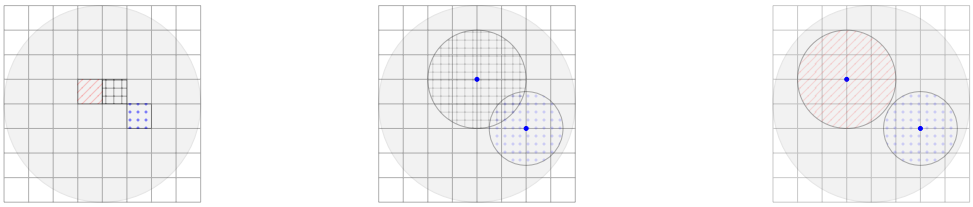
All circles are packed into the surrounding circle with the help of Constraints (3.3a) and (3.3b). Constraints (3.3c)-(3.3e) and the sets $\mathcal{N}_{c,k}$ are introduced for ensuring the non-overlapping packing of the circles. Constraints (3.3f)-(3.3h) are domain constraints for the decision variables.

3.1.2 Relaxed Version

In Section 3.1.1, if a circle is assigned to a candidate point, it means that the center of corresponding circle is located exactly at the corresponding candidate point. In relaxed version, assigning a circle to a candidate point is interpreted differently from the interpretation given in Section 3.1.1: Assigning a circle to a candidate point means locating its center to a point included by the region denoted with the candidate point. The region represented with a candidate point is the cell whose lower-left corner point is the corresponding candidate point.

Figure 3.3 Example placements of circles for the relaxed version.

(a) Candidate regions. (b) An infeasible configuration. (c) A feasible configuration.



For instance, the candidate regions indicated with dashed lines are denoted by their lower-left corner points, i.e., the points at $(3, 4)$, $(4, 4)$, and $(5, 3)$, respectively from left to right in Figure 3.3a. However, we cannot identify the exact location of the center of any circle even if we know the containing cell in this version.

The constructed relaxed version will cover all feasible solutions for the original problem as well as a subset of infeasible solutions; however, we limit the subset of allowed infeasible solutions. If two circles do not have any feasible configuration for two candidate regions, locating the circles to the corresponding candidate regions is banned in this formulation. This is guaranteed as follows: we eliminate locating two circles to two candidate regions even if the farthest distance between these two regions is less than the sum of the radii of the corresponding circles to exclude such infeasible configurations.

As an example, we can look at the candidate regions given in Figure 3.3a. If we consider locating the largest circle to the candidate region at the middle and the second largest circle to the candidate region at the rightmost, it is also an infeasible configuration for the proposed relaxed version since the farthest distance of these regions (1.58 units) is smaller than the sum of their radii (1.75 units). However, if the candidate region containing the center of the largest circle is changed to the candidate region at the leftmost, then it will be feasible for the relaxed version since the farthest distance between the corresponding regions (1.8 units) is greater than the sum of their radii. Although the introduced relaxed version prevents some explicit infeasible configurations, it only considers two circles at the same time. Hence, placing all circles without any overlapping may not be possible for a feasible solution of the relaxed version.

By considering two different perspectives stated in Remark 3.1, we propose two different formulations for the relaxed version as well. In the first perspective, we introduce the binary decision variable $\mathrm{w}_{i,j,c}$ similar to the variable $z_{i,j,c}$. If the center of circle c is contained in the cell denoted by the point (i,j), then, $\mathrm{w}_{i,j,c}$ will be assigned one. We also define the set $\mathcal{S}_c$ as follows:

$$\mathcal{S}_c = \left\{ i,j \in \mathcal{I} : \left\| \begin{pmatrix} i-\theta \\ j-\theta \end{pmatrix} \right\|_2 \leq \frac{R-r_c}{\delta} \text{ or } \left\| \begin{pmatrix} i-\theta+1 \\ j-\theta+1 \end{pmatrix} \right\|_2 \leq \frac{R-r_c}{\delta} \right\}.$$

In the definition of $\mathcal{S}_c$, we investigate the coordinates of the cell (i,j)'s closest point to the center of the surrounding circle. More precisely, the set $\mathcal{S}_c$ includes the points corresponding to the cells with at least one point included by this cell whose distance to the boundary of the surrounding circle is at least r_c. Then, the corresponding formulation containing $\mathrm{w}_{i,j,c}$ is given below:

$$\text{(3.4a)} \quad \sum_{(i,j)\in\mathcal{S}_c} \mathrm{w}_{i,j,c} = 1 \qquad c \in \mathcal{C}$$

$$\text{(3.4b)} \quad \mathrm{w}_{i,j,c} + \mathrm{w}_{g,h,k} \leq 1 \quad c,k \in \mathcal{C}; (i,j) \in \mathcal{S}_c, (g,h) \in \mathcal{S}_k : \left\| \begin{pmatrix} |i-g|+1 \\ |j-h|+1 \end{pmatrix} \right\|_2 < \frac{r_c+r_k}{\delta}$$

$$\text{(3.4c)} \quad \mathrm{w}_{i,j,c} \in \{0,1\} \qquad c \in \mathcal{C}, (i,j) \in \mathcal{S}_c.$$

Similarly, Constraint (3.4a) ensures that all circles are packed. The overlapping of circles is restricted by Constraint (3.4b) where Constraint (3.4c) is domain constraint for decision variable $\mathrm{w}_{i,j,c}$.

As stated in Remark 3.1, formulation (3.4) is the naïve formulation. By using the

second perspective, we introduce the binary decision variables $q_{i,c}$ and $p_{j,c}$ similar to the variables $a_{i,c}$ and $b_{j,c}$. If the center of circle c is contained by the candidate region denoted by the point (i,j), then, the variables $q_{i,c}$ and $p_{j,c}$ will be assigned one. Similarly, the number of decision variables also increases linearly with the number of cells in these formulations.

By using a similar idea with the restricted version, we define another set $\mathcal{O}_{c,k}$ similar to $\mathcal{N}_{c,k}$ for the pairs of circles c,k. The set $\mathcal{O}_{c,k}$ includes the pairs of the minimum required number of cells where the farthest distance of the cells between the centers of circles c,k are bigger than the sum of their radii. This set is defined as follows:

$$\mathcal{O}_{c,k} = \Big\{(u_1,u_2) : u_1,u_2 \in \mathcal{I}\backslash\{\theta\},\, u_1,u_2 \geq 0,\, (u_1+1)^2+(u_2+1)^2 \geq ((r_c+r_k)/\delta)^2 \\ (u_1\delta)^2+((u_2+1)\delta)^2 < (r_c+r_k)^2,\, ((u_1+1)\delta))^2+(u_2\delta)^2 < (r_c+r_k)^2\Big\}.$$

The defined set $\mathcal{O}_{c,k}$ includes the pairs (u_1,u_2) such that the diagonal of the rectangle with side lengths (u_1+1) and (u_2+1) is larger than $(r_c+r_k)/\delta$. Similarly, there is no other rectangles with diagonal length larger than $(r_c+r_k)/\delta$ which is totally included in this rectangle. Also, if centers of circles $c,k \in \mathcal{C}$ are included in two cells whose farthest distance is at least u_1 cells on x-axis and u_2 cells on y-axis, the corresponding proposed variable $\Pi_{u_1,u_2,c,k}$ will be one. With the help of the sets $\mathcal{O}_{c,k}$ and the variables $\Pi_{u_1,u_2,c,k}$, we eliminate the obvious infeasible solutions of CPP. In the second approach, the resulting formulation is as follows:

$$\text{(3.5a)} \qquad \sum_{i\in\mathcal{I}:\,\exists j,(i,j)\in\mathcal{S}_c} q_{i,c} = 1 \qquad c \in \mathcal{C}$$

$$\text{(3.5b)} \qquad \sum_{j\in\mathcal{I}:\,\exists i,(i,j)\in\mathcal{S}_c} p_{j,c} = 1 \qquad c \in \mathcal{C}$$

$$\text{(3.5c)} \qquad \left|\sum_{i\in\mathcal{I}} i\, q_{i,c} - \sum_{g\in\mathcal{I}} g\, q_{g,k}\right| \geq \sum_{(u_1,u_2)\in\mathcal{O}_{c,k}} u_1\Pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c<k$$

$$\text{(3.5d)} \qquad \left|\sum_{j\in\mathcal{I}} j\, p_{j,c} - \sum_{h\in\mathcal{I}} h\, p_{h,k}\right| \geq \sum_{(u_1,u_2)\in\mathcal{O}_{c,k}} u_2\Pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c<k$$

$$\text{(3.5e)} \qquad \sum_{(u_1,u_2)\in\mathcal{O}_{c,k}} \Pi_{u_1,u_2,c,k} = 1 \qquad c,k \in \mathcal{C} : c<k$$

$$\text{(3.5f)} \qquad q_{i,c} \in \{0,1\} \qquad c \in \mathcal{C},\, i \in \mathcal{I}$$

$$\text{(3.5g)} \qquad p_{j,c} \in \{0,1\} \qquad c \in \mathcal{C},\, j \in \mathcal{I}$$

$$\text{(3.5h)} \qquad \Pi_{u_1,u_2,c,k} \in \{0,1\} \qquad (u_1,u_2) \in \mathcal{O}_{c,k}.$$

Constraints (3.5a) and (3.5b) ensure that all circles are packed into the surrounding

container. Since any two circles can intersect, Constraints (3.5c)-(3.5e) restrict the allowable intersection by using the same idea which forbids to place circles to two cells whose maximum distance is less than the sum of the radii of corresponding circles.

Now, we will prove that this formulation is a relaxed version of the original problem. Assume that the set of all feasible solutions of the problem (3.1) as $\mathcal{D}_{(3.1)}$, and those of formulation (3.5) as $\mathcal{D}_{(3.5)}$.

Theorem 3.1. *For each feasible solution in $\mathcal{D}_{(3.1)}$, there is a corresponding feasible solution in $\mathcal{D}_{(3.5)}$.*

Proof. Take an arbitrary solution $\{\forall c \in \mathcal{C}, (x_c^*, y_c^*)\} \subset \mathcal{D}_{(3.1)}$ of the problem (3.1). Construct a solution for formulation (3.5) such that if $i_c^* \leq x_c^* < (i_c^* + 1)$ and $j_c^* \leq y_c^* < (j_c^* + 1)$, then assign $p_{i_c^*,c}$ and $q_{j_c^*,c}$ one for each circle $c \in \mathcal{C}$.

Consider circles c and k with coordinates (x_c^*, y_c^*) and (x_k^*, y_k^*) in the feasible solution of the problem (3.1). In the constructed solution for formulation (3.5), $p_{i_c^*,c} = 1$, $q_{j_c^*,c} = 1$ such that $i_c^* \leq x_c^* < (i_c^* + 1)$ and $j_c^* \leq y_c^* < (j_c^* + 1)$; and $p_{g_k^*,k} = 1$, $q_{h_k^*,k} = 1$ such that $g_k^* \leq x_k^* < (g_k^* + 1)$ and $h_k^* \leq y_k^* < (h_k^* + 1)$, and the remaining variables are assigned zero. Hence, Constraints (3.5a)-(3.5b) are satisfied.

The differences between corresponding coordinates are as follows:

$$i_c^* - g_k^* - 1 < x_c^* - x_k^* < i_c^* - g_k^* + 1 \implies x_c^* - x_k^* < |i_c^* - g_k^*| + 1,$$

$$j_c^* - h_k^* - 1 < y_c^* - y_k^* < j_c^* - h_k^* + 1 \implies y_c^* - y_k^* < |j_c^* - h_k^*| + 1.$$

Since the farthest distance between the candidate regions denoted by the points (i_c^*, j_c^*) and (g_k^*, h_k^*) will be $\|(|i_c^* - g_k^*| + 1, |j_c^* - h_k^*| + 1)\|_2$, there is a pair $(u_1, u_2) \in \mathcal{O}_{c,k}$ such that $i_c^* - g_k^* \geq u_1, j_c^* - h_k^* \geq u_2$. For the corresponding (u_1, u_2) value, assign $\Pi_{u_1,u_2,c,k}$ as one. Then, Constraints (3.5c)-(3.5e) are satisfied.

Since the circles c and k are selected arbitrarily, the same results are valid for any two circles. As a result, the constructed solution is also feasible for formulation (3.5). Since the feasible solution of formulation (3.1) is arbitrarily selected, then, there is a corresponding feasible solution of the formulation (3.5) for any feasible solution of the problem (3.1). □

By Theorem 3.1, formulation (3.5) is a relaxed version of problem (3.1).

3.2 Logarithmic-Sized Discretization Formulations

In this section, we define the logarithmic version of decision variables to decrease the number of decision variables as introduced in (Vielma, Ahmed & Nemhauser, 2010a) and (Vielma, Ahmed & Nemhauser, 2010b). In the reformulation, we will define a parameter; Θ is such that $\Theta = \lceil \log_2 (2\theta) \rceil$. We will now introduce the logarithmic-sized formulations corresponding to the restricted and the relaxed versions of problem (3.1).

3.2.1 Restricted Version

In this version, we define two vectors of binary decision variables corresponding to the candidate points defined in Section 3.1: α_c and β_c. The binary column vector corresponds to the corner point of cell on x-axis (y-axis) in base 2 where the center of circle c is located is denoted by $\alpha_c = (\alpha_{c,1}, \alpha_{c,2}, \cdots, \alpha_{c,\Theta})$ ($\beta_c = (\beta_{c,1}, \beta_{c,2}, \cdots, \beta_{c,\Theta})$) for the circle $c \in \mathcal{C}$. We also introduce another set $\mathcal{F}_c$ defined as follows:

$$\mathcal{F}_c = \{(g,h) \in \mathcal{L}_c : g,h \geq 0, (g,h+1) \notin \mathcal{L}_c, (g+1,h) \notin \mathcal{L}_c\}.$$

The set $\mathcal{F}_c$ is defined for determining all of the points in $\mathcal{L}_c$ with the help of a subset of the points in $\mathcal{L}_c$. This set is defined with another decision variable $\psi_{i,j,c}$ for ensuring that the circle c is totally included by the surrounding circle. Assume that the center of circle c is located at the point (g,h). Then, the decision variable $\psi_{i,j,c}$ is assigned value of one if $2\theta - i \leq g \leq i, 2\theta - j \leq h \leq j$, i.e., $(g,h) \in \mathcal{L}_c$. In other words, the center of circle c is assigned to a point included in the set $\mathcal{L}_c$, i.e., the circle c is fully contained in the surrounding circle. Then, the corresponding formulation is given as follows:

$$\text{(3.6a)} \quad \sum_{t=1}^{\Theta} 2^{(t-1)} (\alpha_{c,t} - \alpha_{k,t}) \geq \sum_{(u_1,u_2) \in \mathcal{N}_{c,k}} u_1 \, \pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c < k$$

$$\text{(3.6b)} \quad \sum_{v=1}^{\Theta} 2^{(v-1)} (\beta_{c,v} - \beta_{k,v}) \geq \sum_{(u_1,u_2) \in \mathcal{N}_{c,k}} u_2 \, \pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c < k$$

$$\text{(3.6c)} \quad \sum_{(u_1,u_2) \in \mathcal{N}_{c,k}} \pi_{u_1,u_2,c,k} = 1 \qquad c,k \in \mathcal{C} : c < k$$

$$\text{(3.6d)} \quad \left| \sum_{t=1}^{\Theta} 2^{(t-1)}\, \alpha_{c,t} \right| \leq \sum_{(i,j)\in\mathcal{F}_c} i\, \psi_{i,j,c} \qquad c \in \mathcal{C}$$

$$\text{(3.6e)} \quad \left| \sum_{v=1}^{\Theta} 2^{(v-1)}\, \beta_{c,v} \right| \leq \sum_{(i,j)\in\mathcal{F}_c} j\, \psi_{i,j,c} \qquad c \in \mathcal{C}$$

$$\text{(3.6f)} \quad \sum_{(i,j)\in\mathcal{F}_c} \psi_{i,j,c} = 1 \qquad c \in \mathcal{C}$$

$$\text{(3.6g)} \quad \alpha_c, \beta_c \in \{0,1\}^{\Theta} \qquad c \in \mathcal{C}$$

$$\text{(3.6h)} \quad \pi_{u_1,u_2,c,k} \in \{0,1\} \qquad c,k \in \mathcal{C}, (u_1,u_2) \in \mathcal{N}_{c,k}$$

$$\text{(3.6i)} \quad \psi_{i,j,c} \in \{0,1\} \qquad (i,j) \in \mathcal{F}_c,\ c \in \mathcal{C}.$$

In this formulation, Constraints (3.6a)-(3.6c) satisfy the non-overlapping feasibility rule, and Constraints (3.6d)-(3.6f) ensure that each circle is fully contained by the surrounding circle. Finally, Constraints (3.6g)-(3.6i) are the domain restrictions for the decision variables.

As an example illustration of the notation, we can look at the point denoted by $(5,8)$ where the center of the largest circle is located in Figure 3.2b. Then, the decision variables α_1 and β_1 are assigned as $\alpha_1 = (1,0,1,0)$ and $\beta_1 = (0,0,0,1)$ where $\theta = 6$ and $\Theta = \lceil \log_2 12 \rceil = 4$. Also, the decision variables $\psi_{7,8,1} = 1$ and $\psi_{8,7,1} = 0$ since the center of circle 1 is assigned to the point $(5,8)$ where the set $\mathcal{F}_1 = \{(7,8),(8,7)\}$.

Theorem 3.2. *Formulation* (3.3) *is equivalent to formulation* (3.6).

Proof. Assume that $\mathcal{D}_{(3.6)}$ is the set of feasible solutions of formulation (3.6).

For the forward direction, start from a feasible solution of formulation (3.3) where the center of circle c is located at the point (i_c^*, j_c^*) for each circle $c \in \mathcal{C}$. Then, $a_{i_c^*,c} = 1, b_{j_c^*,c} = 1$ and the remaining decision variables are assigned zero for each circle c. Then, construct a solution for formulation (3.6) by selecting the decision variables α_c, β_c for each circle c as follows:

$$\text{(3.7a)} \quad \sum_{i\in\mathcal{I}} i\, a_{i,c} = i_c = \left(\sum_{t=1}^{\Theta} 2^{(t-1)}\, \alpha_{c,t} \right),$$

$$\text{(3.7b)} \quad \sum_{j\in\mathcal{I}} j\, b_{j,c} = j_c = \left(\sum_{v=1}^{\Theta} 2^{(v-1)}\, \beta_{c,v} \right).$$

In addition, assign the same values in formulation (3.6) for the variables $\pi_{u_1,u_2,c,k}$ given in formulation (3.3). Since locating circles c and k to candidate points at

(i_c^*, j_c^*) and (i_k^*, j_k^*) is feasible, Constraints (3.6a)-(3.6c) are satisfied by the equalities (3.7a) and (3.7b). Moreover, since $(i_c^*, j_c^*) \in \mathcal{L}_c$ for each circle c, assign $\psi_{g_c^*, h_c^*, c} = 1$ such that $g_c^* = \max\{i_c^*, 2\theta - i_c^*\}$ and $h_c^* = \max\{j_c^*, 2\theta - j_c^*\}$; and the remaining ones as zero. Then, by the definition of the sets $\mathcal{L}_c$'s and $\mathcal{M}_c$'s, Constraints (3.6d)-(3.6f) are satisfied. Then, the constructed solution is feasible for formulation (3.6).

For the backward direction, start from a feasible solution for formulation (3.6) where the vectors α_c^*, β_c^* are given, and $\pi_{u_1^*, u_2^*, c, k} = 1, \psi_{g_c^*, h_c^*, c} = 1$ for each circle c. Calculate (i_c^*, j_c^*) satisfying the equalities (3.7a) and (3.7b) for $\alpha_c = \alpha_c^*, \beta_c = \beta_c^*$. As a result, assign one to the decision variables $a_{i_c^*, c}, b_{j_c^*, c}$ corresponding to the coordinates (i_c^*, j_c^*) and zero to other variables in the constructed solution. Then, $(i_c^*, j_c^*) \in \mathcal{L}_c$ in the constructed solution by the definition of variables $\psi_{g_c^*, h_c^*, c}$ and Constraints (3.6d)-(3.6f) which means that Constraints (3.3a) - (3.3b) are satisfied. Finally, use the same values for the variables $\pi_{u_1^*, u_2^*, c, k}$ in the constructed solution as given in formulation (3.6). From the feasibility of $\alpha_c^*, \beta_c^*, \alpha_k^*, \beta_k^*, \pi_{u_1^*, u_2^*, c, k}$ for circles c and k, Constraints (3.3c)-(3.3e) are satisfied. Hence, the constructed solution is feasible for formulation (3.3).

Thus, these formulations are equivalent. □

3.2.2 Relaxed Version

For introducing this formulation, we define two decision variables by using a similar idea with the one used in Section 3.1. The decision variable $\gamma_c = (\gamma_{c,1}, \gamma_{c,2}, \cdots, \gamma_{c,\Theta})$ ($\omega_c = (\omega_{c,1}, \omega_{c,2}, \cdots, \omega_{c,\Theta})$) denotes the x-axis (y-axis) coordinate of the left-lower corner point of the cell in base 2 where the center of circle c is contained. For instance, if circle c's center is included by the leftmost region given in Figure 3.3a, the corresponding variables are assigned as $\gamma_c = (1, 1, 0, 0)$ and $\omega_c = (0, 0, 1, 0)$). We also define a set $\mathcal{M}_c$ as follows:

$$\mathcal{M}_c = \{(g, h) \in \mathcal{S}_c : g, h \geq 0, (g, h+1) \notin \mathcal{S}_c, (g+1, h) \notin \mathcal{S}_c\}.$$

The set $\mathcal{M}_c$ is defined for determining all of the points in $\mathcal{S}_c$ with the help of a subset of points in $\mathcal{S}_c$. By forming a rectangle $\{(i, j) \in \mathcal{I}^2 : 2\theta - g \leq i \leq g, 2\theta - h \leq j \leq h\}$ for each $(g, h) \in \mathcal{M}_c$, the union of the rectangles contain all the points given in $\mathcal{S}_c$. We also define another variable $\eta_{i,j,c}$ ensuring that there is at least one point of the cell where the center of circle c is located. Assume that the center of circle c is included by the cell corresponding to the point (g, h). Then, $\eta_{i,j,c}$ is assigned one

if (g,h) is included by the rectangle corresponding to the point $(i,j) \in \mathcal{M}_c$. The corresponding formulation is as follows:

$$\sum_{t=1}^{\Theta} 2^{(t-1)}(\gamma_{c,t} - \gamma_{k,t}) \geq \sum_{(u_1,u_2)\in\mathcal{O}_{c,k}} u_1 \, \Pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c < k \tag{3.8a}$$

$$\sum_{v=1}^{\Theta} 2^{(v-1)}(\omega_{c,v} - \omega_{k,v}) \geq \sum_{(u_1,u_2)\in\mathcal{O}_{c,k}} u_2 \, \Pi_{u_1,u_2,c,k} \qquad c,k \in \mathcal{C} : c < k \tag{3.8b}$$

$$\sum_{(u_1,u_2)\in\mathcal{O}_{c,k}} \Pi_{u_1,u_2,c,k} = 1 \qquad c,k \in \mathcal{C} : c < k \tag{3.8c}$$

$$\left| \sum_{t=1}^{\Theta} 2^{(t-1)} \, \gamma_{c,t} \right| \leq \sum_{(i,j)\in\mathcal{M}_c} i \, \eta_{i,j,c} \qquad c \in \mathcal{C} \tag{3.8d}$$

$$\left| \sum_{v=1}^{\Theta} 2^{(v-1)} \, \omega_{c,v} \right| \leq \sum_{(i,j)\in\mathcal{M}_c} j \, \eta_{i,j,c} \qquad c \in \mathcal{C} \tag{3.8e}$$

$$\sum_{(i,j)\in\mathcal{M}_c} \eta_{i,j,c} = 1 \qquad c \in \mathcal{C} \tag{3.8f}$$

$$\gamma_c, \omega_c \in \{0,1\}^{\Theta} \qquad c \in \mathcal{C} \tag{3.8g}$$

$$\Pi_{u_1,u_2,c,k} \in \{0,1\} \qquad c,k \in \mathcal{C}, (u_1,u_2) \in \mathcal{O}_{c,k} \tag{3.8h}$$

$$\eta_{i,j,c} \in \{0,1\} \qquad (i,j) \in \mathcal{M}_c, \; c \in \mathcal{C}. \tag{3.8i}$$

Similarly, in this model, Constraints (3.8a)-(3.8c) satisfy the non-overlapping feasibility condition partially while allowing some intersections. Constraints (3.8d)-(3.8f) ensure that there is at least one point of the corresponding cell for locating the center of the circle such that it is fully contained by the surrounding circle. Finally, Constraints (3.8g)-(3.8i) are the domain constraints for the decision variables.

This problem is introduced for the relaxed version of problem (3.1), since the decision variables determines the lower-left corners of the cells where the centers of circles are located, and the exact locations of them is unknown. As a result, it is possible for circles c and k to overlap in this formulation like formulation (3.5) by the definition of the set $\mathcal{O}_{c,k}$.

Theorem 3.3. *Formulation* (3.5) *is equivalent to formulation* (3.8).

Proof. By using the same steps in Theorem 3.2, we can show that the relaxed version of formulation (3.5) is equivalent to formulation (3.8). □

4. SOLUTION METHODS AND ENHANCEMENTS

In this chapter, we discuss our solution methods designed for the CPP. In the following section, we will introduce a bisection-type solution algorithm which iteratively consider different values for the radius of the surrounding circle. Our solution algorithm solves the restricted and relaxed formulations alternately by using the discretized circle while utilizing the MILP models in Chapter 3. We will also present our algorithmic enhancements, which significantly improves the performance of our solution approach.

4.1 Discretized-Space Circle Packer (DCPACK)

In this section, we will explain our solution procedure. DCPACK algorithm depends on updating the upper and lower bounds for the radius of the surrounding circle until they are close enough. In this method, we should start by using initial upper and lower bounds for the surrounding circle. To initialize these bounds, there are two possible situations. If we know a surrounding circle in which all circles can be packed, its radius gives an upper bound. Otherwise, the upper bound can be selected as the summation of the diameters of the circles at the worst case. Similarly, if we know a surrounding circle into which there is no feasible placement of packing all circles, then it is a lower bound for the optimal solution; otherwise, the lower bound can be initialized as the summation of the radii of the largest two circles.

Our algorithm, whose pseudo-code is given in Algorithm 4.1.1, aims to progressively improve the upper and lower bounds for the radius of the surrounding circle. In Algorithm 4.1.1, we start by initializing the radius of the surrounding circle as the half of the summation of the initial upper and lower bounds at Step 1. Then, the given surrounding circle is divided into smaller squares with side length δ at Step 2. Then, we solve a restricted version formulation defined in Chapter 3, which we

Algorithm 4.1.1 DCPACK Algorithm

Require: $\mathcal{C}$; r_c, $\forall c \in \mathcal{C}$; U; L; δ; ε.
Ensure: (x_c, y_c), $\forall c \in \mathcal{C}$; U; L.
1: $R \leftarrow \frac{U+L}{2}$
2: Divide the container with diameter R into square cells with length size δ.
3: Solve the restriction model.
4: **if** the restriction model is feasible **then**
5: $U \leftarrow R$. Go to Step 13.
6: **end if**
7: Solve the relaxation model for the container with diameter R.
8: **if** the relaxation model is feasible **then**
9: $\delta \leftarrow \frac{\delta}{2}$. Go to Step 2.
10: **else**
11: $L \leftarrow R$. Go to Step 13.
12: **end if**
13: **if** $1 - \frac{L}{U} > \varepsilon$ **then**
14: Go to Step 1.
15: **else**
16: STOP!
17: **end if**

called as a "restriction model", to decrease the upper bound. Formulation (3.3) and formulation (3.6) can be solved as a "restriction model" within this algorithm. If the restriction model gives a feasible configuration, the corresponding radius value gives an upper bound for the minimum value of the radius of the surrounding circle. Then, we update the upper bound at Step 5, and check the gap between the bound at Step 13. If the bounds are close enough, i.e., $1 - L/U < \varepsilon$, Algorithm 4.1.1 terminates with a solution which is good enough. However, if the relative gap between the bounds are larger than ε, Algorithm 4.1.1 goes back to Step 1 with the updated upper and lower bounds.

If the restricted model is infeasible, we solve a relaxed version formulation defined in Chapter 3, which we called as a "relaxation model". Formulations (3.5) and (3.8) are designed for a relaxed version of the original problem, and used as a "relaxation model". Since the infeasibility of the relaxed version ensures the infeasibility of the original problem, the infeasibility of the "relaxation model" for a given R value indicates that the R value is a lower bound for the optimal value of the radius of the surrounding circle. Hence, whenever both the restriction model and the relaxation model are infeasible for a given radius, we update the lower bound for the original problem at Step 11. Then, Algorithm 4.1.1 checks the relative gap between the updated lower and upper bounds small enough. If it is small enough, the algorithm terminates. Otherwise, this algorithm update the radius of the surrounding circle at Step 1 for restarting the procedure with the new radius.

If a given radius value for the surrounding circle is feasible in the relaxation model which is infeasible in the restriction model, we cannot say anything for the radius value within Algorithm 4.1.1. In that case, we need to decrease the side length of the cells to get a finer discretization. Then, the algorithm goes back to Step 2 to solve the restriction model by using the new discretization of the surrounding circle by updating the length size at Step 9. The restriction model can have a feasible placement of all circles with the new discretization since there will be more candidate points to locate the centers of circles.

Within Algorithm 4.1.1, we use the different formulations introduced in Chapter 3 to examine the effect of using different-sized formulations by a computational study in Chapter 5.

4.2 Algorithmic Enhancements

In this section, we will give the details of some improvements proposed for Algorithm 4.1.1. In Section 4.2.1, we describe the methods for obtaining initial lower bounds for the radius of the surrounding circle. In Section 4.2.2, we will give the details of Algorithm 4.2.2 which decreases the number of decision variables by decreasing the solution space for each circle. Some other improvements are given in Section 4.2.3.

4.2.1 Initializing Lower Bounds

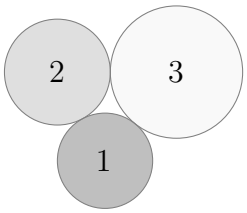
There are different methods for initializing the lower bound in Algorithm 4.1.1. Since we are given the radii of circles to be packed, we can find lower bounds by using basic geometrical knowledge. First of all, the sum of the radii of the largest two circles is a lower bound (LB1) for the radius of the surrounding circle because it is a requirement that the radius of the surrounding circle should be greater than or equal to the sum for packing these two circles into it. Moreover, any surrounding circle should include all of these circles. Hence, the sum of their areas is a lower bound for the area of the surrounding circle, and another lower bound (LB2) can be determined by using this area. Also, Algorithm 4.2.2 can be performed iteratively to calculate another lower bound (LB3) by using a subset of these circles. Algorithm 4.2.2 is

given at Section 4.2.2, and the method used for obtaining LB3 will be described in that section.

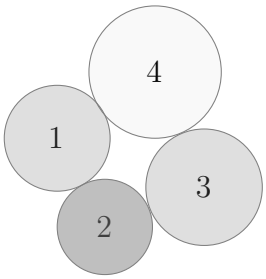
For any configuration of the circles, there will be idle regions where none of the circles are packed. The idle regions can be categorized in two groups: the area between three adjacent circles and the area between the surrounding circle and the circles adjacent to the surrounding circle. A lower bound for the area of the idle regions in any configuration of the circles can be calculated where the number of circles in the instance is greater than three, i.e., $n > 3$. Then, we can improve LB2 by adding the lower bound for the area of the idle regions to the total area of the circles. We name the idle region between adjacent circles as "the first idle region". For any configuration including the optimal one, there will be idle regions between the adjacent circles as shown in Figure 4.1.

Figure 4.1 The idle region between adjacent circles

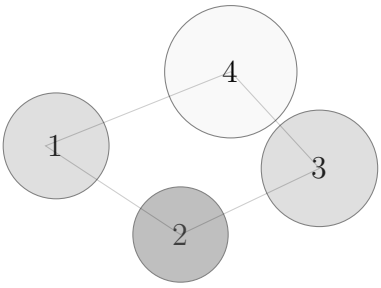
(a) The idle region between three adjacent circles



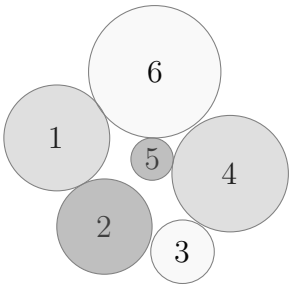
(b) The idle region between n adjacent circles where $n > 3$ and $e_{c,l} = r_c + r_l$



(c) The idle region between four adjacent circles where $e_{c,l} > r_c + r_l$



(d) Different case for adjacent circles



Before investigating these idle regions, we will give some definitions. First of all, we can consider a configuration of circles as a graph such that the center of each circle denotes a node, and there is an edge between the centers of any two circles if their centers can be connected by a line without intersecting with another circle. Assume that there are k adjacent circles such that circle c is adjacent to circles $(c-1)$ and $(c+1)$ for any $2 \leq c \leq (k-1)$ as well as the circles 1 and k are adjacent to each other.

The coordinates of the center of the circle c are given with (x_c, y_c). Let $e_{c,l}$ denote the edge length connecting the centers of the circles c and l, where $e_{c,l} \geq r_c + r_l$ by the non-overlapping rule. For example, for the circles given in Figure 4.1a, there are three nodes denoted by 1, 2, and 3 with the edges (1,2), (2,3), and (3,1) where $e_{1,2} = r_1 + r_2$, $e_{2,3} = r_2 + r_3$, and $e_{1,3} = r_1 + r_3$. Other example configurations of four circles are given in Figure 4.1b with edge lengths equal to the sum of the radii of the corresponding circles, and Figure 4.1c with edge lengths larger than the sum of the radii of the corresponding circles.

Let $Idle^*_{1,\cdots,k}$ denote the area of the idle region between the adjacent circles $1, \cdots, k$. In this notation, each circle can touch their adjacent circles or not, i.e., $e_{l-1,l} \geq r_{l-1} + r_l$ and $e_{l,l+1} \geq r_{l+1} + r_l$ for $l \in \{2, \cdots, k-1\}$ and $e_{1,k} \geq r_1 + r_k$, $e_{1,2} \geq r_1 + r_2$ and $e_{k-1,k} \geq r_{k-1} + r_k$. For instance, the areas of the idle regions between the circles $1, 2, 3$, and 4 are represented by $Idle^*_{1,\cdots,4}$ for both configurations given in Figures 4.1b and 4.1c. If there are sub-cycles within a cycle like given in Figure 4.2c, the total area of the idle region is given with the sum of the areas of the idle regions within the sub-cycles. For instance, there are three sub-cycles between circles $1, 2, 3, 4, 5$, and 6 in Figure 4.1d, i.e., the sub-cycles are $1-2-5-6$; $2-3-4-5$; and $4-5-6$ for the cycle of six circles, and corresponding idle region is given as the summation, $Idle^*_{1,2,5,6} + Idle^*_{2,3,4,5} + Idle^*_{4,5,6}$.

As a special configuration of the adjacent circles $1, \cdots, k$, assume that the edge length connecting the centers of adjacent circles c and l is equal to the sum of their radii, i.e., $e_{c,l} = r_c + r_l$. The idle region is denoted by $Idle_{1,\cdots,k}$ in this configuration. For instance, the area of the idle region is represented by $Idle_{1,\cdots,4}$ in Figure 4.1b since the edge lengths are $e_{1,2} = r_1 + r_2$, $e_{1,4} = r_1 + r_4$, $e_{2,3} = r_2 + r_3$, and $e_{3,4} = r_3 + r_4$.

The area of the idle region $Idle_{1,\cdots,k}$ is calculated by using the following equalities for different k values:

$$Idle_{1,2,3}\text{: } \sqrt{(r_1+r_2+r_3)(r_1)(r_2)(r_3)} - \sum_{\substack{(c,k,l)\in\{(1,2,3),\\(2,1,3),(3,1,2)\}}} \frac{(r_c)^2 cos^{-1}\left(\frac{r_c(r_c+r_k+r_l)-r_k r_l}{(r_c+r_l)(r_c+r_k)}\right)}{2},$$

$$Idle_{1,2,3,4}\text{: } Idle_{1,2,3} + Idle_{1,3,4},$$

$$Idle_{1,\cdots,k}\text{: } Idle_{1,2,3} + Idle_{1,3,4} + Idle_{1,4,5} + \cdots + Idle_{1,k-1,k}.$$

For calculating the idle region $Idle_{1,\cdots,k}$, we divide k circles into groups of three circles. Then, we consider each group of three circles as adjacent circles as a common case, and then we calculate the area of the idle regions between each group of three circles.

Now, we will prove that we can calculate a lower bound for the total area of the idle

regions located between larger number of adjacent circles as the sum of the area of the idle regions between the subgroups of three circles as proven in Theorem 4.1.

In addition, Theorem 4.1 ensures that the idle region between any k adjacent circles without any sub-cycle (where $e_{c,l} \geq r_c + r_l$) is larger than the idle region between them when $e_{c,l} = r_c + r_l$ for all $c, l \in \mathcal{C}$.

Theorem 4.1. *The area of the idle regions between k circles if $e_{c,l} = r_c + r_l$ for any adjacent circles c and k is less than the area of the idle regions for any other configuration of these k circles, i.e., $Idle_{1,\cdots,k} \leq Idle^{*}_{1,2,\cdots,k}$*

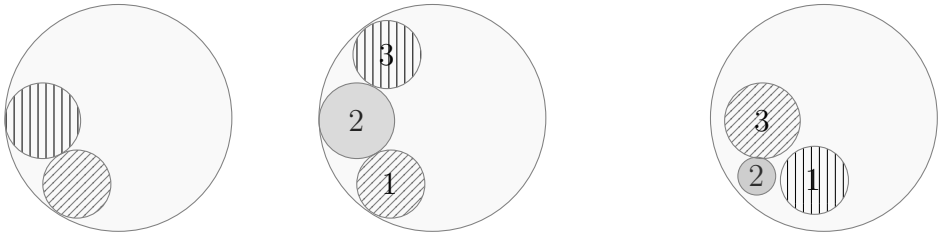
The proof of Theorem 4.1 is given at Appendix A.

Figure 4.2 The idle region between the surrounding circle and the packed circles for different configurations.

(a) Idle region between the surrounding circle and two adjacent circles.

(b) Idle region between the surrounding circle and three succeeding circles.

(c) Idle region between surrounding circle and three succeeding circles for a different case.



Based on a similar idea, we can also detect the idle regions between the surrounding circle and the circles adjacent to the surrounding circle like given in Figure 4.2a. The idle region given in Figure 4.2a depends on the radii of two adjacent circles and the surrounding circle. This idle region is denoted by $Idle^{R}_{0,c,k}$ where index 0 denotes the surrounding circle with radius R.

This idle region $Idle^{R}_{0,c,k}$ exists if there is an edge between the circles c and k, and the ray starting from the center of the surrounding circle and containing the center of circle c (k) do not intersect with another circle after the center of circle c (k). During this procedure if the configuration of adjacent circles is like in Figure 4.2b, then, the circles 1 and 3 are not adjacent. Then, the total idle region is given with the sum of two idle regions: $Idle^{R}_{0,1,2}$ and $Idle^{R}_{0,2,3}$. However, if it is like in Figure 4.2c, then, the circles 1 and 3 are adjacent also, but the total idle area in that case will be the sum of three areas: $Idle_{1,2,3}$, $Idle^{R}_{0,1,2}$ and $Idle^{R}_{0,2,3}$.

The area of the idle region depends on the radius of the surrounding circle, and will be the minimum value in the optimal solution of problem (3.1). However, the

minimum radius of the surrounding circle is unknown. Hence, the radius of the surrounding circle should be selected appropriately such that the resulting idle region will be still the minimum possible idle region. The best known lower bound for the radius of the surrounding circle gives the minimum idle region under certain statements (the best lower bound is equal to the maximum of LB1, LB2 and LB3). In this notation, the edge length $e_{0,c}$ corresponds to the distance between the circle c and the surrounding circle which is equal to $r_0 - r_c - \|(x_c, y_c)\|_2$ where the center of the surrounding circle is located at origin. These statements are given by the following theorem:

Theorem 4.2. *$Idle^{R}_{0,c,k} < Idle^{R^*}_{0,c,k}$ where R^* is the optimal value of the radius of the surrounding circle radius if the following statements are valid:*

- *If $R_1 \leq R_2$, then, $Idle^{R_1}_{0,c,k} \geq Idle^{R_2}_{0,c,k}$ between $L \leq R \leq U$,*
- *$\frac{\partial^2}{\partial R^2}(Idle^{R}_{0,c,k})$ has no roots for $L \leq R \leq U$ where L and U are the given lower and upper bounds for the radius of the surrounding circle.*

The proof of this theorem is given in Appendix A. If $Idle^{R}_{0,c,k} < Idle^{R^*}_{0,c,k}$; then, we can use the initial upper bound to calculate the idle regions. Moreover, we can recalculate a lower bound by using the idle regions between the adjacent circles and the total area of the given circles.

To calculate a lower bound for the idle region in the optimal packing, we propose an integer programming formulation. In this formulation, we will define a new set which is denoted by $\overline{\mathcal{C}} = \mathcal{C} \cup \{0\}$ where 0 is the index of the surrounding circle. We will also define two binary variables: $d_{c,k,l}$ and $f_{k,l,m}$ where $c, k, l, m \in \overline{\mathcal{C}} : c < k < l$. If the circles c, k, and l are adjacent to each other, the corresponding binary variable $d_{c,k,l}$ will be 1. Also, we define a parameter $\Delta_{c,k,l}$ corresponding to the idle region between circles c, k, and l. If circles c, k and l are adjacent to the surrounding circle as shown in Figure 4.2c (i.e., $d_{c,k,l} = 1$, $d_{0,c,k} = 1$, $d_{0,k,l} = 1$, $d_{0,c,l} = 1$), the binary variable $f_{c,k,l}$ will be 1 to handle this case. Assume that circle k is the circle between the circles c, l and the surrounding circle. Then, the idle region between the circles c and l includes the circle k, the idle region between the circles c and k, and the idle region between the circles k and l. Hence, these idle regions are added to the objective function two times; and they should be eliminated. The parameter $\rho_{c,k,l}$ is defined for this case, which is the amount of decrease in the idle region of the circles 0, c, k, and l; i.e., $\rho_{c,k,l} = \Delta_{0,c,k} + \Delta_{0,k,l} + \Delta_{c,k,l} + \pi(r_k)^2$.

In this section, we also define two other parameters: min_c and max_c which are the maximum and minimum number of circles whereas circle c can have at neighbours, respectively. The calculation of the parameters min_c and max_c will be detailed later.

Then, the formulation is as follows:

$$\text{(4.1a)} \quad \min \sum_{c,k,l \in \overline{\mathcal{C}}: c<k<l} \left(\Delta_{c,k,l} d_{c,k,l} - \rho_{c,k,l} f_{c,k,l} \right)$$

$$\text{(4.1b)} \quad \text{s.t.} \sum_{k,l \in \overline{\mathcal{C}}} d_{c,k,l} + \sum_{k,l \in \overline{\mathcal{C}}} d_{k,c,l} + \sum_{k,l \in \overline{\mathcal{C}}} d_{k,l,c} \geq min_c \qquad c \in \overline{\mathcal{C}}$$

$$\text{(4.1c)} \quad \sum_{k,l \in \overline{\mathcal{C}}} d_{c,k,l} + \sum_{k,l \in \overline{\mathcal{C}}} d_{k,c,l} + \sum_{k,l \in \overline{\mathcal{C}}} d_{k,l,c} \leq max_c \qquad c \in \overline{\mathcal{C}}$$

$$\text{(4.1d)} \quad f_{c,k,l} \leq d_{c,k,l} \qquad c,k,l \in \overline{\mathcal{C}}$$

$$\text{(4.1e)} \quad f_{c,k,l} \leq d_{0,c,k}, \quad f_{c,k,l} \leq d_{0,k,l}, \quad f_{c,k,l} \leq d_{0,c,l} \qquad c,k,l \in \overline{\mathcal{C}}$$

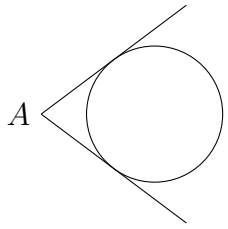
$$\text{(4.1f)} \quad f_{c,k,l} \geq d_{c,k,l} + d_{0,c,k} + d_{0,k,l} + d_{0,c,l} - 3 \qquad c,k,l \in \overline{\mathcal{C}}$$

$$\text{(4.1g)} \quad d_{c,k,l}, f_{c,k,l} \in \{0,1\} \qquad c,k,l \in \overline{\mathcal{C}}.$$

Constraints (5.1c)-(5.1d) ensures that circle c have at least min_c and at most max_c circles as its neighbours. By these constraints, we guarantee that each circle will create an idle region. Constraints (5.1e)-(5.1g) assign the value of the decision variable $f_{c,k,l}$ according to the values of the decision variables $d_{c,k,l}$, $d_{0,c,k}$, $d_{0,k,l}$, and $d_{0,c,l}$. If all the decision variables $d_{c,k,l}$, $d_{0,c,k}$, $d_{0,k,l}$, and $d_{0,c,l}$ are assigned to one, the decision variable $f_{c,k,l}$ will be one; 0, otherwise. If one of the decision variables $d_{c,k,l}$, $d_{0,c,k}$, $d_{0,k,l}$, and $d_{0,c,l}$ are zero, the Constraints (5.1e)-(5.1f) ensures that $f_{c,k,l} = 0$. If all of them are one, Constraints (5.1e)-(5.1f) will be redundant. Then, Constraint (5.1g) assigns $f_{c,k,l}$ the value of one. Finally, Constraints (5.1h) are the domain constraints for the decision variables.

As we stated before, the number of neighbours of circle c in the optimal solution is between min_c and max_c. To calculate the value of the parameter min_c, the largest circles are considered as the neighbours of circle c to provide a lower bound for the number of circles adjacent to circle c as seen in Theorem 4.3.

Figure 4.3 Two tangents from an external point.



To calculate the value of min_c, we perform Algorithm 4.2.1 by selecting $k_1 = 1$. Selecting $k_1 = 1$ means that the circles are listed according to their radii in decreasing

Algorithm 4.2.1 min_c and max_c Calculation Algorithm

Require: $\mathcal{C}$; r_c, $\forall c \in \mathcal{C}$; k_1.
Ensure: K.
1: $sum \leftarrow 0$ and $k \leftarrow k_1$
2: $K \leftarrow k$.
3: **if** $k_1 = 1$ **then**
4: $k \leftarrow k+1$.
5: **else**
6: $k \leftarrow k-1$.
7: **end if**
8: $sum \leftarrow sum + cos^{-1}\left(\frac{(r_c+r_{k-1})^2+(r_c+r_k)^2-(r_{k-1}+r_k)^2}{2r_k r_{k-1}}\right)$.
9: **if** $sum + cos^{-1}\left(\frac{(r_c+r_1)^2+(r_c+r_k)^2-(r_1+r_k)^2}{2r_k r_1}\right) \leq 2\pi$ **then**
10: Go to Step 2.
11: **else**
12: STOP!
13: **end if**

order. With the help of Cosine Theorem, we will prove that the parameter min_c gives a lower bound for the number of neighbours of circle c in the optimal configuration of problem (3.1).

Theorem 4.3. *The parameter* $\min_c$ *is a lower bound on the number of neighbours of the circle* c *in the optimal configuration of the problem* (3.1) *if* $\min_c$ *is calculated by Algorithm 4.2.1 for* $k_1 = 1$.

Proof of Theorem 4.3. By Cosine Theorem, the total angle occupied by the adjacent k circles to circle c is denoted by sum at each step. By the definition of adjacency, if $sum \geq 2\pi$, the angle where all lines started from the center of the circle c intersect with a neighbour circle l for $l < k$. Since the k circles are the largest k circles, the corresponding angles are the largest (obvious from the cosine function). Then, the number of circles is minimum if the angles are calculated by assigning the largest K circles adjacent such that the summation of the corresponding angles is less than or equal to 2π. □

The parameter max_c value is calculated by using the smallest circles as neighbours to maximize the number of circles neighbour to circle c as shown in Theorem 4.4. Then, k_1 is assigned to n in Algorithm 4.2.1. If $k_1 = n$, this means that the circles are listed in increasing order according to their radii.

Theorem 4.4. *If* max_c *is calculated by Algorithm 4.2.1 for* $k_1 = n$*, then, the number of neighbours of the circle* c *in the optimal configuration of problem* (3.1) *is less than* $\max_c$.

Proof. Contrary to Theorem 4.3, the angle of the slice is the smallest where all lines started from the center of the circle c intersecting with adjacent circle k which are the smallest. Then, the number of circles is maximum if the angles are calculated by assigning the smallest K circles such that the summation of the corresponding angles is less than or equal to 2π. □

Theorem 4.5. *Formulation* (4.1) *gives a lower bound for the idle region in the optimal solution.*

Proof. By Theorems 4.3 and 4.4, Constraints (4.1b)-(4.1c) hold for the optimal configuration. Since the problem is minimization problem and the idle region calculated for circles c, k, l is a lower bound for the corresponding idle region, the objective value of the corresponding formulation gives a lower bound for the idle region in the optimal configuration. □

Then, we can find another lower bound (LB4) by using the summation of the objective function of formulation 4.1 and the total areas of the circles given to be packed.

4.2.2 Solution Space Reductions

So far, the feasible region to place the center of each circle is determined as the whole surrounding circle. However, we can reduce the feasible region for each circle by using analytical properties. While determining the feasible regions for the circles, we start with identifying the reduced feasible regions for the largest two circles.

First, without loss of generality, we assume that the center of the largest circle is placed to the first quadrant, and the center of the second largest circle is placed to the half space defined by $\{(x, y) \in \mathbb{R}^2 : y - x \geq 0\}$ with the help of symmetry to eliminate symmetric solutions. Then, the solution space of problem (3.1) with additional constraints $x_1 \geq 0, y_1 \geq 0, y_2 \geq x_2$ is equal to the first reduced region; the corresponding formulation is given as follows:

$$\min R \tag{4.2a}$$

$$\text{s.t. } (3.1b) - (3.1d) \tag{4.2b}$$

$$x_1, y_1 \geq 0 \tag{4.2c}$$

$$y_2 \geq x_2 \tag{4.2d}$$

Constraints (4.2b)-(4.2d) ensure the feasibility of the CPP. Constraint (4.2c) restricts the region for placing the center of the largest circle to the first quadrant. Constraint (4.2d) limits the region for placing the second largest circle to the half space given with $\{(x,y) \in \mathbb{R}^2 : y - x \geq 0\}$.

Let $\mathcal{D}_{(4.2)}$ be the set of all feasible solutions of the problem (4.2).

Theorem 4.6. *For any feasible solution in $\mathcal{D}_{(3.1)}$, there is a corresponding feasible solution in $\mathcal{D}_{(4.2)}$.*

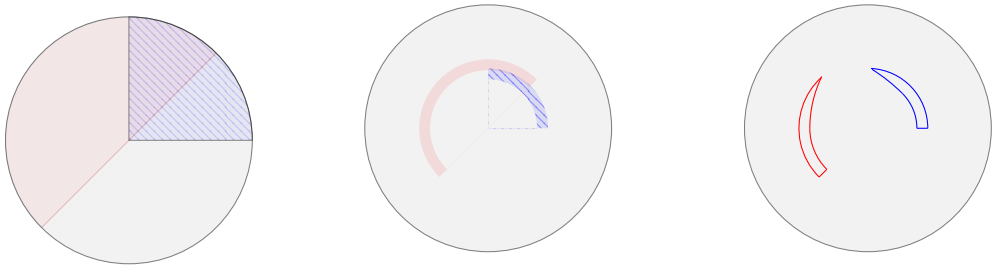
Proof. The reduced region is given in Figure 4.4a. Assume that τ is the counter clock-wise angle between the lines connecting the center of circle 1 and the center of circle 2 to the center of the surrounding circle in a feasible solution in $\mathcal{D}_{(3.1)}$.

If $\tau \leq \pi$, there is a corresponding solution in $\mathcal{D}_{(4.2)}$ found by rotating it $\frac{\pi}{2}$ degrees or multipliers according to the placement of the largest circle.

If $\tau > \pi$, take the reflection of the corresponding feasible solution at first. Then, by rotating it $\frac{\pi}{2}$ degrees or multipliers according to the placement of the largest circle, we can get the corresponding solution in $\mathcal{D}_{(4.2)}$. Hence, there is a corresponding feasible solution in $\mathcal{D}_{(4.2)}$ for each feasible solution in $\mathcal{D}_{(3.1)}$. □

Figure 4.4 Solution space reductions corresponding to locate the largest two circles with radii 6 and 7 units into a circle with radius 13.6 units.

(a) First reduced regions. (b) Improved reduced regions. (c) The resulting reduced regions.



In Figure 4.4a, the blue-dashed and red undashed regions show the reduced regions to place the largest and the second largest circles, respectively. The regions in Figure 4.4a are given with $\{(x_1,y_1),(x_2,y_2) \in \mathbb{R}^2 : (x_c)^2 + (y_c)^2 \leq R^2, x_1 \geq 0, y_1 \geq 0, y_2 \geq x_2\}$. Then, the regions for placing the centers of the largest and second largest circles can be reduced by using the feasibility rules of the problem (4.2), respectively, as follows:

$$\{(x_1,y_1) \in \mathbb{R}^2 : (\max\{2r_2 + r_1 - R, 0\})^2 \leq (x_1)^2 + (y_1)^2 \leq (R - r_1)^2, x_1 \geq 0, y_1 \geq 0\},$$

$$\{(x_2,y_2) \in \mathbb{R}^2 : (\max\{2r_1 + r_2 - R, 0\})^2 \leq (x_2)^2 + (y_2)^2 \leq (R - r_2)^2, y_2 \geq x_2\}.$$

The above regions are obtained by the feasibility rules of the problem (3.1) since all circles should be packed without overlapping which are given in Figure 4.4b.

The reduced regions given in Figure 4.4b will be combined with the knowledge that any two circles cannot overlap. Then, the corresponding regions can be decreased by eliminating the points where there is no feasible point to locate the center of the other circle. After this elimination step, we get the feasible regions given in Figure 4.4c for the largest two circles. These resulting regions are the minimum possible feasible regions for these circles within this procedure. Then, by using the reduced regions for the largest two circles, we can reduce the feasible regions for locating the centers of other circles by performing an iterative algorithm. Let the set of the corner points of the reduced regions are called as $\mathcal{S}_1$, $\mathcal{S}_2$ for the largest two circles, and $\mathcal{S}_c$ be the corner points of the resulting feasible regions obtained by Algorithm 4.2.2 for $c < k$. The steps of Algorithm 4.2.2 are given below for the k^{th} circle at the descending order of circles according to their radii:

Algorithm 4.2.2 Identifying Feasible Regions Algorithm (IFRA)

Require: $\mathcal{C}$; r_c, $\forall c \in \mathcal{C}$; $\mathcal{S}_c$, $\forall c \in \mathcal{C} : c < k$; R, k.
Ensure: True or False.

1: **for** $\{p_1, \cdots, p_k : \|(p_c, p_l)\|_2 \geq (r_c + r_l), p_c \in \mathcal{S}_c\}$ where $c < k$ **do**
2: $\quad \mathcal{G} = \{p_k = (x_k, y_k) \in \mathbb{R}^2 : (R - r_k) \geq \|p_k\|_2\} \geq \max\{0, 2r_1 + r_k - R\}\}$
3: $\quad \mathcal{H} = \left\{p_k = (x_k, y_k) \in \mathbb{R}^2 : c < k, \|p_k, p_c\|_2 \geq r_c + r_k\right\} \cap \mathcal{G}$
4: **end for**
5: **if** $\mathcal{H} = \emptyset$ **then**
6: $\quad$ return FALSE.
7: **else**
8: $\quad$ return TRUE.
9: **end if**

In Algorithm 4.2.2, we discussed the sets $\mathcal{G}$ and $\mathcal{H}$ which are nonconvex. For this sets, we did not store the sets as a whole. Instead of this, we find the intersection points of the circles with radii $(r_c + r_k)$ with the corresponding boundary functions for each p_c, $c < k$. For each intersection point, we check the non-overlapping feasibility rule for other circles. If there is a feasible intersection point, then, there is a feasible solution for placing k circles into the circle with radius R. Theorem 4.7 proves the statement.

Theorem 4.7. *If Algorithm 4.2.2 results in False for any k, there is no feasible configuration of circles in the given surrounding circle; otherwise, there is at least one feasible placement.*

Proof. If Algorithm 4.2.2 results in True for a circle at order k, then, by locating its center to a point in $\mathcal{H}$ and other circles to the corresponding points p_c for $c < k$; this

configuration is feasible for this problem.

If Algorithm 4.2.2 results in False, we will show that there is no feasible placement for the first k circles. Assume that there is a feasible configuration for the first k circles, but $\mathcal{H} = \emptyset$. From the feasibility of the configuration, there are points p_c such that $\|(p_c, p_l)\|_2 \geq (r_c + r_l)$ and $\|p_c\|_2 \leq R - r_c$ hold. Also $\mathcal{G} \neq \emptyset$, and there is a corresponding point $p_k \in \mathcal{G}$ which also satisfies $\|(p_k, p_c)\|_2 \geq r_c + r_k$ from feasibility. Hence, $p_k \in \mathcal{H} \neq \emptyset$ which contradicts with our assumption. So, there cannot be any feasible placement if Algorithm 4.2.2 results in False which completes proof. □

By using the resulting reduced regions obtained by Algorithm 4.2.2, we can eliminate the decision variables corresponding to the candidate points which are not included by the resulting regions for each circle.

4.2.3 Other Improvements

In addition to the previous section, we introduce a set of procedures handled during the branching process. First of all, the decision variables are prioritized according to the radius of the corresponding circle. The variables corresponding larger circles have the larger priorities.

During the branching process, if there is not enough place to locate the remaining circles, we use branch and callback to add some cuts which are added on two different cases: (i) not enough area to place all the remaining circles, (ii) no candidate points to place all the remaining circles. The idle regions where none of the remaining circles can be placed is eliminated at first. Then, the remaining idle area is calculated to compare with the sum of the areas of the remaining circles with the minimum idle area between them to be packed. If the remaining area is less than the sum of the areas of the remaining circles, then, the branch is eliminated. To check the remaining candidate points, if the farthest candidate points within the sets is less than the sum of the radii, then, the branch is eliminated. Also, according to the values of the assigned decision variables, the decision variables corresponding to the points where assigning these variables as one will result in overlapping of circles is assigned as zero with the help of conditional constraints.

Finally, to obtain better upper bound values, we use the best-known upper bounds from Circle Packing Contest of Al Zimmermann's Programming Contests and Packomania website with the algorithm given in Huang et al. (2006).

5. COMPUTATIONAL STUDY

In this section, we evaluate our algorithm on a series of benchmark instances up to 162 circles from a the Packomania website (Packomania, 2020), the Circle Packing Contest of Al Zimmermann's Programming Contests website (Zimmermann's, 2005), and instances including equal circles from a paper (Huang & Ye, 2011). We denote the instances from the Circle Packing Contest of Al Zimmermann's Programming with "$Zimm-n$" which represents the instance with n-circles where $5 \leq n \leq 30$ (Zimmermann's, 2005). Also, we will use the same labels for the benchmark instances given in the Packomania website (Packomania, 2020). Finally, we will denote the instances including equal circles introduced by Huang & Ye (2011) with "$Eq-n$" containing n-circles. Our computing platform is a Windows 10 Laptop with Intel Core i7-8565U processor and 32 GB of RAM. CPU times are given in minutes, and all algorithms are coded in Python.

Before analyzing the performance of our algorithm, we will solve problem (3.1) with the global solvers Baron and Gurobi 9.1 to show the effectiveness of the global solvers in Section 5.1. Then, in Section 5.2, we will compare the performance of our algorithm with the different-sized formulations introduced in Chapter 3. Then, we will analyze the amount of time used for pre-processing within Algorithm 4.1.1. In Section 5.3, we will compare our algorithm for the equal circle case with a solution method proposed problem in Huang & Ye (2011). Then, the comparison of our solution procedure with global optimization method defined in literature for CPP within different surrounding containers by Stoyan & Yas'kov (2004) in Section 5.4.

5.1 Results Obtained by Baron and Gurobi

In this section, we just use the first six instances given problem in Zimmermann's (2005) to analyze the performance of problem (3.1) and its improved version. For the

basic version, we did not share any knowledge with the global solvers in terms of lower and upper bounds or solution space reductions. Then, we will solve problem (3.1) initialized with all knowledge which we had as the improved version. We coded these formulations in Python, and solve it by solvers Gurobi 9.1 and Baron with time limit 30 minutes. The instances terminated by time limit is represented by "TLE". The optimality gap is set to 1%. The results obtained by each solver are given in Table 5.1 as follows:

Table 5.1 Results obtained by the global solvers

	Baron						**Gurobi 9.1**					
	Basic Ver.			Improved Ver.			Basic Ver.			Improved Ver.		
Ins.	Upper Bound	Solution Time (m)	GAP (%)	Upper Bound	Solution Time (m)	GAP (%)	Upper Bound	Solution Time (m)	GAP (%)	Upper Bound	Solution Time (m)	GAP (%)
$Zimm-5$	9.00	1.26	0.01	9.00	0.01	0.00	9.00	0.50	0.99	9.03	0.72	0.36
$Zimm-6$	11.06	12.83	0.01	11.11	0.27	0.46	11.06	17.31	0.52	11.06	1.28	0.04
$Zimm-7$	13.46	TLE	3.43	13.46	10.75	2.28	13.46	TLE	3.44	13.46	TLE	2.28
$Zimm-8$	16.22	TLE	7.53	16.22	TLE	5.85	16.38	TLE	8.45	16.44	TLE	7.08
$Zimm-9$	19.28	TLE	11.83	19.28	TLE	9.73	19.84	TLE	14.31	19.75	TLE	11.90
$Zimm-10$	22.00	TLE	13.64	22.65	TLE	13.73	23.22	TLE	18.16	23.38	TLE	16.40

As one can see easily, the global solvers cannot solve problem (3.1) even for 8-circle instance for both basic and improved versions. In the improved version, our given lower bounds are not improved during the solution times by the solvers, but the global solvers can detect and improve upper bounds for problem (3.1). Hence, this problem is quite challenging to solve with the help of the global solvers even for the small-sized instances. For the problems that were terminated by time limit, the GAP values are calculated according to the final results obtained by the solvers to compare with Algorithm 4.1.1.

5.2 Performance Analysis of Algorithm 4.1.1

As stated before, Algorithm 4.1.1 solves formulations proposed for a restricted and a relaxed versions iteratively. We introduced two different-sized formulations for these versions in Chapter 3, and we first give the results obtained by two versions of Algorithm 4.1.1 improved by the proposed algorithmic enhancements given in Section 4.2.

As a reminder, formulations (3.3) and (3.5) are proposed for the restricted and relaxed versions with linear-number of decision variables, respectively; and the

formulations (3.6) and (3.8) contain logarithmic number of decision variables designed for the restricted and relaxed versions, respectively. In this section, the time limit is 30 minutes. In Table 5.2, "With Linear-Sized Formulation" reports the results of Algorithm 4.1.1 which solves the formulation (3.3) as the restricted version and the formulation (3.5) as the relaxed version where "With Logarithmic-Sized Formulation" solves the formulations (3.6) and (3.8), respectively. The linear-sized formulations are the ones which are not naïve. For this comparison, we used Algorithm 4.1.1 with the enhancements proposed in Section 4.2.

Table 5.2 Comparison of using different-sized formulations within Algorithm 4.1.1

	With Linear-Sized Formulation			With Logarithmic-Sized Formulation		
Instance	Upper Bound	Solution Time (m)	GAP (%)	Upper Bound	Solution Time (m)	GAP (%)
$Zimm-5$	9.0084	0.06	0.08	9.0014	0.06	0.76
$Zimm-6$	11.0749	0.21	0.16	11.0710	0.24	0.13
$Zimm-7$	13.4623	2.41	0.50	13.4671	1.07	0.64
$Zimm-8$	16.2217	5.42	0.43	16.2243	2.57	0.81
$Zimm-9$	22.0046	11.39	0.75	22.0023	5.12	0.39
$Zimm-10$	25.6901	TLE	3.53	24.9605	8.41	0.91

As seen in Table 5.2, using formulation (3.6) as the restricted version and the formulation (3.8) as the relaxed version in Algorithm 4.1.1 is better than using other formulations in terms of solution time. Hence, since working with the formulations including a logarithmic number of decision variables in terms of the number of cells improves the performance of Algorithm 4.1.1, we will focus on this version in the following experiments.

According to the comparison of Tables 5.1 and 5.2, Algorithm 4.1.1 with logarithmic-sized formulations also dominate the global solvers in terms of the solution time and solution quality. Even Algorithm 4.1.1 with linear-sized formulations dominates the global solvers in terms of the solution time and solution quality, although, it performs worse than Algorithm 4.1.1 with logarithmic-sized formulations.

After comparing the effects of using different formulations within Algorithm 4.1.1, we investigate the effects of the mechanisms defined in Section 4.2. In Table 5.3, we will give the results obtained by Algorithm 4.1.1 for two different versions: the basic version where no algorithmic enhancements are performed and the enhanced version utilizing the enhancements proposed in Section 4.2.

Table 5.3 Performance analysis of the enhancements introduced for Algorithm 4.1.1 with logarithmic-sized formulations

	Basic Version			**Improved Version**			
Instance	Upper Bound	Total Sol. Time (m)	GAP (%)	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)
$Zimm-5$	9.002	2.07	0.017	9.001	0.04	0.06	0.764
$Zimm-6$	11.139	9.25	0.727	11.071	0.17	0.24	0.126
$Zimm-7$	13.471	17.72	0.819	13.467	0.86	1.07	0.641
$Zimm-8$	16.321	28.20	0.991	16.224	1.59	2.57	0.810
$Zimm-9$	22.264	39.08	0.923	22.002	2.01	5.12	0.390
$Zimm-10$	24.961	58.25	0.947	24.961	2.77	8.41	0.907

For advancing Algorithm 4.1.1's performance, we introduced some algorithmic enhancements in Section 4.2. With the help of these enhancements, Algorithm 4.1.1's effectiveness increases as seen in Table 5.3 in terms of solution time.

As seen in Table 5.3, the enhancements improve Algorithm 4.1.1's performance, and they do not increase the solution time. In fact, solving Algorithm 4.1.1 without the enhancements is more expensive than performing enhancement methods together with Algorithm 4.1.1. We also try to understand the effect of introduced enhancement methods one by one. We proposed four versions of Algorithm 4.1.1 in Tables 5.4 and 5.5: Version 1 is the one without any lower bound improvement (Section 4.2.1), Version 2 is the one without any upper bound improvement (Section 4.2.3), Version 3 is the one without any solution space reductions (Section 4.2.2), and Version 4 is the one without any cuts added during the branching process (Section 4.2.3).

Table 5.4 Effects of the qualities of starting lower and upper bounds for Algorithm 4.1.1

	Version 1				**Version 2**			
Instance	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)
$Zimm-5$	9.002	0.01	1.43	0.022	9.006	0.03	2.12	0.066
$Zimm-6$	11.140	0.04	6.63	0.863	11.084	0.14	5.73	0.432
$Zimm-7$	13.469	0.15	13.56	0.891	13.470	0.68	11.41	0.796
$Zimm-8$	16.273	0.41	21.32	0.884	16.269	1.11	18.83	0.827
$Zimm-9$	22.006	0.68	29.53	0.869	22.192	1.33	26.59	0.961
$Zimm-10$	24.960	0.91	47.21	0.726	24.961	1.88	43.52	0.934

Initializing lower and upper bounds is an important factor in Algorithm 4.1.1 according to Table 5.4. The pre-processing time required for initializing the bounds is less than the necessary time spent in Algorithm 4.1.1 without the pre-processing methods.

Hence, the methods for initializing the upper and lower bounds are effective since the solution times do not decrease much without initializing them with pretty good bounds.

Table 5.5 Effects of other enhancements one by one introduced for Algorithm 4.1.1

	Version 3				**Version 4**			
Instance	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)
$Zimm-5$	9.001	0.03	0.06	0.862	9.001	0.04	0.07	0.546
$Zimm-6$	11.091	0.15	0.26	0.356	11.087	0.17	0.28	0.251
$Zimm-7$	13.466	0.81	1.04	0.697	13.468	0.84	1.13	0.592
$Zimm-8$	16.224	1.53	6.48	0.691	16.224	1.56	2.91	0.862
$Zimm-9$	22.003	1.92	17.21	0.409	22.002	2.04	6.74	0.396
$Zimm-10$	24.960	2.61	25.03	0.917	24.960	2.79	11.43	0.952

According to Table 5.5, we can say that the decrease in total solution time obtained by using solution space reductions is less than total time required for obtaining these solution space reductions since the solution time of each formulation at each iteration decreases by the elimination of the corresponding decision variables.

Other than observing the effect of the sub-methods designed for enhancing Algorithm 4.1.1's performance one by one, we want to determine the efficiency of the algorithm we used for initializing the upper bound if we do not know the best-known value from the literature. As a result, we will solve the instances by only using the upper bound initialization algorithm given problem in Huang et al. (2006), and we compare the results with the results by initializing the upper bounds as equal to the best-known values in Table 5.6. Similarly, the optimality gap is set to 1%, and Algorithm 4.1.1 terminates after 120 minutes of processing. The instances that terminate due to time limit is denoted by "TLE".

Table 5.6 Effects of using best-known values within Algorithm 4.1.1

	Upper Bound Initialization Algorithm				**Initializing with Best-known Values**			
Instance	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)	Upper Bound	Pre-processing Time (m)	Total Sol. Time (m)	GAP (%)
$Zimm-11$	28.371	3.72	14.43	0.591	28.371	2.71	12.92	0.573
$Zimm-12$	31.546	4.90	20.43	0.988	31.545	3.91	17.26	0.959
$Zimm-13$	35.096	6.04	37.05	0.426	35.096	5.41	36.43	0.373
$Zimm-14$	38.839	7.88	54.26	0.906	38.838	6.40	48.13	0.903
$Zimm-15$	42.457	9.94	63.56	0.851	42.457	8.51	61.30	0.838
$Zimm-16$	46.291	12.93	73.59	0.683	46.291	10.71	68.73	0.681
$Zimm-17$	50.129	15.07	99.26	0.833	50.120	12.87	90.36	0.457
$Zimm-18$	54.240	19.98	111.51	0.934	54.240	17.19	108.09	0.931
$Zimm-19$	58.401	24.91	TLE	1.974	58.401	21.63	TLE	1.972
$Zimm-20$	62.791	29.03	TLE	4.289	62.559	25.17	TLE	3.568

As seen in Table 5.6, Algorithm 4.1.1 performs good enough without knowing the best-known values by using the upper bound initialization algorithm stated in Section 4.2.3. Although the upper bound initialization algorithm could not found the best-known radii for some instances, the general performance of Algorithm 4.1.1 is not effected conspicuously.

According to Table 5.6, we can also say that Algorithm 4.1.1 can solve problems up to 18 circles within 120 minutes time limit whether or not the best-known is given to us. In addition, the solution quality can be increased by solving the problems containing 19 and 20 circles with a longer time limit. Apart from the size of the problems, since our algorithm is a global optimization method, the solution quality is guaranteed in our solution method, especially for such a problem for which finding a tight lower bound is hard to identify.

5.3 Comparison with an Algorithm Designed for Equal Circle Case

In this section, we will give the results obtained for a subset of the equal circle instances. In the literature, the circles from 1 to 200 circles are solved where each circle is a unit circle. For the packing of n-equal circles into a larger circle problem, an algorithm is proposed by Huang & Ye (2011). In this section, we will compare our results and solution times with the algorithm given in Huang & Ye (2011).

In the proposed algorithm in Huang & Ye (2011), there are two main procedures: the local-search procedure and the basin-hopping procedure. During the algorithm, the non-overlapping constraint is observed by a parameter defined as elastic force like the other feasibility constraint stating that each circle is fully contained by the surrounding circle. The local-search procedure tries to find a better packing by performing small moves from the current positions of the circles, and the basin-hopping procedure may result in larger moves to explore different regions in the solution space. They combined these methods to construct the algorithm. The comparison Algorithm 4.1.1 with the algorithm introduced in Huang & Ye (2011) is given in Table 5.7. The upper bounds are obtained by the upper bound initialization algorithm given problem in Huang et al. (2006). The solution times given for the algorithm in Huang et al. (2006) are the total solution times where the initial solutions are constructed.

Table 5.7 Results obtained by Algorithm 4.1.1 and algorithm problem in Huang & Ye (2011)

	Algorithm 4.1.1			**Algorithm problem in Huang & Ye (2011)**	
No.	Upper Bound	Sol. Time (m)	GAP (%)	Upper Bound	Sol. Time (m)
$Eq-20$	5.122	4.17	0.64	5.122	4.23
$Eq-21$	5.251	4.48	0.49	5.252	3.51
$Eq-22$	5.439	4.85	0.93	5.439	5.17
$Eq-23$	5.545	5.46	0.84	5.545	5.46
$Eq-24$	5.650	5.81	0.70	5.651	3.23
$Eq-25$	5.753	6.34	0.91	5.753	4.14
$Eq-26$	5.828	7.01	0.53	5.828	4.80
$Eq-27$	5.906	7.63	0.47	5.906	5.61
$Eq-28$	6.015	8.28	0.71	6.015	6.17
$Eq-29$	6.138	9.11	0.96	6.138	6.84
$Eq-30$	6.198	9.43	0.62	6.198	5.72
$Eq-31$	6.291	9.21	0.99	6.291	7.09
$Eq-32$	6.429	9.91	0.87	6.429	7.17
$Eq-33$	6.486	10.47	0.91	6.486	7.43
$Eq-34$	6.611	11.22	0.74	6.611	8.91
$Eq-35$	6.696	12.08	0.97	6.697	7.76
$Eq-40$	7.123	14.42	0.89	7.124	10.59

Algorithm 4.1.1 presents an upper and a lower bound for the radius of the surrounding circle. Hence, we can ensure that the obtained circle is at most 1% away from the optimal-sized circle in which all circles are packed. However, the algorithm introduced in Huang & Ye (2011) only compares the obtained solutions with the best-known values (Packomania, 2020). Although the algorithm proposed by Huang & Ye (2011) solves the instances in shorter times, the increase in the solution time is not crucial. In fact, Algorithm 4.1.1 also verifies the solution quality and ensures a solution with 1% optimality gap with some additional solution time requirements.

We also observe that Algorithm 4.1.1 can solve the equal circle problem for the instances with larger number of circles as well as the solution times are shorter than the unequal circle problem. We believe that the observation is a result of the tighter initial lower bounds. Since the minimum idle regions between the circles do not change according to different configurations of circles for the equal circle case, and the obtained lower bounds are tighter for the equal circle packing problem.

5.4 Comparison with an Algorithm Given problem in Stoyan & Yas'kov (2004) within a Rectangle

For the unequal circle packing problems, we compare our algorithm with another algorithm proposed by Stoyan & Yas'kov (2004). First of all, the surrounding container is given as a rectangular strip in Stoyan & Yas'kov (2004) where the authors aim to minimize the length of the strip. To handle this issue, we changed the feasible region definitions in Algorithm 4.2.2 within the surrounding container a little bit, and the definitions of the sets $\mathcal{L}_c, \mathcal{S}_c$. For the other parts of the solution procedure, it is the same with the defined procedure.

During the modification process, the guiding points are included by the rectangular strip whose width is known. To ensure that a corresponding circle is totally included by the surrounding rectangle, guiding points are included by a rectangle whose boundary is r_c units away from the boundary for circle c. In addition, the idle regions between the circles and the surrounding container should be updated accordingly. For that, the idle region is equal to the region between the tangent passing through both circles. The reduced regions are calculated according to the shape of the container. For the two largest circles, the center of the first circle is located at the right upper part of the rectangle, and the center of the second circle is located on the upper part of the diagonal connecting the right upper and left lower corner points of the rectangle. Moreover, there are some set definitions within Algorithm 4.2.2 to be updated for which the distance of a point to the boundary is calculated according to the rectangular-shape.

With the procedures stated before, Algorithm 4.1.1 is updated. After that, we compare it with the algorithm proposed by Stoyan & Yas'kov (2004). Stoyan & Yas'kov (2004) obtain the initial solution by using the best-knowns' configurations or one-by-one circle placement algorithm (Stoyan & Yas'kov, 2004). We used the same labels for the instances which are given in the website (Packomania, 2020). During the comparisons, the initial upper bounds are obtained by the algorithm introduced problem in Huang et al. (2006) which is updated accordingly to pack the circles into rectangular strip since the algorithm introduced problem in Huang et al. (2006) performs better than their initialization approach.

Table 5.8 Results obtained by Algorithm 4.1.1 and the algorithm proposed by Stoyan & Yas'kov (2004)

			Algorithm 4.1.1			**Algorithm in Stoyan & Yas'kov (2004)**	
No.	Number of Circles	Strip Size (units)	Upper Bound	Solution Time (m)	GAP (%)	Upper Bound	Solution Time (m)
$SY1$	30	9.5	17.291	47.35	0.91	17.461	34.14
$SY2$	20	8.5	14.375	23.07	0.37	15.604	8.11
$SY2-1$	20	9.0	13.572	24.16	0.12	13.653	7.64
$SY2-2$	20	9.5	12.758	19.83	0.76	12.547	6.01
$SY2-3$	20	11.0	11.163	20.49	0.85	11.214	6.69
$SY3$	25	9.0	14.321	31.29	0.82	15.171	18.09
$SY3-1$	25	8.5	15.904	36.40	0.77	16.463	17.26
$SY3-2$	25	9.5	13.691	34.05	0.94	13.713	13.52
$SY3-3$	25	11.0	11.564	29.52	0.37	11.827	16.52

According to the results given in Table 5.8, the algorithm problem in Stoyan & Yas'kov (2004) performs worse in terms of the solution quality, and the optimality gap is unknown for these solutions. Stoyan & Yas'kov (2004) state that the proposed algorithm really depends on the initialization procedure, and the initial solution affects the solution quality of the algorithm. Also, Stoyan & Yas'kov (2004) requires an initial solution with all corresponding coordinates for the circles; however, Algorithm 4.1.1 requires an upper bound and a lower bound for the strip size during the initialization procedure which can be assigned as the sum of the radii of the circles and 0, respectively. Although the initial upper and lower bounds are not well-defined, Algorithm 4.1.1's performance is not affected adversely since it is a bisection-type algorithm and remove the half of the possible values at each iteration. However, when comparing the algorithms in terms of the solution times, we can say that our procedure requires more time to solve the instances. However, the strength of Algorithm 4.1.1 is guaranteeing the solution quality where Stoyan & Yas'kov (2004) does not propose any optimality gap during the solution procedure. Also, for some of the instances, Algorithm 4.1.1 finds better upper bounds than the algorithm in Stoyan & Yas'kov (2004).

5.5 Case Study from a Real Life Instance

Before finalizing the thesis, we will also talk about a case study we perform for a real life problem from automotive industry (Sugihara et al., 2004). During the manufacturing of automobiles, there is a problem of determining the dimensions

of a hole bundle of wires passes through. The dimension of this hole is crucial since the larger holes result in a decrease in the strength of the automobile. In the given instance, there are 162 wires passing from the hole, and this problem can be considered as a circle packing problem. The circles to be packed are given in Table 5.9.

Table 5.9 The dimensions of the 162 circles

Category of wire	23A	20G	31G	44G	45G	51G	25J	28J	30J	31J	22X	10E
Radius of circles (mm)	1.8	0.7	0.8	0.9	1.05	1.3	0.55	0.65	0.75	0.9	0.9	1.75
Number of circles to be packed	3	11	31	5	9	12	25	54	1	6	4	1

The real life instance includes a set of circles with the same dimensions, and there are ten different-sized groups of circles. To solve this instance, we design a solution procedure depending on clustering of these circles to get a small number of larger circles. Then, we run Algorithm 4.1.1 to pack the larger circles into another surrounding circle.

The clustering process determines the circles to be packed together into a larger circle. Hence, the crucial point is the amount of idle region within each cluster as well as between the resulting larger circles. As a result, we introduce an integer programming formulation for determining the clustering of the circles with the minimum total idle region.

Let m be the number of different clusters. For each cluster j, assume that the circles are divided into v groups, and $H_{i,j}$ is the set circles located in the i^{th} group of cluster j. A lower bound for the idle area is determined by formulation (4.1), and let $\mathcal{V}_{\mathcal{H}_{\rangle,|}}$ be the objective function value of formulation (4.1) solved for the circles in set $H_{i,j}$.

Denote the number of circles in $H_{i,j}$ with $t_{i,j}$, and let $\mathcal{R}_{i,j}$ be the minimum radius of the circle to pack the circles in $H_{i,j}$. Finally, the set of corresponding v new circles with radii of $\mathcal{R}_{1,j}, \cdots, \mathcal{R}_{v,j}$ is denoted by $H_{0,j}$ for the cluster j.

The decision variables $d^{i,j}_{c,k,l}$ and $f^{i,j}_{c,k,l}$ are defined for formulation 4.1 for each group $H_{i,j}$ of cluster j. Hence, the formulation is as follows:

$$\text{(5.1a)} \qquad \min_{j\in\{1,\cdots,m\}} \sum_{i\in\{0,\cdots,v\}} \mathcal{V}_{\mathcal{H}_{i,j}}$$

$$\text{s.t.}\mathcal{V}_{\mathcal{H}_{i,j}} = \min_{c,k,l\in\overline{\mathcal{C}}} \left\{ \sum_{\substack{c,k,l\in\mathcal{H}_{i,j} \\ :c<k<l}} \left(\Delta^{i,j}_{c,k,l} d^{i,j}_{c,k,l} - \rho^{i,j}_{c,k,l} f^{i,j}_{c,k,l}\right) \right\} \quad \begin{matrix} i\in\{0,\cdots,v\}, \\ j\in\{1,\cdots,m\} \end{matrix} \tag{5.1b}$$

$$\sum_{k,l\in\overline{\mathcal{C}}} d^{i,j}_{c,k,l} + \sum_{k,l\in\overline{\mathcal{C}}} d^{i,j}_{k,c,l} + \sum_{k,l\in\overline{\mathcal{C}}} d^{i,j}_{k,l,c} \geq min_c \qquad c\in\mathcal{H}_{i,j} \tag{5.1c}$$

$$\sum_{k,l\in\overline{\mathcal{C}}} d^{i,j}_{c,k,l} + \sum_{k,l\in\overline{\mathcal{C}}} d^{i,j}_{k,c,l} + \sum_{k,l\in\overline{\mathcal{C}}} d^{i,j}_{k,l,c} \leq max_c \qquad c\in\mathcal{H}_{i,j} \tag{5.1d}$$

$$f^{i,j}_{c,k,l} \leq d^{i,j}_{c,k,l} \qquad c,k,l\in\mathcal{H}_{i,j} \tag{5.1e}$$

$$f^{i,j}_{c,k,l} \leq d^{i,j}_{0,c,k}, \quad f^{i,j}_{c,k,l} \leq d^{i,j}_{0,k,l}, \quad f^{i,j}_{c,k,l} \leq d^{i,j}_{0,c,l} \qquad c,k,l\in\mathcal{H}_{i,j} \tag{5.1f}$$

$$f^{i,j}_{c,k,l} \geq d^{i,j}_{c,k,l} + d^{i,j}_{0,c,k} + d^{i,j}_{0,k,l} + d^{i,j}_{0,c,l} - 3 \qquad c,k,l\in\mathcal{H}_{i,j} \tag{5.1g}$$

$$d^{i,j}_{c,k,l}, f^{i,j}_{c,k,l} \in \{0,1\} \qquad c,k,l\in\mathcal{H}_{i,j} \tag{5.1h}$$

$$\mathcal{H}_{i,j} \in \mathcal{L}_j \qquad j\in\{1,\cdots,m\} \tag{5.1i}$$

In this formulation, the objective is the minimum required idle area for the cluster j. With the help of Constraint (5.1b), the minimum idle region in each group is determined. Constraints (5.1c)- (5.1g) are the constraints solved for each group $H_{i,j}$ of circles in each cluster j. Constraint (5.1h)-(5.1i) are domain constraints.

Table 5.10 The clusters corresponding to the obtained solution.

Cluster 1	1 : 1.8, 3 : 1.3, 10 : 0.65
Cluster 2	1 : 1.8, 3 : 1.3, 10 : 0.65
Cluster 3	1 : 1.8, 3 : 1.3, 10 : 0.65
Cluster 4	1 : 1.75, 3 : 1.05, 3 : 0.7, 8 : 0.65
Cluster 5	1 : 1.3, 3 : 1.05, 1 : 0.75, 2 : 0.7, 8 : 0.65
Cluster 6	1 : 1.3, 3 : 1.05, 3 : 0.7, 8 : 0.65
Cluster 7	1 : 1.3, 3 : 1.05, 3 : 0.7, 8 : 0.65
Cluster 8	16 : 0.8
Cluster 9	15 : 0.8
Cluster 10	15 : 0.9

By solving the clustering problems, we divided the circles into 10 groups without considering the circles with the smallest radius, 0.55. The groups are given in Table 5.10 where t is the number of the circles with radius r for the notation given by $t : r$. After that, we solve the larger ten circles packing problem with Algorithm 4.1.1. Then, we finalize the solution by perturbing the idle regions such that the circles with radius 0.55 can be placed.

The lower bound for the instance is obtained by identifying the minimum possible

area of the idle regions. To calculate this area, we consider that each circle has three triples as its neighbours, i.e., there will be at least 486 triples in the optimal solution. Starting from the group of circles with the smallest radius, we find the number of triples they can construct such that all three circles are the same as well as one or two of the three circles have the succeeding radius in the list where the circles are listed in ascending order. When the number of such triples reach 486, the corresponding areas for the determined triples are summed. The summation is the minimum possible area of the idle regions, and the total area of the circles is added to the summation to find a lower bound for the area of the surrounding circle. By using this lower bound for the area of the surrounding circle, a lower bound for the radius of the surrounding circle is determined. Finally, we find a solution with a gap around 4.15% where the radius of the outer layer is 12.45 mm.

The unequal circle packing problem is difficult to solve even for small number of circles. Even the instance includes some equal circles as well as the unequal ones, there is no solution for packing 162 circles into a single container given in the literature. For instance, Sugihara et al. (2004) packs 162 circles by dividing it into subgroups, and no solution is provided to pack all circles. In real life application, there is an analytic method calculated by multiplying a constant (a density factor assumed to be valid for any packing) with the sum of the areas of the circles. This analytic method is a basic calculation method which does not exploit any structural properties of the problem. Indeed, the proposed solution by the analytical model may not be feasible for a given instance since the density factor is a constant. Although it is possible for some heuristic to solve larger number of circles, the optimality gap cannot be provided by any solution. Hence, the solution quality with an optimality gap 4.15% is unique to our best knowledge in the literature.

6. CONCLUSIONS

The thesis considers the problem of packing a given set of circles into the minimum dimension surrounding container. There are two main rules of this problem: the circles do not overlap and each circle should be fully included in the surrounding container. Since the problem is hard to solve globally in the continuous space, we discretized the solution space into smaller cells. On the discretized solution space, we present a solution approach based on a bisection-type algorithm on the radius of the surrounding container. With the help of discretized solution space, we propose a restricted and a relaxed versions of the original problem.

In the restricted version, we assume that the centers of circles are located at one of the guiding points which are the corner points of the cells. For the restricted version, we introduce three different-sized MILP formulations: two formulations including linear number of decision variables, and another one containing logarithmic number of decision variables. Since the restricted version examines a subset of the solution space, it gives an upper bound for the radius of the surrounding container. Hence, we propose a relaxed version of the original problem. In the relaxed version, we allow some intersections of circles while restrain obviously infeasible solutions. Similarly, we introduce three more MILP formulations to get lower bounds for the surrounding container's radius: two formulations with linear number of decision variables, and a formulation with logarithmic number of decision variables.

Our algorithm solves the formulations designed for the restricted and the relaxed versions of the original problem at each iteration. Then, we propose some algorithmic enhancements to improve the performance of Algorithm 4.1.1 by exploiting the special structure of the original problem. We perform a computational study to analyze the performance of our algorithm for different configurations. We use three different type datasets: instances with unequal circles to pack into a circle from Zimmermann's (2005), the ones with equal circles packing into a circle from Huang & Ye (2011), and the instances containing unequal circles to pack into a rectangular strip with fixed width from Stoyan & Yas'kov (2004).

By our computational study, we first analyze the performance of the global solvers in CPP, and observe that the global solvers Baron and Gurobi cannot solve the instance even with 8-circle within 30 minutes time limit. During the solution process, we realized that the main problem of the global solvers is determining and improving the lower bound for the dimension of the surrounding circle. However, Algorithm 4.1.1 with logarithmic-sized formulations performs better for the same instances, and the instances are solved within 10 minutes with an optimality gap 1%. Even Algorithm 4.1.1 with linear-sized formulations dominates the global solvers in terms of the solution time and solution quality although it performs worse than Algorithm 4.1.1 with logarithmic-sized formulations. We also compare our algorithm with the solution methods given by Huang & Ye (2011) and Stoyan & Yas'kov (2004), where Huang & Ye (2011) considers the equal circle packing problem and Stoyan & Yas'kov (2004) tries to pack the circles into a rectangular strip. According to the results, even if our solution procedure needs more solution time than other algorithms given in the literature proposing heuristic solutions for the instances, the solution time of our algorithm does not increase significantly as well as our algorithm guarantees the solution quality. Thus, our algorithm performs better than or at least can compete with the proposed solution methods.

We also observe that Algorithm 4.1.1 solves the equal circle problem in shorter times than the unequal circle problem. The main reason of the observation is that the lower bound obtained for the equal circle packing problem case is tighter than the one obtained for the unequal circle packing problem because of the minimum idle regions between the circles do not change according to the configurations of circles for the equal circle case, hence, the algorithm starts with a better lower bound. Based on the literature review and the computational study, our another observation is that the main strength of our algorithm is providing a lower bound for the optimal value of the radius of the surrounding circle since there is almost no study providing this information during the solution procedure. The studies from literature generally compare the solutions with the best-known values and requires an initial configuration for positions of the circles to start the algorithms. Proposing lower bounds is also important for the problems which are solved for the first time with our algorithm since we can state the quality of our solution with the help of the obtained lower bound. For instance, there is no best-known for the instance given in Sugihara et al. (2004) as our best knowledge, but we know that the proposed solution is at most 4.15% away from the optimal solution.

BIBLIOGRAPHY

Castillo, I., Kampas, F. J., & Pintér, J. D. (2008). Solving circle packing problems by global optimization: Numerical results and industrial applications. *European Journal of Operational Research*, *191*(3), 786 – 802.

Castillo, I. & Sim, T. (2010). A spring-embedding approach for the facility layout problem. *JORS*, *61*, 1063.

Dowsland, K., Gilbert, M., & Kendall, G. (2007). A local search approach to a circle cutting problem arising in the motor cycle industry. *Journal of the Operational Research Society*, *58*(4), 429–438. cited By 8.

Drezner, Z. & Erkut, E. (1995). Solving the continuous p-dispersion problem using non-linear programming. *Journal of the Operational Research Society*, *46*(4), 516–520.

Francesco, C., Cerrone, C., & Cerulli, R. (2014). A tabu search approach for the circle packing problem. In *Proceedings of the 2014 17th International Conference on Network-Based Information Systems*, NBIS '14, (pp. 165–171)., USA. IEEE Computer Society.

Galiev, S. I. & Lisafina, M. S. (2013). Linear models for the approximate solution of the problem of packing equal circles into a given domain. *European Journal of Operational Research*, *230*(3), 505 – 514.

He, K., Huang, M., & Yang, C. (2015). An action-space-based global optimization algorithm for packing circles into a square container. *Computers & Operations Research*, *58*, 67 – 74.

Hifi, M. & M'Hallah, R. (2004). Approximate algorithms for constrained circular cutting problems. *Computers & Operations Research*, *31*(5), 675 – 694.

Hifi, M. & M'Hallah, R. (2009). A literature review on circle and sphere packing problems: Models and methodologies. *Adv. Operations Research*, *2009*, 150624:1–150624:22.

Huang, W. & Ye, T. (2011). Global optimization method for finding dense packings of equal circles in a circle. *European Journal of Operational Research*, *210*(3), 474 – 481.

Huang, W., Zeng, Z., Xu, R., & Fu, Z. (2012). Packing unequal disks in a circular container using a population based algorithm. In *2012 Sixth International Conference on Genetic and Evolutionary Computing*, (pp. 437–440).

Huang, W. Q., Li, Y., Akeb, H., & Li, C. M. (2005). Greedy algorithms for packing unequal circles into a rectangular container. *Journal of the Operational Research Society*, *56*(5), 539–548.

Huang, W. Q., Li, Y., Jurkowiak, B., Li, C. M., & Xu, R. C. (2003). A two-level search strategy for packing unequal circles into a circle container. In Rossi, F. (Ed.), *Principles and Practice of Constraint Programming – CP 2003*, (pp. 868–872)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Huang, W. Q., Li, Y., Li, C. M., & Xu, R. C. (2006). New heuristics for packing unequal circles into a circular container. *Comput. Oper. Res.*, *33*(8), 2125–2142.

Li, W. & Sun, T. (2009). Silica artificial opal incorporated with silver nanoparticles. *Materials Chemistry and Physics*, *116*.

Litvinchev, I., Infante, L., & Ozuna Espinosa, E. (2015a). *Approximate Packing:*

Integer Programming Models, Valid Inequalities and Nesting, volume 105, (pp. 117–135). Springer.

Litvinchev, I., Infante, L., & Ozuna Espinosa, E. (2015b). Packing circular-like objects in a rectangular container. *Journal of Computer and Systems Sciences International*, *54*, 259–267.

Litvinchev, I. & Ozuna, E. (2014). Approximate packing circles in a rectangular container: Valid inequalities and nesting. *Journal of Applied Research and Technology*, *12*(4), 716 – 723.

Locatelli, M. & Raber, U. (2002). Packing equal circles in a square: A deterministic global optimization approach. *Discrete Appl. Math.*, *122*(1-3), 139–166.

López, C. & Beasley, J. (2016). A formulation space search heuristic for packing unequal circles in a fixed size circular container. *European Journal of Operational Research*, *251*(1), 64 – 73.

Maranas, C. D., Floudas, C. A., & Pardalos, P. M. (1995). New results in the packing of equal circles in a square. *Discrete Mathematics*, *142*(1), 287 – 293.

Packomania (2020). Packomania website. `http://www.packomania.com/`. Accessed: 2021-01-30.

Stoyan, Y. & Yas'kov, G. (2004). A mathematical model and a solution method for the problem of placing various-sized circles into a strip. *European Journal of Operational Research*, *156*(3), 590 – 600.

Stoyan, Y. & Yaskov, G. (2012). Packing equal circles into a circle with circular prohibited areas. *International Journal of Computer Mathematics*, *89*, 1355 – 1369.

Sugihara, K., Sawai, M. K., Sano, H. S., Kim, D.-S., & Kim, D. (2004). Disk packing for the estimation of the size of a wire bundle. *Japan Journal of Industrial and Applied Mathematics*, *21*, 259–278.

Szabó, P. (2000). Some new structures for the "equal circles packing in a square" problem. *CEJOR. Central European Journal of Operations Research*, *8*.

Torres, R., Marmolejo, J., & Litvinchev, I. (2020). Binary monkey algorithm for approximate packing non-congruent circles in a rectangular container. *Wireless Networks*, *26*, 4743–.

Vielma, J., Ahmed, S., & Nemhauser, G. (2010a). Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, *58*, 303–315.

Vielma, J. P., Ahmed, S., & Nemhauser, G. (2010b). A note on "a superior representation method for piecewise linear functions". *INFORMS Journal on Computing*, *22*(3), 493–497.

Wang, H., Huang, W., Zhang, Q., & Xu, D. (2002). An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, *141*(2), 440 – 453.

Zeng, Z., Yu, X., He, K., Huang, W., & Fu, Z. (2016). Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. *European Journal of Operational Research*, *250*(2), 615 – 627.

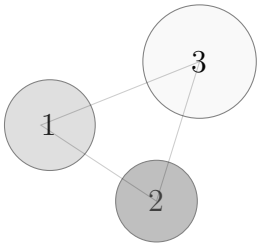
Zimmermann's, C. (2005). Al zimmermann's programming contests. `http://www.recmath.org/contest/CirclePacking/`. Accessed: 2021-01-30.

APPENDIX A

The proof of Theorem 4.1

Before the proof of Theorem 4.1, we will give some notation and a lemma. Consider the adjacent circles $1,\cdots,k$ with edges $(1,2),\cdots,(k-1,k),(k,1)$. The area of the polygon connecting the centers of the circles $1,\cdots,k$ is denoted by $A_{1,\cdots,k}$, and the angle at the corner point c of this polygon is shown by a_c.

Figure A.1 Idle region between 3 adjacent circles where $e_{c,l} > r_c + r_l$



Lemma A.1. *For $0 \le a_1 \le \pi$, $f(a_1)$ is a decreasing function of a_1 defined as follows where $r_1 \le r_2 \le r_3$:*

$$f(a_1) = (r_1+r_2)(r_1+r_3)\sin(a_1) - (r_1)^2 a_1 \\ -(r_2)^2 \sin^{-1}\left(\tfrac{(r_1+r_3)}{e_{2,3}} sin(a_1)\right) - (r_3)^2 \sin^{-1}\left(\tfrac{(r_1+r_2)}{e_{2,3}} sin(a_1)\right).$$

Proof of Lemma A.1. $f'(a_1)$ is given as follows:

$$f'(a_1) = \cos(a_1)(r_1+r_2)(r_1+r_3) - (r_1)^2 \\ -\cos(a_1)\frac{(r_2)^2(r_1+r_3)}{e_{2,3}\sqrt{1-\left(\frac{(r_1+r_3)\sin(a_1)}{e_{2,3}}\right)^2}} - \cos(a_1)\frac{(r_3)^2(r_1+r_2)}{e_{2,3}\sqrt{1-\left(\frac{(r_1+r_2)\sin(a_1)}{e_{2,3}}\right)^2}}$$

$$= \cos(a_1)(r_1+r_2)(r_1+r_3) - (r_1)^2 \\ -\cos(a_1)\left(\frac{(r_2)^2(r_1+r_3)}{\sqrt{(e_{2,3})^2-(r_1+r_3)^2\sin^2(a_1)}} + \frac{(r_3)^2(r_1+r_2)}{\sqrt{(e_{2,3})^2-(r_1+r_2)^2\sin^2(a_1)}}\right).$$

Assume that $e_{2,3} = r_2 + r_3 + K$ where $K \ge 0$. From the cosine's theorem, the following equality is known:

$$(r_1)^2 + r_1r_2 + r_1r_3 - (r_1+r_2)(r_1+r_3)\cos(a_1) = r_2r_3 + r_2K + r_3K + K^2.$$

Then, the $\cos(a_1)$ and $\sin(a_1)$ values are calculated as follows by using the cosine's theorem and the formulations for calculating the area of the triangle:

$$\cos(a_1) = \frac{(r_1)^2+r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2}{(r_1+r_2)(r_1+r_3)}$$

and

$$\sin(a_1) = \frac{\sqrt{(2r_1+2r_2+2r_3+K)(2r_1-K)(2r_2+K)(2r_3+K)}}{2(r_1+r_2)(r_1+r_3)}.$$

Hence, $f'(a_1)$ can be written as follows:

$$\begin{aligned} f'(a_1) = {} & r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2 \\ & -\frac{2((r_1)^2+r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2)(r_2)^2}{\sqrt{4(r_1+r_2)^2(r_2+r_3+K)^2-(2r_1+2r_2+2r_3+K)(2r_1-K)(2r_2+K)(2r_3+K)}} \\ & -\frac{2((r_1)^2+r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2)(r_3)^2}{\sqrt{4(r_1+r_3)^2(r_2+r_3+K)^2-(2r_1+2r_2+2r_3+K)(2r_1-K)(2r_2+K)(2r_3+K)}}. \end{aligned}$$

Name the parts of $f'(a_1)$ as given below:

$$f'_1(a_1) = r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2$$

$$f'_2(a_1) = 2((r_1)^2+r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2)(r_2)^2$$

$$f'_3(a_1) = \sqrt{4(r_1+r_2)^2(r_2+r_3+K)^2-(2r_1+2r_2+2r_3+K)(2r_1-K)(2r_2+K)(2r_3+K)}$$

$$f'_4(a_1) = 2((r_1)^2+r_1r_2+r_1r_3-r_2r_3-r_2K-r_3K-K^2)(r_3)^2$$

$$f'_5(a_1) = \sqrt{4(r_1+r_3)^2(r_2+r_3+K)^2-(2r_1+2r_2+2r_3+K)(2r_1-K)(2r_2+K)(2r_3+K)}$$

Then, $f'(a_1)$ is written like below:

$$f'(a_1) = f'_1(a_1) - \frac{f'_2(a_1)}{f'_3(a_1)} - \frac{f'_4(a_1)}{f'_5(a_1)} \leq 0 \tag{A.1}$$

Finally, since $r_1 \leq r_2 \leq r_3$, we know that $a_1 \geq a_2 \geq a_3$. By using this knowledge, $a_1 \geq \frac{\pi}{3}$, i.e., $-1 < \cos(a_1) \leq 0.5$. From this knowledge, the following inequalities are true:

$$0 \leq 2(r_1)^2+2r_1r_2+2r_1r_3-r_2K-r_3K-K^2$$

$$0 \leq 3r_2r_3+2r_2K+2r_3K+2K^2-(r_1)^2-r_1r_2-r_1r_3$$

Also, $2r_1 \geq K$ is true from the triangle inequality. By using this knowledge, we can verify the inequality which is a rewritten version of Equation (A.1) given below:

$$2\Big(f'_1(a_1)f'_2(a_1)f'_5(a_1)\Big)^2 f'_3(a_1) + 2\Big(f'_1(a_1)f'_3(a_1)f'_4(a_1)\Big)^2 f'_5(a_1)$$

$$+2\Big(f_2'(a_1)f_4'(a_1)\Big)^2 f_3'(a_1)f_5'(a_1) - \Big(f_1'(a_1)\Big)^4\Big(f_3'(a_1)f_5'(a_1)\Big)^2$$
$$-\Big(f_2'(a_1)\Big)^4\Big(f_5'(a_1)\Big)^2 - \Big(f_4'(a_1)\Big)^4\Big(f_3'(a_1)\Big)^2 \leq 0.$$

Hence, $f'(a_1) \leq 0$ is true. Then, $f(a_1)$ is a decreasing function of a_1 as stated. □

Theorem 4.1 $Idle_{1,\cdots,k} \leq Idle^*_{1,\cdots,k}$.

Proof of Theorem 4.1. For $k = 3$, we will show that $Idle_{1,2,3} \leq Idle^*_{1,2,3}$. For illustration, consider the idle region between circles 1,2, and 3 in Figure A.1. In this configuration, the length of edge connecting the centers of circles c and l is given with $e_{c,l}$. In addition, the area of the triangle with corner points located at the centers of circles is denoted by $B_{1,2,3}$ where its angle at the corner located at circle c's center is denoted by b_c. Then, the idle region $Idle^*_{1,2,3}$ is calculated as follows:

$$Idle^*_{1,2,3} = B_{1,2,3} - \frac{(r_1)^2 b_1}{2} - \frac{(r_2)^2 b_2}{2} - \frac{(r_3)^2 b_3}{2}.$$

In the following configuration, relocate the center of circle 2 (circle 3) to the point on the edge connecting circles 1 and 2 (circles 1 and 3) which is $r_1 + r_2\,(r_1 + r_3)$ units away from the center of circle 1. The new triangle's area is denoted by $D_{1,2,3}$ where b_c shows the angle of the triangle at the center of circle c:

$$D_{1,2,3} = \frac{(r_1+r_2)(r_1+r_3)\sin(b_1)}{2} \leq \frac{(e_{1,2})(e_{1,3})\sin(b_1)}{2} = B_{1,2,3}.$$

Then, the idle region in this configuration is calculated as follows:

$$D_{1,2,3} - \frac{(r_1)^2 b_1}{2} - \frac{(r_2)^2 b_2}{2} - \frac{(r_3)^2 b_3}{2} \leq B_{1,2,3} - \frac{(r_1)^2 b_1}{2} - \frac{(r_2)^2 b_2}{2} - \frac{(r_3)^2 b_3}{2}$$

In the first configuration, if the length of the edge connecting circles 2 and 3 is $(r_2 + r_3)$ units, this concludes with $Idle_{1,2,3} \leq Idle^*_{1,2,3}$. Otherwise, the length of the edge connecting circles 2 and 3 is larger than $(r_2 + r_3)$ which results in $a_1 \leq b_1$.

$$\begin{aligned}
Idle_{1,2,3} &= A_{1,2,3} - \frac{(r_1)^2 a_1}{2} - \frac{(r_2)^2 a_2}{2} - \frac{(r_3)^2 a_3}{2} \\
&= \frac{(r_1+r_2)(r_1+r_3)\sin(a_1)}{2} - \frac{(r_1)^2 a_1}{2} - \frac{(r_2)^2 a_2}{2} - \frac{(r_3)^2 a_3}{2} \\
&\leq \frac{(r_1+r_2)(r_1+r_3)\sin(b_1)}{2} - \frac{(r_1)^2 b_1}{2} - \frac{(r_2)^2 b_2}{2} - \frac{(r_3)^2 b_3}{2} \ \Big(\text{by Lemma A.1}\Big) \\
&= D_{1,2,3} - \frac{(r_1)^2 b_1}{2} - \frac{(r_2)^2 b_2}{2} - \frac{(r_3)^2 b_3}{2} \\
&\leq B_{1,2,3} - \frac{(r_1)^2 b_1}{2} - \frac{(r_2)^2 b_2}{2} - \frac{(r_3)^2 b_3}{2} = Idle^*_{1,2,3}.
\end{aligned}$$

Then, assume that $Idle_{1,\cdots,k} \leq Idle^*_{1,\cdots,k}$.Then, $Idle_{1,\cdots,k,(k+1)} \leq Idle^*_{1,\cdots,k,(k+1)}$ is true for $(k+1)$ as follows:

$$\begin{aligned} Idle^*_{1,\cdots,k,(k+1)} &= Idle^*_{1,\cdots,k} + Idle^*_{1,k,(k+1)} \\ &\geq Idle_{1,\cdots,k} + Idle^*_{1,k,(k+1)} \\ &\geq Idle_{1,\cdots,k} + Idle_{1,k,(k+1)} = Idle_{1,\cdots,k,(k+1)}. \end{aligned}$$

Hence, $Idle_{1,\cdots,k} \leq Idle^*_{1,\cdots,k}$ by induction. □

The proof of Theorem 4.2

Proposition A.1. *Suppose a univariate function f is continuous on $[L,U]$ and it is differentiable on (L,U). If the derivative is smaller than or equal to zero for all R in (L,U), then the function is non-increasing on $[L,U]$.*

Proof of Theorem 4.2. Assume that the idle region between the surrounding circle with radius R and the circles c and k is denoted by $E^R_{0,c,k}$. Then, the idle region $Idle^R_{0,c,k}$ is calculated with the formula given below for circles r_c and r_k where the radius of the surrounding circle is R:

$$\begin{aligned} Idle^R_{0,c,k} = &\frac{(R)^2 cos^{-1}\left(\frac{(R)^2 - Rr_c - Rr_k - r_c r_k}{(R)^2 - Rr_c - Rr_k + r_c r_k}\right)}{2} - \sqrt{(R)(r_c)(r_k)(R - r_c - r_k)} - \frac{\pi((r_c)^2 + (r_k)^2)}{2} \\ &- \frac{(r_c)^2 cos^{-1}\left(\frac{(r_c)^2 + r_c r_k - Rr_c + Rr_k}{Rr_c + Rr_k - (r_c)^2 - r_c r_k}\right)}{2} - \frac{(r_k)^2 cos^{-1}\left(\frac{(r_k)^2 + r_c r_k - Rr_k + Rr_c}{Rr_c + Rr_k - r_c r_k - (r_k)^2}\right)}{2}. \end{aligned}$$

Since, we do not know the optimal value of R in our problem, we need to show that the idle region is decreasing function of R between $L \leq R \leq U$. So, we can calculate the idle regions by using a known upper bound for R. So, we will calculate the derivatives on R of this area:

$$\begin{aligned} \left(Idle^R_{0,c,k}\right)' = &Rcos^{-1}\left(\frac{(R)^2 - Rr_c - Rr_k - r_c r_k}{(R)^2 - Rr_c - Rr_k + r_c r_k}\right) + \frac{R^2\sqrt{r_c r_k R(R - r_c - r_k)}(2R - r_c - r_k)}{2R(R - r_c)(R - r_k)(R - r_c - r_k)} \\ &+ \frac{(r_c)^2\sqrt{r_c r_k R(R - r_c - r_k)}(R - r_k)}{2R(R - r_c)(R - r_k)(R - r_c - r_k)} + \frac{(r_k)^2\sqrt{r_c r_k R(R - r_c - r_k)}(R - r_c)}{2R(R - r_c)(R - r_k)(R - r_c - r_k)} \\ &- \frac{\sqrt{r_c r_k R(R - r_c - r_k)}(R - r_c)(R - r_k)(2R - r_c - r_k)}{2R(R - r_c)(R - r_k)(R - r_c - r_k)} \end{aligned}$$

and

$$\left(Idle^R_{0,c,k}\right)'' = cos^{-1}\left(\frac{(R)^2 - Rr_c - Rr_k - r_c r_k}{(R)^2 - Rr_c - Rr_k + r_c r_k}\right)$$

$$+\sqrt{r_c r_k}\frac{-4R^5+8r_cR^4+8r_kR^4-7(r_c)^2R^3-7(r_k)^2R^3-10r_cr_kR^3+3(r_c)^3R^2+3(r_k)^3R^2}{2\sqrt{R(R-r1-r2)}(R-r_c)^2(R-r_k)^2(R-r_c-r_k)}$$

$$+\sqrt{r_c r_k}\frac{7r_c(r_k)^2R^2+7r_k(r_c)^2R^2-3r_c(r_k)^3R-6(r_cr_k)^2R-3(r_c)^3r_kR+2(r_c)^2(r_k)^3+2(r_c)^3(r_k)^2}{2\sqrt{R(R-r1-r2)}(R-r_c)^2(R-r_k)^2(R-r_c-r_k)}.$$

We should show that $\left(Idle^R_{0,c,k}\right)$ is a non-increasing function between $L \leq R \leq U$ to show $Idle^R_{0,c,k} < Idle^{R^*}_{0,c,k}$. Then, if $Idle^{R_1}_{0,c,k} \geq Idle^{R_2}_{0,c,k}$ if $R_1 \leq R_2$ and $\left(Idle^R_{0,c,k}\right)''$ has no roots between $L \leq R \leq U$ for the given bounds L and U; it results in that the direction of the function $Idle^R_{0,c,k}$ does not change within the interval $L \leq R \leq U$.

Hence, if $\left(Idle^{\frac{L+U}{2}}_{0,c,k}\right)' < 0$, the function $Idle^R_{0,c,k}$ is non-increasing for$L \leq R \leq U$ by Proposition A.1. □