University of São Paulo

Institute of Mathematics and Computer Science

Undergraduate Project II

Bio-inspired algorithms for packing identical unitary radius circles within a circular region

Author:
Tiago Moreira Trocoli da Cunha

Supervisor:
Dr. Marina Andretta

January 26, 2018

Acknowledgments

I would first like to give a special thanks to my supervisor Dr. Marina Andretta of the Institute of Mathematics and Computer Science at University of São Paulo because her help gave this document took its actual form. Also, I owe a debt of gratitude to Tatiana, my therapist, who assisted me throughout this stage of life. And for everyone indirectly involved, friends and family, words are not enough to express my gratitude.

Abstract

The second final year project addressed a NP-hard problem, which was the minimization of the radius of a circular container to pack a given number of identical unitary radius circles without having overlapping regions between them. Exact methods can solve it but they may take hours to solve instances with thirty to fifty circles. Thus, the objective was to develop an effective bio-inspired heuristic that gives "good" results. Numerical experiments show that the proposed heuristic takes few seconds to solve each instance with solutions up to 4% worse than an exact method, in most cases.

Keywords: packing problem, bio-inspired algorithms, differential evolution, particle swarm optimization, quasi-physical method.

Contents

Chapter 1

Introduction

Packing problems are a class of optimization problems that involve either putting as many objects in a given container as possible or minimizing the region of a container to pack a given number of objects. It is applied not only in the industry, but in our daily life, having great interest in mathematical and economical standpoint. For example, in warehousing, objects have to be put in shelves and optical fibers need to be inserted in a piper with perimeter as small as possible [2]. In regard to the second final year project, due to short time, we will only work on minimizing the radius of a circular container to pack a given number of identical unitary radius circles, without having overlapping regions between them.

This kind of problem was proved to be NP-hard [8] leading researchers to develop heuristic techniques to find approximate solution in a reasonable time, even if they are not rigorous. However, they also have been trying to tackle it by employing non-linear programming methods. For instance, in [4], the authors developed Reformulation Descent Algorithm, and affirmed it is more effective than the classical non-linear methods for up to 100 items. Other effective approach is described in [7], where the researchers employ a hybrid Simulating Annealing and Tabu Search. Huang and Kang presented a heuristic quasi-physical strategy [5] which "finds natural phenomena in the physical world equivalent to the original mathematical problem". However in [6], it was used a modified Simulating Annealing to try to jump out of local minimum that outperformed the quasi-physical and quasi-human techniques. It is important to mention that the Particle Swarm Optimization, one of the algorithms we worked on this project, was applied successively to pack as many items as possible into a given fixed-shaped figure [3].

1.1 Motivation

The key factor of heuristic and meta-heuristic comparing to exact optimization techniques is the ability to solve large-scale instances within reasonable time, with the price of losing a guarantee for achieving the optimal solution. Thus, heuristics are an appropriate choice for solving large instances of NP-hard problems.

Heuristics inspired by nature, also known as bio-inspired algorithms, have been proven successful in optimizing a large amount of problems, such as scheduling, vehicle routing, protein folding, image processing, to mention but a few [12]. Among them are Particle Swarm Optimization (PSO) and Differential Evolution (DE), two continuous bio-inspired algorithm.

Although there are some experiments using bio-inspired algorithms for packing problems, we didn't find any articles using them to minimize the radius of circular container given a fixed number of circles inside it. Moreover, exact methods can solve the problem but they may take hours to solve instances with thirty to fifty circles [1].

1.2 Objective and Methodology

The objective of this project is to use PSO and DE combined with mathematical or natural model of the problem to speed up the resolution of packing identical unitary radius circles in a circular container, minimizing its radius and without overlapping, in such a way that gives "good" results.

The results will be compared with a method based on nonlinear programming in [1]. For this project to work successfully, we need to research articles about bio-inspired algorithms and how researchers deal with similar problems. Based on them, we need to carry out many experiments by changing or developing a new method, tuning PSO's and DE's parameters, modifying the model and etc, always comparing with [1].

1.3 Hypothesis

We want to investigate the behavior of bio-inspired algorithms towards the packing problem in terms of efficacy and efficiency, so we expect them to solve the problem faster than the exact method in [1] with "good" results.

1.4 Outline of the Monograph

Chapter 2 states mathematically the packing problem. Chapter 3 makes up most of this document, we give a brief review of the history of bio-inspired algorithms, we explain the Differential Evolution, Particle Swarm Optimization and Quasi-physical method.

In Chapter 4, we set forth the heuristics that we developed using algorithms previously described in Chapter 2. Chapter 5 lays out how the experiments were carried out, the results of them and the validation of the hypothesis. The final chapter sets out the conclusion of this project plus a brief proposal for a future work. Finally, the appendix shows fifty pictures of the solutions obtained by our best proposed heuristic.

Chapter 2

Statement of the Problem

Given a number of unitary circles, we want to find a circular container with its radius as small as possible, such that, all the unitary circles are positioned in it and without overlapping regions between them. The Figure 2.1 illustrates this problem.

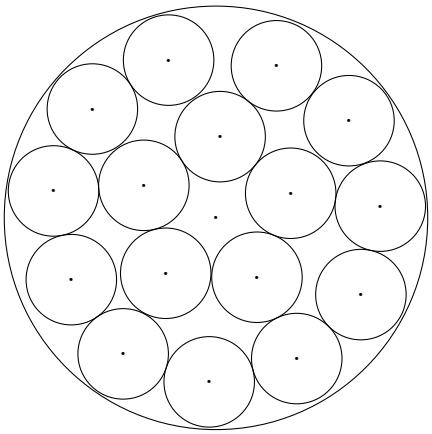


Figure 2.1: An illustrative example of the problem.

Let $c_i = (x_i, y_i) \in \mathbb{R}^2$ and $R_i \in \mathbb{R}^2$ be the center and radius of the ith circle, respectively, for $i = 1, ..., n$. Owing to the overlapping constraint on circles, the optimization must guarantee that it is not violated, so the distance between any two circles is required to be equal to or greater than the sum of their radius. Inequalities (2.1) and Figure 2.2 represent this constraint.

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq R_i + R_j, \forall i \neq j. \tag{2.1}$$

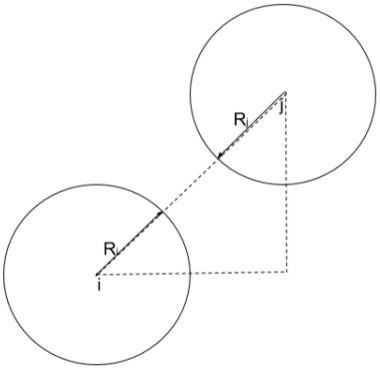


Figure 2.2: An illustrative example of the Inequalities (2.1).

Let us assume the circular container, with radius R_0, is centered in the origin of Cartesian plane. Since the circles must be packed in the circular container, the distance between each circle's and container's center is required to be at most the difference of the latter's and former's radius, in this order. Inequalities (2.2) and Figure 2.3 represent this constraint.

$$\sqrt{x_i^2 + y_i^2} \leq R_0 - R_i, \forall i. \tag{2.2}$$

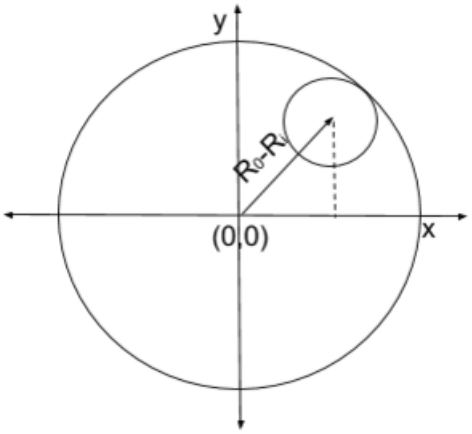


Figure 2.3: An illustrative example of the inequalities 2.2.

Therefore, the mathematical model of this problem is expressed below:

$$\begin{aligned} &\textit{minimize} && R_0 \\ &\textit{subject to} && \sqrt{x_i^2 + y_i^2} \leq R_0 - R_i, \forall i, \\ & && \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq R_i + R_j, \forall i \neq j. \end{aligned} \tag{2.3}$$

A question that may arise is regarded to the number of constraints and variables relating to n (number of circles). Since, there are two variables for each circle plus the circular container, we have $k = 2n + 1$ variables. The first constraint (2.1) is applied to each object with relation to $n-1$ others plus the second constraint, so it is straightforward to verify that $m = \frac{n(n-1)}{2} + n$. In other words, the number of constraints and variables are in the order of magnitude of $O(n^2)$ and $O(n)$, respectively.

Chapter 3

Literature Review

Classical nonlinear optimization algorithms are exact method that are based on mathematical analysis to find an optimal solution. Some of well-know techniques are Newton Methods, Quasi-Newton Methods and Gradient Descent. Some of them rely on a model of the response-sampling surface and based on some mathematical theorems, they decide which direction to take at the local surface [15]. But an emerging new group of algorithms takes a different perspective when dealing with optimization problems.

The development of these new algorithms takes its source of inspiration in the nature. After all, analyzing the nature one can find many optimization techniques that have been enhancing during million of years having the impressive feature of solving difficult task out of simple behaviors, sometimes without previous knowledge of the task.

One of the first articles about bio-inspired algorithm is dated in the late of 1950s by the name of "A Learning Machine, Part I" from the IBM researchers R. M. Friedberg, B. Dunham, and J. H. North [13, 14]. In spite of this, it could be a surprise to many people that Alan Turing made history, once more, when conceptualized the Unorganized Machine, possibly the first attempt to simulate a neural network [17]. Furthermore, he was ahead of his time again, when he came up with the idea of "genetical search" to train his artificial neural network, and he wrote: "picture of the cortex as an unorganized machine is very satisfactory from the point of view of evolution and genetics" [16]. Unfortunately, his early death let these breakthrough delayed for a few decades.

Since then, various bio-inspired algorithms have been developed, as shown in Table 3.1, bringing into existence sub-fields in which these algorithms are classified to, as described below:

- Swarm Intelligence: this class of algorithms finds its inspiration from collective behavior of social insects.
- Evolutionary Algorithm: these methods are inspired by the mechanics of biological evolution.
- Artificial Immunity System: this class of algorithm concerns the simulation of a model of mammal's immune system.
- Neural Network: as the name suggests, this class involves mathematical models of neural network in computers which is used in optimization, machine learning and other fields.

Throughout this document we will consider optimization in terms of minimization. Also, we use the terms *exploration* and *exploitation*. In the context of optimization, exploration is the process of visiting entirely new regions of a search space, whilst exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points.

Table 3.1: Bio-inspired Algorithms

Evolutionary Algorithm	**Swarm Intelligence**
Genetic algorithm	Ant Colony Optmization
Genetic programming	Particle Swarm Optimization
Evolutionary programming	Cuckoo Search
Evolutionary Strategy	Bees Algorithm
Differential Evolution	Bee Colony Optimization
Artificial Immunity System	**Neural Network**
Dendritic Cell Algorithms	Hopfield network
Negative Selection Algorithm	Kohonen Self-Organizing Maps
Clonal Selection Algorithm	Feedforward neural network
Immune Network Algorithm	Recurrent neural network

3.1 Concepts of Evolutionary Algorithms

Charles Darwin, an English naturalist, was the founder of the theory of natural evolution. In his book entitled "On the Origin of Species" he gave his observations, facts and conclusions while traveling around the world that lead him to formulate the Theory of Evolution by Natural Selection [23]. Darwin's theory is constituted of four principles: "more offspring are produced than can survive given selection pressures", "phenotypic variation exists among individuals and the variation is heritable", "those individuals with heritable traits better suited to the environment will survive" and "when reproductive isolation occurs new species will form" [21,22].

"Evolution is change in the heritable characteristics of biological populations over successive generations" [19]. These changes occur through natural selection, survival of the fittest, reproduction, mutation, competition and symbiosis. It is important to mention that, by no means evolution happens in a single organism, but only in populations. The changes must be passed on to generations via reproduction, so evolution transcend the lifetime of a single individual.

We can consider evolution as an optimization process which aims to improve the ability of the system to survive in dynamically changing and competitive environments [8]. So, evolutionary algorithm (EA) refers to population based stochastic problem solving technique that simulates the evolutionary processes, such as natural selection, survival of the fittest and reproduction.

Also, EA has a set of individuals that are represented by chromosomes that correspond to solutions. On each iteration, the whole population is reproduced using crossover and having its fitness evaluated. Below are the definition of the EA's parameters.

- A *phenotype* is a candidate solution to the problem.
- A *chromosome* or *individual* is an encoding representation of a phenotype.
- A *Gene* or *variable* is a region of a chromosome.
- A *population* is a set of chromosomes.
- A *generation* is the population on a single iteration.
- *Fitness* is the measure of the performance of a phenotype on the problem.

In terms of computational implementation, population is a data structure composed of individuals. Individuals are also data structures that are made of variables with primitive data types. EA has four processes known as genetic operations, described below.

3.1.1 Reproduction

Reproduction is a biological process where new individuals, called offspring, are produced from their parents through crossover. Putting aside the biological definition, reproduction can be simulated as a mathematical process in which two or more solutions, generate one or more others, via crossover, based on some rules, generally involving probability. There are various type of crossover in the literature, we explain briefly three of them.

The one-point crossover selects one segment of each parent's chromosome to make up a new chromosome. Two point crossover selects two points of parents' chromosome and the segment between this two points of one parent's chromosome plus the remaining segment of another is combined to generate a new chromosome. The uniform crossover selects various segments of parents' chromosome to generate a new one. The Figure 3.1 illustrate these crossover operators where the rectangles represent the two parents' chromosome and their offspring.

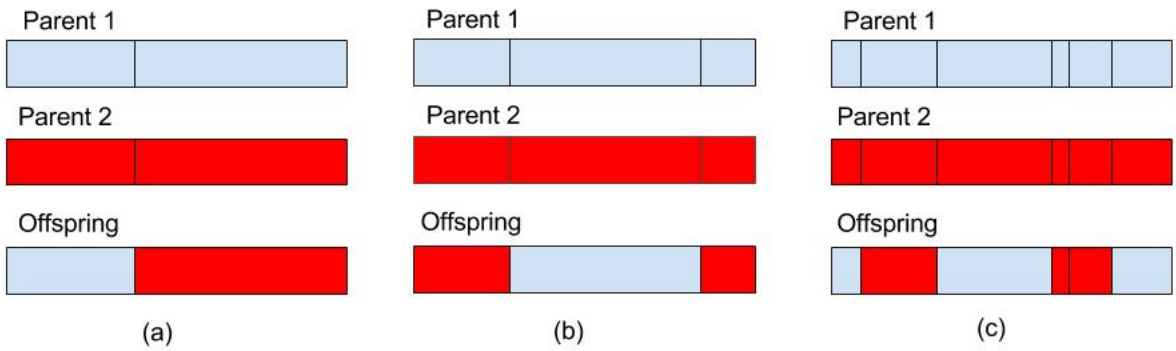


Figure 3.1: Graphical illustration of one-point (a), two-point (b) and uniform crossover (c).

3.1.2 Selection

Selection can be viewed as sexual selection that occurs within the reproduction process in which individuals are selected based on their fitness to generate offspring. Individuals who have high fitness, in other words, are more adapted and have more probability to be chosen. Therefore, this operation is directly related to EA's convergence and its exploitation. If the selection is too "aggressive", it will impose adaptive traits sooner and will not allow the EA have time to explore more possibilities in the search space which can lead to premature convergence. Conversely, if the selection is too "soft", EA's convergence will take longer to converge.

3.1.3 Mutation

Mutation is a rare event wherein chromosome is changed during the reproduction process by chance. By imposing randomness on EA, mutation is directly related to exploration. So the objective of mutation is to enhance the exploration characteristic of the EA. It is important to mention that mutation must be balanced carefully or EA will become nothing less than a brute force algorithm.

3.1.4 Elitism

Elitism is the process in which the best individuals are chosen to make up the next generation. In other words, the fittest individuals "survive" and unfit "die" in the virtual environment. It is a way to simulate the natural selection. So, elitism is directly related to exploration and exploitation features. If the selective pressure is aggressive, the EA probably will be trap in a

local minimum. Therefore, in order to balance it, the EA needs to allow some individuals who are not the fittest to survive. There are several ways to do this, like taking only the 10% of the best and the others could be chosen at random.

3.1.5 Standard Evolutionary Algorithm

There are numbers of ways to develop EA, as it was mentioned in Table 3.1 and these number is increasing each day. Although each of them has its own particularity, they all fall into a similar general form. First, we have to define a stopping condition, that could be a maximum number of generations, when an average value of population's fitness is below a given number and etc. Second, EA has to set up the first population in which the individuals' variables are chosen randomly. Third, it has to perform the selection, reproduction, mutation and selection of the new generation by elitism, in this order. The third part is processed again and again until the stopping condition is met, and then EA determines the fittest individual. Algorithm 1 describes this procedure.

Algorithm 1 Standard Evolutionary Algorithm [20]

Input: The numbers of individuals n_M and variables n_S.
Output: The fittest individual, $x_{fittest}$.
Require: A given evaluation function, $f(x)$.

1: Let $t = 0$.
2: Let $G_t = \left\{x_1^t, ..., x_{n_M}^t\right\}$ be the first generation consisting of n_M individuals with n_S variables each.
3: **while** *stopping condition isn't true* **do**
4: Evaluate the fitness, $f(x_i^t)$, of each individual in G_t.
5: Perform reproduction to create offspring.
6: Mutate the offspring.
7: Select the new population, G_{t+1}.
8: Advance to the new generation, $t = t + 1$.
9: **end while**
10: Determine the fittest individual, $x_{fittest} \in G_t | f(x_{fittest}) = \min\left\{f(x_1^t), ..., f(x_{n_M}^t)\right\}$.

3.2 Differential Evolution

Differential Evolution (DE) is a population based stochastic technique developed by Storn and K. Price [34] and applied to continuous, not necessarily differentiable, optimization. The authors were trying to solve the Chebychev Polynomial Fitting Problem in the middle of 1990s, when they came up with the ideal of using vector differences for perturbing the vector population, the DE grew out of this research as a "simple and efficient heuristic for global optimization". It is believed to be one of the most popular meta-heuristics nowadays with a large number of real-world applications [35]. Since there are several variants of DE in the literature, we will describe the one known as DE/rand/1/bin.

Definition 1: the candidate solutions are the vectors $x_i^t = (x_{i1}^t, ..., x_{in_S}^t)$ that are the position of ith individual, with n_S variables in the generation t.

Definition 2: for each individual i, the mutation operator produces a trial vector u_i^t by combining randomly three chosen individuals i_1, i_2, and i_3 with a given scale factor, called differential weight (β), as follow:

$$u_i^t = x_{i_1} + \beta(x_{i_2} - x_{i_3}),\ i \neq i_1 \neq i_2 \neq i_3,\ \beta \in \mathbb{R}_{>0}. \tag{3.1}$$

Definition 3: for each individual i, the crossover operation generates one offspring, $\widehat{x}_i^t = (\widehat{x}_{i1}^t, ..., \widehat{x}_{in_S}^t)$, that is a randomly discrete combination of the trial's and the parent's vector as follows:

$$\widehat{x}_{ij}^t = \begin{cases} u_{ij}^t, & \text{if } p_j < p_{rc} \\ x_{ij}^t, & \text{otherwise} \end{cases},\ \forall i = \{1, ..., n_M\}, j = \{1, ..., n_S\}, \tag{3.2}$$

where $p_{rc} \in (0, 1)$ is a given real number known as crossover probability, $p_j \sim U(0, 1)$, is a uniformly distributed continuous random variable, n_S is the number of variables and n_M is the number of individuals.

Definition 4: the elitist operation determines which individuals will take part in the next generation based on their fitness as showed in equation (3.3):

$$x_i^{t+1} = \begin{cases} \widehat{x}_i^t, & \text{if } f(\widehat{x}_i^t) < f(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases},\ \forall i = \{1, ..., n_M\}. \tag{3.3}$$

For each iteration, the DE begins by calculating the individuals' fitness. Afterwords, in the mutation phase, trial vectors are created and they are used to produce the offspring in the crossover operation. Then, in the elitist phase, for each comparison between offspring and its parent the fittest one is selected to enter to the next generation.

Some parameters play a especial role in this algorithm. The differential weight (β) determines the amplitude of differential variations ($x_{i_2} - x_{i_3}$). Therefore, the greater the β, the greater will be the time of exploration process increasing the region of exploration around the chosen individuals. On the other hand, smaller values of β can speed up the convergence by restricting the amplitude of the region around the chosen ones to be explored.

The crossover probability controls the exploration-exploitation ability indirectly, by influencing the number trial vector's elements in the offspring. So, by increasing the probability the diversity, which means exploration, tends to also increase due to the tendency of more number of mutating elements in offspring. In the opposite way, the crossover probability builds exploration up and prevents diversification.

The population size is a key factor in exploration ability of DE. The greater the population, the more differential vectors can be combined to make up the next generation, boosting the exploration of DE. But the population is also related to the computational complexity of it, so population and others parameters must be balanced wisely. Algorithm 2 describes the DE.

Algorithm 2 Differential Evolution (DE)

Input: The numbers n_M, n_S, β and p_{rc}. The initial generation $G_0 = \left\{x_1^0, ..., x_{n_M}^0\right\}$.
Output: The best individual, $x_{fittest}$.
Require: A given evaluation function, $f(x)$.

Let $t = 0$.
while *stopping condition isn't true* **do**
  $G_{t+1} = \emptyset$
  for *each individual* $i = 1, ..., n_M$ **do**
    Choose three individuals, i_1, i_2, i_3, such that $i \neq i_1 \neq i_2 \neq i_3$.
    for *each variable* $j = 1, ..., n_S$ **do**
      Determine a random variable, $p_j \sim U(0, 1)$.
      if $p_j < p_{rc}$ **then**
        $\widehat{x}_{ij}^t = x_{i_1 j}^t + \beta(x_{i_2 j}^t - x_{i_3 j}^t)$.
      else
        $\widehat{x}_{ij}^t = x_{ij}^t$.
      end if
    end for
    if $f(x_i^t) > f(\widehat{x}_i^t)$ **then**
      $G_{t+1} = G_{t+1} \cup \{\widehat{x}_i^t\}$.
    else
      $G_{t+1} = G_{t+1} \cup \{x_i^t\}$.
    end if
  end for
  Advance to the next generation, $t = t + 1$.
end while
Determine $x_{fittest}$, where $x_{fittest} \in G_t \mid f(x_{fittest}) = \min\left\{f(x_1^t), ..., f(x_{n_M}^t)\right\}$.

3.3 Swarm Intelligence

Taking inspiration from [21], suppose one is searching for gemstones in a field and he is using a metal detector to measure how far he is from the gems. He can find some stones with this technology, but doing this alone can be difficult. So one calls his friends to take part in the mission using a walkie talkies to communicate to each other. When someone has a strong signal he can call his closest partners to join in this particular area, so the entire group takes advantage of not only individual's information but of its neighborhood. Intuitively, finding stones alone is much more difficult than in a group. This is a simple illustration of the benefits of work in groups. So swarm intelligence is about cooperative work in order to maximize the chance to success.

More precisely, swarm intelligence (SI) is a property of a system whereby individuals, sometimes not intelligence, exhibit collective intelligence behavior. The individuals, also called agents, are social and they interact with themselves and the local environment out of simple rules from which intelligence emerges. By this definition, SI has no centralized control, the intelligence comes out of decentralized agent's behavior [26]. We call this decentralized system that somehow finds its way to some pattern, a self-organizing system. So, owing to it, the swarm behavior converge to an optimal pattern.

Self-organization is a process where a system produces itself some kind of order through the interactions of its parts [23]. So, the emergence of order is bottom-up, which means that perturbations cause the system's parts to interact synergistically in such a way that a new quality on a high level emerges. And once new quality emerged, it influences the behaviors of

the system's parts, a process called downwards causation. The bottom-up process leads the system to a state of equilibrium, some researchers also called this as "order out of chaos" or "order from noise" [22].

Self-organizing systems are observed throughout many scientific fields. In economics, Paul Krugman, Noble Prize in 2008, written "The Self-Organizing Economy" where he states the relation between market and self-organization [24]. In physics, there are many physical processes such as spontaneous magnetization, Bose–Einstein condensation and thermodynamic systems, to mention but a few. In chemistry, there are also many processes, such as molecular self-assembly, autocatalytic networks and self-assembled monolayers. In biology, we could mention homeostasis, swarm behavior, neural network and the creation of structures by social animals. In computer science and applied mathematics, it is used in heuristics. It is interesting to mention that in view of a diverse group of mathematician, physicist and biologists, a recent theory put self-organization together with evolution as one of underlying mechanism in this biological process [25].

So, scientists study how to simulate swarm in machines, creating mathematical models in order to develop heuristics and machine learning algorithms. Particle Swarm Optimization, Ant Colony Optimization, Stochastic Diffusion Search and Bee Colony Optimization are popular successful methods that have been used in real life applications.

3.4 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a population stochastic based search algorithm and bio-inspired of the class of swarm intelligence that does not need the optimization problem be differentiable [26]. It was developed by Kennedy, Eberhart and Shi [27, 28] in the year of 1995 with the intention to simulate social behavior. Afterwards, a substantial aspect of theoretical analysis was published by Bonyadi and Michalewicz [29].

The particles represent solutions and are autonomous agents that do not learn at individual level, but rather at collective one. Thus, they uses a simple communication model to interact to each other and a simple behavior to interact to their local environment. Each agent use its own experience and of its neighborhood to navigate on the search space. The experiential knowledge of an agent is called cognitive component and is proportional to its best position found since the first time step. The experiential knowledge of its neighborhood is called social component and is proportional to the best position found by its neighborhood since the first time step. From now on, agent and particle will be used interchangeably. Below are the descriptions of the elements of PSO with notation taken from [32].

Definition 1: vector $p_i^t = (p_{i1}^t, ..., p_{in_S}^t)$ is the position of ith particle with n_S variables on tth iteration.

Definition 2: vector $p_i^{best,t}$ is the best position found by ith particle up to tth iteration, called personal best position. Each particle finds its personal best position by comparing its position to its previous personal best one. So, we can define $p_i^{best,t}$ mathematically in this way:

$$p_i^{best,t} = \begin{cases} p_i^t, & \text{if } f(p_i^{best,t-1}) > f(p_i^t), \\ p_i^{best,t-1}, & \text{otherwise.} \end{cases} \quad \forall i = \{1, ..., n_M\}. \tag{3.4}$$

Definition 3: vector $p^{global,t}$ is the swarm best position on tth iteration. Let the swarm have n_M particles, mathematically we can define $p^{global,t}$ in this way:

$$p^{global,t} \in \left\{ p_1^{best,t}, ..., p_{n_M}^{best,t} \right\} \mid f(p^{global,t}) = \min \left\{ f(p_1^{best,t}), ..., f(p_{n_M}^{best,t}) \right\}. \tag{3.5}$$

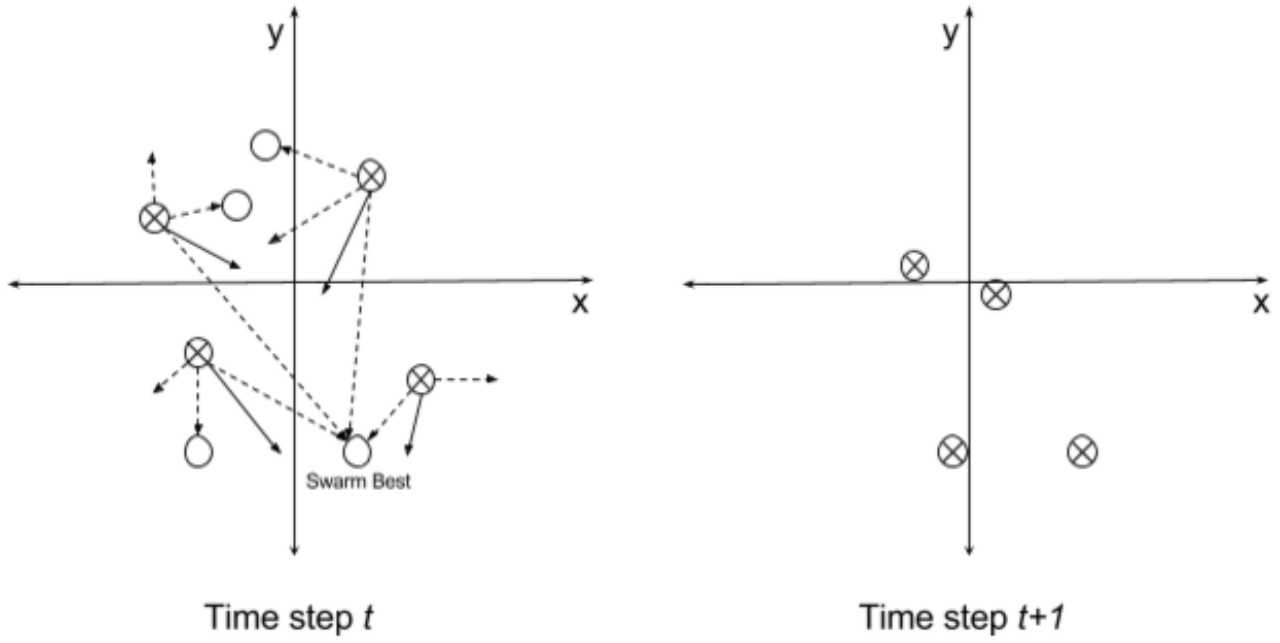


Figure 3.2: Graphical illustration of particles's motion.

Definition 4: the particles navigate on the surface using two simple equations that are the core of PSO. One is the velocity update, whilst the another is the position update.

$$\triangle p_i^{t+1} = w \triangle p_i^t + c_1 r_1 (p_i^{best,t} - p_i^t) + c_2 r_2 (p^{global,t} - p_i^t), \tag{3.6}$$

$$p_i^{t+1} = p_i^t + \triangle p_i^{t+1}. \tag{3.7}$$

Here, $\triangle p_i^t$ is the velocity of particle i on t iteration. The parameters w, c_1 and c_2 are the inertial weight and accelerations of swarm, respectively. The purpose of inertia weight is to incorporate information from previous iterations. The cognitive component, $c_1 r_1 (p_i^{best,t} - p_i^t)$, is the personal experience of the agent. The social component, $c_1 r_1 (p^{global,t} - p_i^t)$, is its social experience and is responsible for the information exchange between all agents of the swarm. The parameters r_1 and r_2 are uniformly distributed random variables between 0 and 1. A graphical illustration of the particle's motion is in Figure 3.2, where the crossed circles are the particle's position (p_i^t), the empty ones are the personal best position ($p_i^{best,t}$), the solid vectors point to the new positions (p_i^{t+1}) by summing the social, cognitive and velocity vectors. We might notice that at time $t+1$ the particles are closer to the best position than at t, this is the objective of PSO, getting closer to it but still moving around others regions.

In the literature, there are two types of PSO, the Global Best PSO, which was described previously, and the Local Best PSO. The difference is related to their neighborhood. The first one, the neighborhood is the entire swarm, whereas the second it is defined by a graph of neighborhood. Since in this project we work only with the global one, we will not describe the other.

The standard algorithm of the Global Best PSO is showed in Algorithm 3.

Algorithm 3 Global Best Particle Swarm Optimization (PSO) [30]

Input: The numbers n_M, n_S, c_1, c_2, the weight w. The particles $P = \{p_1, ..., p_{n_M}\}$.
Output: The swarm best position, p^{global}.
Require: A given evaluation function, $f(x)$.

Let $p_i^{best} = p_i, \forall i = 1, ..., n_M$.
Let $p_i^{global} = \min\left\{f(p_1^{best}), ..., f(p_{n_M}^{best})\right\}$.
while *stopping condition isn't true* **do**
  for *each particle* $i = 1, ..., n_M$ **do**
    if $f(p_i) < f(p_i^{best})$ **then** ▶ Set the particle's best position.
      $p_i^{best} = p_i$.
    end if
    if $f(p_i^{best}) < f(p^{global})$ **then** ▶ Set the global best position.
      $p^{global} = p_i^{best}$.
    end if
  end for
  for *each particle* $i = 1, ..., n_M$ **do** ▶ Update the velocity and position.
    for *each variable* $j = 1, ..., n_S$ **do**
      Pick random numbers: $r_1,\ r_2 \sim U(0, 1)$.
      $\triangle p_{ij} = wp_i + c_1 r_1(p_{ij}^{best} - p_{ij}) + c_2 r_2(p_j^{global} - p_{ij})$.
      $p_{ij} = p_{ij} + \triangle p_{ij}$.
    end for
  end for
end while

3.5 Quasi-Physical Method

The Quasi-Physical Method was developed by Huang and Zhan [33], which is a tool to improve heuristics for packing problems optimization. They formulate the problem using a physical inspired approach, as the name suggests. In this method, the circles are considered elastic solids attracted by a force towards the center of circular container, which is the origin of Cartesian plane, and repelled by a repulsive force whenever they overlap. To make things clearer, some definitions are described above.

Definition 1: By Hooke's Law, the elastic force is linearly depended on the extension displacement d. That is,

$$\overrightarrow{F} = \kappa d, \tag{3.8}$$

where κ is a constant called spring constant.

Definition 2: A circle's repulsive force between circle i and j is proportional to their overlapping region and is defined as:

$$\overrightarrow{F_{ij}} = \begin{cases} \kappa d_{ij}, & \text{if } \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < R_i + R_j, \\ 0, & \text{otherwise,} \end{cases} \tag{3.9}$$

Where $d_{ij} = R_i + R_j - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \forall i, j = 1, ..., n, i \neq j$.

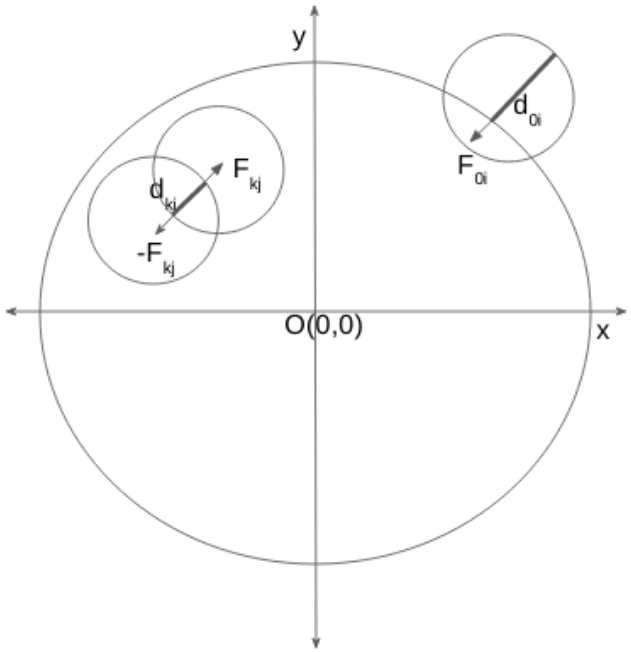


Figure 3.3: An illustrative example of Quasi-Physical Method inspired by [36].

Definition 3: The circular container attractive force is proportional to the circle's region not embedded in the circular container and is defined as:

$$\overrightarrow{F_{0j}} = \begin{cases} -\kappa d_{0j}, & \text{if } \sqrt{x_j^2 + y_j^2} > R_0 - R_j, \\ 0, & \text{otherwise,} \end{cases} \tag{3.10}$$

where $d_{0j} = R_j - R_0 + \sqrt{x_i^2 + y_i^2}, \forall j = 1, ..., n.$

A graphical illustration, Figure 3.3, represents these definitions. The bold lines are d_{ij} and arrows represent forces.

Definition 4: The evaluation function of the system is defined as:

$$U(X) = U(x_1, y_1, ..., x_n, x_n) = \sum_{i=0}^{n} \sum_{j=1, j \neq i}^{n} |F_{i,j,}|, \tag{3.11}$$

Where n is the number of circles and X is called configuration of the system. So, the objective of the optimization turns into minimizing the system's overall forces in order to it becomes U(X) = 0. To it works successfully, the circles must be displaced by the elastic force. Since the Cartesian plane has two dimensional, the force is decomposed into vertical and horizontal ones, therefore, by geometrical properties, it is straightforward to verify that:

$$\overrightarrow{F_{ij}} = F_{ij} \cos(\theta) \overrightarrow{i} + F_{ij} \sin(\theta) \overrightarrow{j}, \tag{3.12}$$

where θ is the angle between the distance from the center of a circle (i) to another one (j) and their horizontal distance, $\cos(\theta) = \frac{(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$ and $\sin(\theta) = \frac{(y_i - y_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$. Since there can be $n - 1$ forces upon a circle i, plus the circular container's attractive force, the new position is the sum of all forces and can be calculated as follows:

$$x_i^{new} = x_i + \sum_{j=0, j\neq i}^{n} F_{ji}\cos(\theta), \tag{3.13}$$

$$y_i^{new} = y_i + \sum_{j=0, j\neq i}^{n} F_{ji}\sin(\theta). \tag{3.14}$$

There are two ways of thinking about how to implement the quasi-physical method. First, all circles move after calculating all forces. Second, each circle moves after calculating all forces applied specifically on it. We named the first one as Static Quasi-Physical Method (Algorithm 4), and the another as Dynamic Quasi-Physical Method (Algorithm 5).

Algorithm 4 Static Quasi-Physical Method (SQP)

Input: The configuration $X = (x_1, y_1, ..., x_n, y_n)$, the radius of n circles and of circular container.

Output: The new configuration $X_{new} = (x_1^{new}, y_1^{new}, ..., x_n^{new}, y_n^{new})$.

1: Let $\triangle x_i = 0$ and $\triangle y_i = 0$, $\forall i = 1, ..., n$.
2: **for** *each circle* $i = 1, ..., n$ **do**
3: **if** $\sqrt{x_i^2 + y_i^2} > R_0 - R_i$ **then**
4: $\triangle x_i = -F_{0i}\cos(\theta)$.
5: $\triangle y_i = -F_{0i}\sin(\theta)$.
6: **end if**
7: **for** *each circle* $j = i + 1, ..., n$ **do**
8: **if** $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < R_i + R_j$ **then**
9: $\triangle x_i = \triangle x_i + F_{ji}\cos(\theta)$.
10: $\triangle y_i = \triangle y_i + F_{ji}\sin(\theta)$.
11: $\triangle x_j = \triangle x_j - F_{ji}\cos(\theta)$.
12: $\triangle y_j = \triangle y_j - F_{ji}\sin(\theta)$.
13: **end if**
14: **end for**
15: **end for**
16: **for** *each circle* $i = 1, ..., n$ **do**
17: $x_i^{new} = x_i + \triangle x_i$.
18: $y_i^{new} = y_i + \triangle y_i$.
19: **end for**

At the line 7 of algorithms 4 and 5, it is observed that when $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = 0$, in other words when two circles overlap each other having the same center, it is impossible to calculate the components of the force and the circles will get stuck together in the same position. Therefore, to deal with it, let α be a computationally "very small" number in the range from 10^{-10} to 10^{-15}, so the new equations are as follow:

$$\cos(\theta) = \frac{(x_i - x_j + \alpha)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + \alpha}, \tag{3.15}$$

$$\sin(\theta) = \frac{(y_i - y_j + \alpha)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + \alpha}. \tag{3.16}$$

The quasi-physical method together with equation (3.11) is used as an evaluation function in heuristics. First we have to apply Algorithm 5 or 4 and then we return the result of equation (3.11) as described in Algorithm 6.

Algorithm 5 Dynamic Quasi-Physical Method (DQP)

Input: The configuration $X = (x_1, y_1, ..., x_n, y_n)$, the radius of n circles and of circular container.

Output: The new configuration $X_{new} = (x_1^{new}, y_1^{new}, ..., x_n^{new}, y_n^{new})$.

Let $\triangle x_i = 0$ and $\triangle y_i = 0$, $\forall i = 1, ..., n$.
for *each circle* $i = 1, ..., n$ **do**
  if $\sqrt{x_i^2 + y_i^2} > R_0 - R_i$ **then**
    $\triangle x_i = -F_{0i} \cos(\theta)$.
    $\triangle y_i = -F_{0i} \sin(\theta)$.
  end if
  for *each circle* $j = 1, ..., n$ **do**
    if $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < R_i + R_j$ **and** $j \neq i$ **then**
      $\triangle x_i = \triangle x_i + F_{ji} \cos(\theta)$.
      $\triangle y_i = \triangle y_i + F_{ji} \sin(\theta)$.
    end if
  end for
  $x_i^{new} = x_i + \triangle x_i$.
  $y_i^{new} = y_i + \triangle y_i$.
end for

Algorithm 6 Quasi-Physical Function

Input: The configuration $X = (x_1, y_1, ..., x_n, y_n)$, the radius of n circles and of circular container.

Output: A function value, S.

Require: An algorithm, SQD or SQP, that moves circles, Move(X).

Let $S = 0$ be a function value.
Get the new configuration, X_{new} = Move(X).
for *each circle* $i = 1, ..., n$ **do**
  if $\sqrt{x_i^2 + y_i^2} > R_0 - R_i$ **then**
    $S = S + |F_{0i} \cos(\theta)| + |F_{0i} \sin(\theta)|$.
  end if
  for *each circle* $j = 1, ..., n$ **do**
    if $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < R_i + R_j$ **and** $j \neq i$ **then**
      $S = S + |F_{ji} \cos(\theta)| + |F_{ji} \sin(\theta)|$.
    end if
  end for
end for

Chapter 4

Proposed Heuristics

This chapter is to set forth in detail the algorithms that we implemented in order to accomplish the objective of this project that is, to develop a bio-inspired heuristic for packing problem with identical unitary radius circles. We built four heuristics, but first we explain the SQDE which is a heuristic that uses DE, Quasi-Physical Function as evaluation function and Static Quasi-Physical Method to move the circles.

Definition 1: Given n circles each of them with center $C(x_i, y_i)$, $\forall i = \{1, ..., n\}$, the individual or particle is a candidate solution, which may no be feasible, that corresponds to a vector $v = (x_1, y_1, ..., x_n, y_n)$.

Definition 2: A candidate solution, v, is feasible if $f(v) < S$. Where $f(\cdot)$ is a Quasi-Physical Function and $S \in \mathbb{R}_{>0}$.

Theoretically, the solution is feasible when its evaluation function value is zero. However, computationally zero is a "very small" number like 10^{-16}. So, we prefer to allow the programmer to decide which zero value he or she whats to use.

The core principle of the heuristic is based on these processes: we fix the container's radius, call the Differential Evolution to try to find a feasible solution with this radius and if DE does not succeed, we increase the circular container's radius, otherwise the heuristic stops. And these processes are repeated again and again. The theory behind this heuristic is simple: by increasing the container's radius, we give the DE opportunity to find a feasible solution more easily.

So, the procedures of the heuristic are explain as follow:

Step 1: Set up the heuristic parameters, including the container's radius R_0.
Step 2: Initialize a population of individual and the number of iteration $i = 0$.
Step 3: Increase $i = i + 1$ and the container's radius, $R_i = R_i + \lambda R_0$.
Step 4: Get the best individual, v_{best}, after calling the DE.
Step 5: If the best individual is not feasible ($f(v_{best}) > S$) and the maximum number of iteration is not reached ($i < W$), go to step 3. Otherwise, go to step 6.
Step 6: R_i is the minimum container's radius found and the heuristic stops.

Where $\lambda \in \mathbb{R}_{>0}$. The maximum number of iteration of the heuristic and the DE is W and $M = \alpha n$, $\alpha \in \mathbb{N}_{>0}$ and n is the number of circles, respectively. In Step 1, the parameters include the DE's parameters, α, λ, n, κ (spring constant), S, W and R_0. In Step 2, we initialize the individuals by choosing their variables randomly between -10^{99} to 10^{99}. Step 3 is self-explanatory. Step 4, we call the DE in order to find a feasible solution to a given problem and

it stops when the maximum iteration of DE is reached or a feasible solution is found. Steps 5 and 6 are self-explanatory.

We also developed three others heuristic. The DQDE which uses DE, Quasi-Physical Function as evaluation function and Dynamic Quasi-Physical Method to move the circles. In the same way, we developed DQPSO and SQPSO.

Since these four approaches fall under the same procedures of SQDE, we do not explain them. Their differences are in input parameters in Step 1.

Chapter 5

Experiments and Discussions

This chapter sets forth the experiments on the four algorithms, described on previous chapter, to validate the hypothesis. The experiments were conduct on an Intel Core i3-3217U 1.80Ghz x 4 processor with 8 gigabyte of RAM memory running on a GNU/Linux operating system. The algorithms were fully written in C programming language standard (C11) ISO/IEC 9899 and compiled by GNU/GCC 6.2 without optimization directive. After many experiments the algorithms parameters were defined as described in Table 5.1.

Table 5.1: Algorithms Parameters

	SQPSO	**DQPSO**	**SQDE**	**DQDE**
R_0	$1.20\sqrt{n}$	$1.05\sqrt{n}$	$1.15\sqrt{n}$	$1.15\sqrt{n}$
n_M	50	50	5	5
S	10^{-15}	10^{-15}	10^{-15}	10^{-15}
W	30	30	30	30
α	80	80	20	20
λ	5%	5%	1%	1%
w	0.3	0.3	_____	_____
β	_____	_____	0.9	0.9
p_{rc}	_____	_____	0.99	0.99
c_1	2.0	2.0	_____	_____
c_2	2.0	2.0	_____	_____
κ	0.5	0.5	0.5	0.5

Experiments show that when the parameter R_0 in SQPSO and DQPSO are equal to SQDE and DQDE, the quality of their solutions do not get better and the time taken to solve the instances decrease. Also, if n_M in SQPSO and DQPSO were equal to SQDE and DQDE, empirically we can show that their solutions would drastically decrease. For the same reason we stated previously, α and λ ended up being different. Therefore, we set R_0, n_M, α and λ in order to the heuristics be as much efficient and efficacy as possible according to our experimental results.

In order to lay out these experiments, we created a kind of table made up of four columns, as shows the tables 5.2 to 5.9. The first column is the number of circles, the instances, carried out in each experiment. The second and third ones each relates to an algorithm and are subdivided into three columns that correspond to the average and standard deviation of the solution found of thirty tests in every instances, their number of trials and time taken in seconds. The two last column shows the time and radius relation each, they are comparative measure between these two algorithms for each instance. The column that shows the sum of violation of circle's boundaries due to overlapping regions, was not created because in all instances of the four algorithms the overlapping regions were zero.

The first experiment compares SQPSO and DQPSO. As set out in Tables 5.2 and 5.3 by analyzing the columns $\frac{Radius\,1}{Radius\,2}$ and $\frac{Time\,1}{Time\,2}$, we can conclude that SQPSO surpasses DQPSO in terms of quality and time taken, except for the first instance.

By using the same analysis in Tables 5.4 and 5.5, we can conclude that DQDE outperforms SQDE from first to twenty-fifth instances, but at the beginning of twenty-sixth instance DQDE's solution and its time starts to deteriorate until it reaches a point that SQDE becomes the best option from thirty to fifty circles. Owing to this deterioration and knowing that we use heuristics when there are "big" number of variables, we prefer SQDE over DQDE to be compared to SQPSO.

In Tables 5.6 and 5.7 we compare SQDE and SQPSO, which had the best results so far. We can easily see that SQDE attains better quality of solutions than SQPSO in almost every case, and outshines it in terms of time taken. Therefore, we chose SQDE to be compared to the exact method.

The algorithm in [1] developed by Ernesto G. Birgin and Jan M. Gentil that I call Exact Method, utilizes ALGENCAN, an algorithms that solves constrained optimization problems by calculating an approximation of Augmented Lagrangian Hessian matrix combined with second order derivatives and other approaches. Besides, this algorithm tries to solve packing problems using a series of procedures like temporarily removing all loose circles, developing a nonlinear system of equations and using Newton's Methods.

In Tables 5.8 and 5.9, we can compare SQDE with the Exact Method. Unfortunately due to lack of time, the numerical results of Exact Method was taken on article [1] instead of being run in the same computer environment of others algorithms. As described on this article, the computer had a 2.4GHz Intel Core 2 Quad with 4 gigabyte of RAM, GNU/Linux operating system, the code was written fully FORTRAN 77 and compiled by the G77 Fortran compiler of GCC with the -O3 optimization directive enabled.

With regard to the quality of solution, we can conclude that the averages of the DE's solutions were at most 5% worse and their time taken outshine what was expected by being at least 97% faster than the Exact Method between the twentieth and fiftieth instances.

Therefore, the algorithm SQDE validated the hypothesis and its results are depicted in Appendix A for all tested instances.

Table 5.2: DQPSO x SQPSO, part I

	Dynamic Quasi-Physical PSO (DQPSO)			Static Quasi-Physical PSO (SQPSO)				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\,1}{Radius\,2}$	$\frac{Time\,1}{Time\,2}$
1	1.050000±0.000000	1.000000±0.000000	0.00±0.00	1.200000±0.000000	4.000000±0.000000	0.00±0.00	0.88	1.00
2	2.069465±0.079442	9.266666±1.123486	0.02±0.00	2.050609±0.000000	9.000000±0.000000	0.02±0.00	1.01	1.00
3	2.211251±0.096781	5.533333±1.117536	0.03±0.00	2.165063±0.000000	5.000000±0.000000	0.03±0.00	1.02	1.00
4	2.540000±0.048989	5.400000±0.489897	0.07±0.00	2.500000±0.000000	5.000000±0.000000	0.07±0.00	1.02	1.00
5	2.795084±0.000000	5.000000±0.000000	0.12±0.00	2.795084±0.000000	5.000000±0.000000	0.12±0.00	1.00	1.00
6	3.102687±0.096609	5.333333±0.788810	0.20±0.03	3.061862±0.000000	5.000000±0.000000	0.19±0.00	1.01	1.05
7	3.064661±0.059976	3.166666±0.453382	0.16±0.03	3.042614±0.000000	3.000000±0.000000	0.14±0.00	1.01	1.14
8	3.403540±0.035276	4.066666±0.249443	0.30±0.02	3.394112±0.000000	4.000000±0.000000	0.28±0.00	1.00	1.07
9	3.759999±0.037416	5.066666±0.249443	0.55±0.03	3.750000±0.000000	5.000000±0.000000	0.50±0.00	1.00	1.10
10	3.973928±0.053748	5.133333±0.339934	0.73±0.05	3.952847±0.000000	5.000000±0.000000	0.64±0.00	1.01	1.14
11	4.051809±0.102075	4.433333±0.615539	0.82±0.12	4.024171±0.073333	4.266666±0.442216	0.70±0.08	1.01	1.17
12	4.185789±0.078528	4.166666±0.453382	0.93±0.12	4.156921±0.000000	4.000000±0.000000	0.78±0.00	1.01	1.19
13	4.356707±0.067185	4.166666±0.372677	1.24±0.14	4.326661±0.000000	4.000000±0.000000	0.96±0.00	1.01	1.29
14	4.514933±0.105092	4.133333±0.561743	1.42±0.23	4.489988±0.000000	4.000000±0.000000	1.15±0.00	1.01	1.23
15	4.725039±0.107238	4.400000±0.553774	1.86±0.28	4.647580±0.000000	4.000000±0.000000	1.41±0.07	1.02	1.32
16	4.893333±0.161107	4.466666±0.805536	2.21±0.49	4.800000±0.000000	4.000000±0.000000	1.67±0.07	1.02	1.32
17	4.988957±0.098149	4.200000±0.476095	2.47±0.37	4.947726±0.000000	4.000000±0.000000	1.93±0.01	1.01	1.28
18	5.126524±0.096176	4.166666±0.453382	2.85±0.37	4.970960±0.105118	3.433333±0.495535	1.95±0.29	1.03	1.46
19	5.128971±0.134742	3.533333±0.618241	2.68±0.66	5.049057±0.081223	3.166666±0.372677	1.91±0.30	1.02	1.40
20	5.456005±0.220605	4.400000±0.986576	4.10±1.14	5.336748±0.076011	3.866666±0.339934	2.91±0.09	1.02	1.41
21	5.529641±0.153514	4.133333±0.669991	4.28±0.80	5.308150±0.085391	3.166666±0.372677	2.56±0.35	1.04	1.67
22	5.761394±0.223991	4.566666±0.955103	5.76±1.36	5.628498±0.000000	4.000000±0.000000	3.86±0.11	1.02	1.49
23	5.890879±0.160060	4.566666±0.667499	6.49±1.21	5.762990±0.043043	4.033333±0.179505	4.39±0.23	1.02	1.48
24	6.009414±0.175878	4.533333±0.718021	7.15±1.34	5.878775±0.000000	4.000000±0.000000	4.88±0.04	1.02	1.47
25	6.150000±0.152752	4.600000±0.611010	8.01±1.19	6.000000±0.000000	4.000000±0.000000	5.44±0.03	1.03	1.47

Table 5.3: DQPSO x SQPSO, part II

	Dynamic Quasi-Physical PSO (DQPSO)			Static Quasi-Physical PSO (SQPSO)				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\,1}{Radius\,2}$	$\frac{Time\,1}{Time\,2}$
26	6.229302±0.126337	4.433333±0.495535	8.75±1.23	6.101826±0.063595	3.933333±0.249443	5.98±0.25	1.02	1.46
27	6.356626±0.174068	4.466666±0.669991	10.0±1.96	6.174761±0.109886	3.766666±0.422952	6.25±0.77	1.03	1.60
28	6.464452±0.162856	4.433333±0.615539	11.1±1.93	6.323345±0.079372	3.900000±0.300000	7.17±0.61	1.02	1.55
29	6.632727±0.214093	4.633333±0.795124	12.5±2.69	6.399370±0.113883	3.766666±0.422952	7.71±0.77	1.04	1.62
30	6.746116±0.259647	4.633333±0.948097	14.4±3.52	6.481383±0.129099	3.666666±0.471404	8.11±1.22	1.04	1.78
31	6.922587±0.235490	4.866666±0.845905	16.5±3.25	6.653478±0.083516	3.900000±0.300000	9.56±0.84	1.04	1.73
32	7.071067±0.263312	5.000000±0.930949	18.2±3.86	6.750512±0.096148	3.866666±0.339934	10.4±0.84	1.05	1.75
33	7.056237±0.218536	4.566666±0.760847	17.7±3.03	6.836029±0.114891	3.800000±0.400000	11.0±1.36	1.03	1.61
34	7.104043±0.219253	4.366666±0.752034	18.4±3.74	6.967987±0.087464	3.900000±0.300000	12.6±1.08	1.02	1.46
35	7.227477±0.237668	4.433333±0.803464	20.4±4.68	7.020414±0.130809	3.733333±0.442216	13.3±1.46	1.03	1.53
36	7.399999±0.248997	4.666666±0.829993	24.1±4.59	7.050000±0.150000	3.500000±0.500000	13.4±2.04	1.05	1.80
37	7.532487±0.268416	4.766666±0.882546	26.1±5.26	7.157383±0.151730	3.533333±0.498887	13.9±2.43	1.05	1.88
38	7.571955±0.220592	4.566666±0.715697	27.0±5.25	7.315104±0.136300	3.733333±0.442216	16.2±1.72	1.04	1.67
39	7.837472±0.315871	5.100000±1.011599	32.6±7.25	7.389914±0.147196	3.666666±0.471404	16.8±2.39	1.06	1.94
40	7.926776±0.335989	5.066666±1.062491	34.4±8.16	7.547302±0.107496	3.866666±0.339934	19.3±1.60	1.05	1.78
41	7.982561±0.247071	4.933333±0.771722	35.7±6.66	7.662405±0.079860	3.933333±0.249443	20.9±0.99	1.04	1.71
42	8.057720±0.261023	4.866666±0.805536	37.9±7.30	7.755286±0.080829	3.933333±0.249443	22.4±1.15	1.04	1.69
43	8.229585±0.229510	5.100000±0.700000	43.1±6.93	7.847068±0.081785	3.933333±0.249443	23.8±1.52	1.05	1.81
44	8.258395±0.215793	4.900000±0.650640	43.8±6.13	7.937788±0.082731	3.933333±0.249443	25.7±1.65	1.04	1.70
45	8.385254±0.193649	5.000000±0.577350	51.3±6.22	8.027484±0.083666	3.933333±0.249443	27.2±1.32	1.04	1.89
46	8.455304±0.230554	4.933333±0.679869	49.7±7.79	8.138795±0.000000	4.000000±0.000000	29.3±0.80	1.04	1.70
47	8.580994±0.286564	5.033333±0.835995	54.8±10.2	8.181081±0.116523	3.866666±0.339934	30.3±2.54	1.05	1.81
48	8.729536±0.243310	5.200000±0.702376	59.7±9.26	8.302296±0.062182	3.966666±0.179505	32.8±2.50	1.05	1.82
49	8.866666±0.317367	5.333333±0.906764	66.2±12.5	8.400000±0.090369	4.000000±0.258198	34.8±2.85	1.06	1.90
50	8.815264±0.257120	4.933333±0.727247	64.1±10.1	8.461711±0.126929	3.933333±0.359010	36.6±3.63	1.04	1.75

Table 5.4: DQDE x SQDE, part I

	Dynamic Quasi-Physical DE (DQDE)			Static Quasi-Physical DE (SQDE)				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\,1}{Radius\,2}$	$\frac{Time\,1}{Time\,2}$
1	1.190000±0.000000	5.000000±0.000000	0.00±0.00	1.190000±0.000000	5.000000±0.000000	0.00±0.00	1.00	1.00
2	1.767766±0.000000	11.00000±0.000000	0.00±0.00	2.008654±0.002538	28.03333±0.179505	0.00±0.00	0.88	1.00
3	2.165063±0.000000	11.00000±0.000000	0.00±0.00	2.165063±0.000000	11.00000±0.000000	0.00±0.00	1.00	1.00
4	2.420000±0.000000	7.000000±0.000000	0.01±0.00	2.427333±0.012092	7.366666±0.604611	0.00±0.00	1.00	10.00
5	2.711605±0.022310	7.266666±0.997775	0.01±0.00	2.802538±0.076011	11.33333±3.399346	0.02±0.00	0.97	0.50
6	3.012872±0.000000	9.000000±0.000000	0.04±0.00	3.025119±0.026645	9.500000±1.087811	0.03±0.00	1.00	1.33
7	3.045259±0.007937	1.100000±0.300000	0.00±0.00	3.065543±0.021312	1.866666±0.805536	0.00±0.00	0.99	0.00
8	3.309259±0.000000	3.000000±0.000000	0.02±0.00	3.353571±0.031566	4.566666±1.116044	0.02±0.00	0.99	1.00
9	3.649999±0.014142	7.666666±0.471404	0.09±0.00	3.675999±0.036932	8.533333±1.231079	0.06±0.00	0.99	1.50
10	3.826355±0.000000	7.000000±0.000000	0.10±0.00	3.928602±0.074691	10.23333±2.361967	0.11±0.02	0.97	0.91
11	3.946783±0.000000	5.000000±0.000000	0.09±0.00	4.002060±0.077229	6.666666±2.328566	0.09±0.03	0.99	1.00
12	4.052998±0.000000	3.000000±0.000000	0.06±0.00	4.170778±0.124128	6.400000±3.583294	0.10±0.06	0.97	0.06
13	4.254550±0.000000	4.000000±0.000000	0.11±0.00	4.327863±0.055454	6.033333±1.538036	0.12±0.03	0.98	0.92
14	4.340322±0.000000	2.000000±0.000000	0.04±0.00	4.461302±0.109165	5.233333±2.917571	0.12±0.08	0.97	0.33
15	4.548173±0.019192	3.433333±0.495535	0.14±0.01	4.634670±0.049328	5.666666±1.273664	0.16±0.03	0.98	0.88
16	4.688000±0.071105	3.200000±1.777638	0.15±0.11	4.745333±0.118876	4.633333±2.971905	0.15±0.11	0.99	1.00
17	4.835028±0.018233	3.266666±0.442216	0.18±0.04	4.916116±0.077270	5.233333±1.874092	0.22±0.08	0.98	0.82
18	4.884693±0.014422	1.133333±0.339934	0.05±0.01	4.986517±0.103189	3.533333±2.432191	0.15±0.13	0.98	0.33
19	5.035981±0.081728	1.533333±1.874981	0.10±0.18	5.166748±0.104209	4.533333±2.390722	0.24±0.15	0.97	0.42
20	5.157863±0.021081	1.333333±0.471404	0.09±0.03	5.278611±0.086010	4.033333±1.923249	0.26±0.13	0.98	0.35
21	5.298985±0.046558	1.633333±1.015983	0.15±0.11	5.445627±0.123578	4.833333±2.696705	0.34±0.22	0.97	0.44
22	5.542507±0.109721	4.166666±2.339278	0.52±0.38	5.581594±0.079414	5.000000±1.693123	0.43±0.16	0.99	1.21
23	5.646292±0.139510	3.733333±2.908989	0.53±0.53	5.732617±0.138849	5.533333±2.895206	0.55±0.32	0.98	0.96
24	5.754667±0.129450	3.466666±2.642389	0.55±0.54	5.837950±0.067032	5.166666±1.368291	0.57±0.17	0.99	0.96
25	5.866666±0.151840	3.333333±3.036811	0.58±0.71	5.918333±0.111417	4.366666±2.228352	0.51±0.32	0.99	1.14

Table 5.5: DQDE x SQDE, part II

	Dynamic Quasi-Physical DE (DQDE)			Static Quasi-Physical DE (SQDE)				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\,1}{Radius\,2}$	$\frac{Time\,1}{Time\,2}$
26	5.952255±0.119896	2.733333±2.351358	0.52±0.61	5.998146±0.072689	3.633333±1.425560	0.46±0.22	0.99	1.13
27	6.060445±0.149689	2.633333±2.880779	0.59±0.81	6.124531±0.133311	3.866666±2.565584	0.58±0.43	0.99	1.02
28	6.184002±0.172134	2.866666±3.253032	0.75±1.03	6.259847±0.120005	4.300000±2.267891	0.73±0.46	0.99	1.03
29	6.250381±0.131860	2.066666±2.448582	0.53±0.85	6.327568±0.093014	3.500000±1.727232	0.66±0.37	0.99	0.80
30	6.468603±0.198166	4.100000±3.618010	1.33±1.39	6.514246±0.137064	4.933333±2.502443	1.01±0.56	0.99	1.32
31	6.560682±0.221819	3.833333±3.983995	1.38±1.69	6.607080±0.140364	4.666666±2.521022	1.09±0.68	0.99	1.27
32	6.614748±0.174620	2.933333±3.086889	1.11±1.43	6.697715±0.117393	4.400000±2.075250	1.10±0.60	0.99	1.01
33	6.816881±0.225595	4.666666±3.927113	2.09±2.01	6.839859±0.188163	5.066666±3.275498	1.36±1.03	1.00	1.54
34	6.855255±0.211143	3.566666±3.621080	1.68±1.99	6.884410±0.141179	4.066666±2.421202	1.20±0.90	1.00	1.40
35	7.044078±0.257392	5.066666±4.350734	2.75±2.68	6.988862±0.165038	4.133333±2.789663	1.26±1.04	1.01	2.18
36	7.164000±0.253897	5.400000±4.231626	3.16±2.87	7.022000±0.147363	3.033333±2.456058	0.96±0.93	1.02	3.29
37	7.323646±0.271847	6.400000±4.469153	4.21±3.19	7.199963±0.132787	4.366666±2.183015	1.60±0.95	1.02	2.63
38	7.399351±0.280004	6.033333±4.542270	4.27±3.61	7.218528±0.086228	3.100000±1.398809	1.16±0.71	1.03	3.68
39	7.525222±0.283786	6.500000±4.544227	4.95±3.78	7.352444±0.129933	3.733333±2.080598	1.48±1.03	1.02	3.34
40	7.637954±0.279849	6.766666±4.424803	5.60±4.05	7.503030±0.135137	4.633333±2.136716	2.18±1.14	1.02	2.57
41	7.784064±0.265392	7.566666±4.144742	6.77±4.17	7.498058±0.078160	3.100000±1.220655	1.50±0.74	1.04	4.51
42	7.921625±0.268088	8.233333±4.136692	8.09±4.43	7.632152±0.138677	3.766666±2.139833	1.94±1.34	1.04	4.17
43	7.934500±0.303347	7.000000±4.626013	7.16±5.28	7.739963±0.121484	4.033333±1.852625	2.18±1.23	1.03	3.28
44	8.103619±0.268671	8.166666±4.050377	8.99±4.83	7.771957±0.101445	3.166666±1.529342	1.81±1.09	1.04	4.97
45	8.099038±0.316512	6.733333±4.718285	7.86±6.03	7.853070±0.113490	3.066666±1.691810	1.76±1.23	1.03	4.47
46	8.328701±0.274328	8.800000±4.044749	11.2±5.62	7.948890±0.121830	3.200000±1.796292	2.04±1.43	1.05	5.49
47	8.379895±0.285249	8.233333±4.160795	11.0±6.11	8.057679±0.117060	3.533333±1.707499	2.43±1.44	1.04	4.53
48	8.498595±0.273252	8.666666±3.944053	12.4±6.14	8.202992±0.138101	4.400000±1.993322	3.34±1.85	1.04	3.71
49	8.512000±0.303835	7.600000±4.340506	11.4±7.19	8.206333±0.138455	3.233333±1.977933	2.42±1.90	1.04	4.71
50	8.711555±0.255212	9.199999±3.609247	15.6±6.54	8.259007±0.113137	2.800000±1.600000	2.19±1.55	1.05	7.12

Table 5.6: SQDE x SQPSO, part I

	Static Quasi-Physical Differential Evolution (SQDE)			Static Quasi-Physical PSO (SQPSO)				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\,1}{Radius\,2}$	$\frac{Time\,1}{Time\,2}$
1	1.190000±0.000000	5.000000±0.000000	0.00±0.00	1.200000±0.000000	4.000000±0.000000	0.00±0.00	0.99	1.00
2	2.008654±0.002538	28.03333±0.179505	0.00±0.00	2.050609±0.000000	9.000000±0.000000	0.02±0.00	0.98	0.00
3	2.165063±0.000000	11.00000±0.000000	0.00±0.00	2.165063±0.000000	5.000000±0.000000	0.03±0.00	1.00	0.00
4	2.427333±0.012092	7.366666±0.604611	0.00±0.00	2.500000±0.000000	5.000000±0.000000	0.07±0.00	0.97	0.00
5	2.802538±0.076011	11.33333±3.399346	0.02±0.00	2.795084±0.000000	5.000000±0.000000	0.12±0.00	1.00	0.17
6	3.025119±0.026645	9.500000±1.087811	0.03±0.00	3.061862±0.000000	5.000000±0.000000	0.19±0.00	0.99	0.16
7	3.065543±0.021312	1.866666±0.805536	0.00±0.00	3.042614±0.000000	3.000000±0.000000	0.14±0.00	1.01	0.00
8	3.353571±0.031566	4.566666±1.116044	0.02±0.00	3.394112±0.000000	4.000000±0.000000	0.28±0.00	0.99	0.07
9	3.675999±0.036932	8.533333±1.231079	0.06±0.00	3.750000±0.000000	5.000000±0.000000	0.50±0.00	0.98	0.12
10	3.928602±0.074691	10.23333±2.361967	0.11±0.02	3.952847±0.000000	5.000000±0.000000	0.64±0.00	0.99	0.17
11	4.002060±0.077229	6.666666±2.328566	0.09±0.03	4.024171±0.073333	4.266666±0.442216	0.70±0.08	0.99	0.13
12	4.170778±0.124128	6.400000±3.583294	0.10±0.06	4.156921±0.000000	4.000000±0.000000	0.78±0.00	1.00	0.13
13	4.327863±0.055454	6.033333±1.538036	0.12±0.03	4.326661±0.000000	4.000000±0.000000	0.96±0.00	1.00	0.13
14	4.461302±0.109165	5.233333±2.917571	0.12±0.08	4.489988±0.000000	4.000000±0.000000	1.15±0.00	0.99	0.10
15	4.634670±0.049328	5.666666±1.273664	0.16±0.03	4.647580±0.000000	4.000000±0.000000	1.41±0.07	1.00	0.11
16	4.745333±0.118876	4.633333±2.971905	0.15±0.11	4.800000±0.000000	4.000000±0.000000	1.67±0.07	0.99	0.09
17	4.916116±0.077270	5.233333±1.874092	0.22±0.08	4.947726±0.000000	4.000000±0.000000	1.93±0.01	0.99	0.11
18	4.986517±0.103189	3.533333±2.432191	0.15±0.13	4.970960±0.105118	3.433333±0.495535	1.95±0.29	1.00	0.08
19	5.166748±0.104209	4.533333±2.390722	0.24±0.15	5.049057±0.081223	3.166666±0.372677	1.91±0.30	1.02	0.13
20	5.278611±0.086010	4.033333±1.923249	0.26±0.13	5.336748±0.076011	3.866666±0.339934	2.91±0.09	0.99	0.09
21	5.445627±0.123578	4.833333±2.696705	0.34±0.22	5.308150±0.085391	3.166666±0.372677	2.56±0.35	1.03	0.13
22	5.581594±0.079414	5.000000±1.693123	0.43±0.16	5.628498±0.000000	4.000000±0.000000	3.86±0.11	0.99	0.11
23	5.732617±0.138849	5.533333±2.895206	0.55±0.32	5.762990±0.043043	4.033333±0.179505	4.39±0.23	0.99	0.13
24	5.837950±0.067032	5.166666±1.368291	0.57±0.17	5.878775±0.000000	4.000000±0.000000	4.88±0.04	0.99	0.12
25	5.918333±0.111417	4.366666±2.228352	0.51±0.32	6.000000±0.000000	4.000000±0.000000	5.44±0.03	0.99	0.09

Table 5.7: SQDE x SQPSO, part II

	Static Quasi-Physical Differential Evolution (SQDE)			Static Quasi-Physical PSO (SQPSO)				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\,1}{Radius\,2}$	$\frac{Time\,1}{Time\,2}$
26	5.998146±0.072689	3.633333±1.425560	0.46±0.22	6.101826±0.063595	3.933333±0.249443	5.98±0.25	0.98	0.08
27	6.124531±0.133311	3.866666±2.565584	0.58±0.43	6.174761±0.109886	3.766666±0.422952	6.25±0.77	0.99	0.09
28	6.259847±0.120005	4.300000±2.267891	0.73±0.46	6.323345±0.079372	3.900000±0.300000	7.17±0.61	0.99	0.10
29	6.327568±0.093014	3.500000±1.727232	0.66±0.37	6.399370±0.113883	3.766666±0.422952	7.71±0.77	0.99	0.09
30	6.514246±0.137064	4.933333±2.502443	1.01±0.56	6.481383±0.129099	3.666666±0.471404	8.11±1.22	1.01	0.12
31	6.607080±0.140364	4.666666±2.521022	1.09±0.68	6.653478±0.083516	3.900000±0.300000	9.56±0.84	0.99	0.11
32	6.697715±0.117393	4.400000±2.075250	1.10±0.60	6.750512±0.096148	3.866666±0.339934	10.4±0.84	0.99	0.11
33	6.839859±0.188163	5.066666±3.275498	1.36±1.03	6.836029±0.114891	3.800000±0.400000	11.0±1.36	1.00	0.12
34	6.884410±0.141179	4.066666±2.421202	1.20±0.90	6.967987±0.087464	3.900000±0.300000	12.6±1.08	0.99	0.10
35	6.988862±0.165038	4.133333±2.789663	1.26±1.04	7.020414±0.130809	3.733333±0.442216	13.3±1.46	1.00	0.09
36	7.022000±0.147363	3.033333±2.456058	0.96±0.93	7.050000±0.150000	3.500000±0.500000	13.4±2.04	1.00	0.07
37	7.199963±0.132787	4.366666±2.183015	1.60±0.95	7.157383±0.151730	3.533333±0.498887	13.9±2.43	1.01	0.12
38	7.218528±0.086228	3.100000±1.398809	1.16±0.71	7.315104±0.136300	3.733333±0.442216	16.2±1.72	0.99	0.07
39	7.352444±0.129933	3.733333±2.080598	1.48±1.03	7.389914±0.147196	3.666666±0.471404	16.8±2.39	0.99	0.09
40	7.503030±0.135137	4.633333±2.136716	2.18±1.14	7.547302±0.107496	3.866666±0.339934	19.3±1.60	0.99	0.11
41	7.498058±0.078160	3.100000±1.220655	1.50±0.74	7.662405±0.079860	3.933333±0.249443	20.9±0.99	0.98	0.07
42	7.632152±0.138677	3.766666±2.139833	1.94±1.34	7.755286±0.080829	3.933333±0.249443	22.4±1.15	0.98	0.09
43	7.739963±0.121484	4.033333±1.852625	2.18±1.23	7.847068±0.081785	3.933333±0.249443	23.8±1.52	0.99	0.09
44	7.771957±0.101445	3.166666±1.529342	1.81±1.09	7.937788±0.082731	3.933333±0.249443	25.7±1.65	0.98	0.07
45	7.853070±0.113490	3.066666±1.691810	1.76±1.23	8.027484±0.083666	3.933333±0.249443	27.2±1.32	0.98	0.06
46	7.948890±0.121830	3.200000±1.796292	2.04±1.43	8.138795±0.000000	4.000000±0.000000	29.3±0.80	0.98	0.07
47	8.057679±0.117060	3.533333±1.707499	2.43±1.44	8.181081±0.116523	3.866666±0.339934	30.3±2.54	0.98	0.08
48	8.202992±0.138101	4.400000±1.993322	3.34±1.85	8.302296±0.062182	3.966666±0.179505	32.8±2.50	0.99	0.10
49	8.206333±0.138455	3.233333±1.977933	2.42±1.90	8.400000±0.090369	4.000000±0.258198	34.8±2.85	0.98	0.07
50	8.259007±0.113137	2.800000±1.600000	2.19±1.55	8.461711±0.126929	3.933333±0.359010	36.6±3.63	0.98	0.06

Table 5.8: SQDE x Exact Method, part I

	Static Quasi-Physical Differential Evolution (SQDE)			Exact Approach				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\ 1}{Radius\ 2}$	$\frac{Time\ 1}{Time\ 2}$
1	1.190000±0.000000	5.000000±0.000000	0.00±0.00	1.0000	00001	0.0000	1.19	1.00
2	2.008654±0.002538	28.03333±0.179505	0.00±0.00	2.0000	00017	0.1100	1.00	0.00
3	2.165063±0.000000	11.00000±0.000000	0.00±0.00	2.1547	00001	0.0000	1.00	0.00
4	2.427333±0.012092	7.366666±0.604611	0.00±0.00	2.4142	00001	0.0000	1.01	1.00
5	2.802538±0.076011	11.33333±3.399346	0.02±0.00	2.7013	00013	0.0900	1.04	0.22
6	3.025119±0.026645	9.500000±1.087811	0.03±0.00	3.0000	00002	0.1000	1.01	0.30
7	3.065543±0.021312	1.866666±0.805536	0.00±0.00	3.0000	00001	0.0000	1.02	0.00
8	3.353571±0.031566	4.566666±1.116044	0.02±0.00	3.3047	00001	0.0500	1.01	0.40
9	3.675999±0.036932	8.533333±1.231079	0.06±0.00	3.6131	28611	271.8200	1.02	0.00
10	3.928602±0.074691	10.23333±2.361967	0.11±0.02	3.8130	00001	0.1300	1.03	0.85
11	4.002060±0.077229	6.666666±2.328566	0.09±0.03	3.9238	00039	29.9000	1.02	0.00
12	4.170778±0.124128	6.400000±3.583294	0.10±0.06	4.0296	00014	16.4300	1.04	0.01
13	4.327863±0.055454	6.033333±1.538036	0.12±0.03	4.2360	00001	0.9000	1.02	0.13
14	4.461302±0.109165	5.233333±2.917571	0.12±0.08	4.3284	00004	9.6700	1.03	0.01
15	4.634670±0.049328	5.666666±1.273664	0.16±0.03	4.5213	00023	28.7900	1.03	0.01
16	4.745333±0.118876	4.633333±2.971905	0.15±0.11	4.6154	00050	72.640	1.03	0.00
17	4.916116±0.077270	5.233333±1.874092	0.22±0.08	4.7920	00001	6.4200	1.03	0.03
18	4.986517±0.103189	3.533333±2.432191	0.15±0.13	4.8637	00001	9.6200	1.03	0.02
19	5.166748±0.104209	4.533333±2.390722	0.24±0.15	4.8637	00001	12.200	1.06	0.02
20	5.278611±0.086010	4.033333±1.923249	0.26±0.13	5.1223	00001	8.9400	1.03	0.03
21	5.445627±0.123578	4.833333±2.696705	0.34±0.22	5.2523	00005	36.8900	1.04	0.01
22	5.581594±0.079414	5.000000±1.693123	0.43±0.16	5.4397	00057	86.9400	1.03	0.00
23	5.732617±0.138849	5.533333±2.895206	0.55±0.32	5.5452	00110	147.6900	1.03	0.00
24	5.837950±0.067032	5.166666±1.368291	0.57±0.17	5.6516	00002	29.9500	1.03	0.02
25	5.918333±0.111417	4.366666±2.228352	0.51±0.32	5.7528	00201	464.8500	1.03	0.00

Table 5.9: SQDE x Exact Method, part II

	Static Quasi-Physical Differential Evolution (SQDE)			Exact Approach				
Circles	**Radius 1**	**Updates 1**	**Time 1**	**Radius 2**	**Updates 2**	**Time 2**	$\frac{Radius\ 1}{Radius\ 2}$	$\frac{Time\ 1}{Time\ 2}$
26	5.998146±0.072689	3.633333±1.425560	0.46±0.22	5.8281	00002	44.5200	1.03	0.01
27	6.124531±0.133311	3.866666±2.565584	0.58±0.43	5.9063	00269	2535.8000	1.04	0.00
28	6.259847±0.120005	4.300000±2.267891	0.73±0.46	6.0149	04386	14142.0000	1.04	0.00
29	6.327568±0.093014	3.500000±1.727232	0.66±0.37	6.1385	00015	100.0100	1.03	0.01
30	6.514246±0.137064	4.933333±2.502443	1.01±0.56	6.1977	00009	267.6600	1.05	0.00
31	6.607080±0.140364	4.666666±2.521022	1.09±0.68	6.2915	00001	71.1400	1.05	0.02
32	6.697715±0.117393	4.400000±2.075250	1.10±0.60	6.4294	00502	790.4700	1.04	0.00
33	6.839859±0.188163	5.066666±3.275498	1.36±1.03	6.4867	04231	14955.0000	1.05	0.00
34	6.884410±0.141179	4.066666±2.421202	1.20±0.90	6.6109	04514	13744.0000	1.04	0.00
35	6.988862±0.165038	4.133333±2.789663	1.26±1.04	6.6971	05801	3260.9000	1.04	0.00
36	7.022000±0.147363	3.033333±2.456058	0.96±0.93	6.7467	00005	259.4800	1.04	0.00
37	7.199963±0.132787	4.366666±2.183015	1.60±0.95	6.7587	00029	1092.9000	1.07	0.00
38	7.218528±0.086228	3.100000±1.398809	1.16±0.71	6.9618	00042	641.6400	1.04	0.00
39	7.352444±0.129933	3.733333±2.080598	1.48±1.03	7.0578	00007	245.7900	1.04	0.01
40	7.503030±0.135137	4.633333±2.136716	2.18±1.14	7.1238	00019	273.2500	1.05	0.01
41	7.498058±0.078160	3.100000±1.220655	1.50±0.74	7.2600	00508	637.9500	1.03	0.00
42	7.632152±0.138677	3.766666±2.139833	1.94±1.34	7.3469	07908	5287.9000	1.04	0.00
43	7.739963±0.121484	4.033333±1.852625	2.18±1.23	7.4199	02005	3610.5000	1.04	0.00
44	7.771957±0.101445	3.166666±1.529342	1.81±1.09	7.4980	00139	1232.2000	1.04	0.00
45	7.853070±0.113490	3.066666±1.691810	1.76±1.23	7.5729	03244	3264.8000	1.04	0.00
46	7.948890±0.121830	3.200000±1.796292	2.04±1.43	7.6501	00032	856.5900	1.04	0.00
47	8.057679±0.117060	3.533333±1.707499	2.43±1.44	7.7241	00818	2506.6000	1.04	0.00
48	8.202992±0.138101	4.400000±1.993322	3.34±1.85	7.7912	02422	12961.0000	1.05	0.00
49	8.206333±0.138455	3.233333±1.977933	2.42±1.90	7.8868	10859	14312.0000	1.04	0.00
50	8.259007±0.113137	2.800000±1.600000	2.19±1.55	7.9475	00127	1605.0000	1.04	0.00

Chapter 6

Conclusions and Future Work

The objective of the second final year project was to develop a heuristic in order to solve one packing problem, which was packing identical unitary radius circles within a circular region minimizing its radius. We wanted to investigate the behavior of bio-inspired algorithms towards the packing problem in terms of efficacy and efficiency, so we expected them to solve the problem faster than the exact method in [1] with "good" results. In order to accomplish it, we studied and researched bio-inspired algorithms and natural methods, we wrote four algorithms in C language and made experiments using them during this semester. We concluded that the hypothesis was validated.

Due to lack of time, we did not get around to use the Exact Method in our computer nor applied the heuristics to solve others packing problems. And it was a challenging and stressful to conclude this project with many courses and a extracurricular activity to deal with in this semester. In regard to future work, we may apply the proposed heuristics to others packing problems and compare them in the same computer environment.

The courses that were important to me to do this project are listed below:

- SSC0501 - Introduction to Computer Science I
- SSC0503 - Introduction to Computer Science II
- SCC0502 - Algorithms and Data Structures I
- SCC0503 - Algorithms and Data Structures II
- SCC0213 - Scientific Methodology for Computer Science Research

Bibliography

[1] Birgin, G. Enesto and Gentil, J. Marcel. "New and improved results for packing identical unitary radius circles within triangles, rectangles and strips". Computers & Operations Research 37, 2010, pp. 1318-1327.

[2] Zhang, De-Fu and Li, Xin. "A Personified Annealing Algorithm for Circles Packing Problem". Acta Automatica Sinica, Vol. 31, No. 4, July 2004.

[3] Young-Bin, Shin and Eisuke, Kita. "Solving two-dimensional packing problem using particle swarm optimization". Computer Assisted Methods in Engineering and Science, vol. 19, 2012, pp. 241–255.

[4] N. Mladenovic, F. Plastria, and D. Urosevic. "Reformulation descent applied to circle packing problems". Computers Operations Research, vol. 32, 2005, pp. 2419-2434.

[5] D. Zhang and W. Huang. "A Simulated Annealing Algorithm for the Circles Packing Problem". Computational Science — ICCS 2004, Lecture Notes in Computer Science, vol. 3036, Springer–Verlag, Berlinand Heidelberg, 2004, pp. 206–214.

[6] Wang. Huaiqing, Huang. Wenqi, Zh.an.g. Quan, Xu. Dongming. "An improved algorithm for the packing of unequal circles within a larger containing circle". European Journal of Operational Research, vol. 141, 2002, pp. 440–453.

[7] De-fu Zhang and An-sheng Deng. "An effective hybrid algorithm for the problem of packing circles into a larger containing circle". Computers and Operations Research, vol. 32, August 2005, pp. 1941–1951.

[8] M.R. Garey, D.S. *Johnson Computers and intractability: a guide to the theory of NP-completeness*. Freeman, San Francisco, 1979.

[9] H. Rania, C. Babak, W. Olivier, V. Gerhard. "A Comparison of Particle Swarm Optimization and the Genetic Algorithm". 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, April 2005, pp. 18-21.

[10] V.G. Gudise and Ganesh K. Venayagamoorthy. "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks". Swarm Intelligence Symposium, May 2003.

[11] Birgin, G. Enesto and Gentil, J. Marcel. "New and improved results for packing identical unitary radius circles within triangles, rectangles and strips". Computers and Operations Research 37, 2010, pp. 1318-1327.

[12] Yudong Zhang, Shuihua Wang and Genlin Ji. "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications". Mathematical Problems in Engineering, Vol. 2015.

[13] R. M. Friedberg, B. Dunham, and J. H. North. "A learning machine: Part II". IBM Journal, vol. 3, no. 7, July 1959, pp. 282–287.

[14] R. M. Friedberg, "A learning machine: Part I". IBM Journal, vol. 2, no. 1, pp. 2–13, January 1958.

[15] Ron Wehrens and Lutgarde M.C. *Classical and Nonclassical Optimization Methods Buydens*. Encyclopedia of Analytical Chemistry, R.A. Meyers (Ed.), 2000, pp. 9678–9689.

[16] Webster CS. "Alan Turing's unorganized machines and artificial neural networks: his remarkable early work and future possibilities". Evolutionary Intelligence, 2012, vol 5, pp. 35-43.

[17] Copeland, B.J. "What Is Artificial Intelligence?" AlanTurin.net, May 2000. Accessed 20 October 2016.

[18] Wikipedia contributors. "Learning.". Wikipedia, The Free Encyclopedia, 13 Sep. 2016. Accessed Web. 13 Sep. 2016.

[19] Wikipedia contributors. "Evolution.". Wikipedia, The Free Encyclopedia, 13 Sep. 2016. Accessed Web. 13 Sep. 2016.

[20] Engelbrecht, Andries . *Computaional Intelligence: An Introduction*. 2rd edition, Wiley, 2007, pp. 128.

[21] Engelbrecht, Andries . *Computaional Intelligence: An Introduction*. 2rd edition, Wiley, 2007, pp. 285.

[22] Prigogine, Ilya and Stengers, Isabelle. *Order out of chaos: Man's new dialogue with nature*. Bantam Books, 1984.

[23] Wikipedia contributors. "Self-organization.". Wikipedia, The Free Encyclopedia, 20 Oct. 2016. Web. 20 October 2016.

[24] Krugman, Paul. *The Self-Organizing Economy*. Blackwell Publishers, 1996.

[25] Brian R. Johnson and Sheung Kwan Lam. "Self-organization, Natural Selection, and Evolution: Cellular Hardware and Genetic Software". BioScience, 2010, Vol. 60, issue 11, pp. 879-885.

[26] Wikipedia contributors. "Swarm Intelligence". Wikipedia, The Free Encyclopedia, 13 Sep. 2016. Accessed Web. 13 Sep. 2016.

[27] Kennedy, J. and Eberhart, Russ. "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks, 199, pp. 1942–1948.

[28] Shi, Yuhui and Eberhart, Russ. "A modified particle swarm optimizer". Proceedings of IEEE International Conference on Evolutionary Computation, 1998, pp. 69–73.

[29] Bonyadi, M. R. and Michalewicz, Zbigniew. "Particle swarm optimization for single objective continuous space problems: a review" . Evolutionary Computation, Vol. 0, No. 0 , pp. 1-54, 2016.

[30] Engelbrecht, Andries . *Computaional Intelligence: An Introduction*. 2rd edition, Wiley, 2007, pp. 293.

[31] Wikipedia contributors. "Particle Swarm Optimization". Wikipedia, The Free Encyclopedia, 13 Sep. 2016. Accessed Web. 13 Sep. 2016.

[32] Sedlaczek, Kai and and Eberhard, Peter. "Augmented Lagrangian Particle Swarm Optimization in Mechanism Design". Journal of System Design and Dynamics, Vol. 1, No. 3, 2007.

[33] W. Q. Huang and R. Ch. Xu. "Two personification strategies for solving circles packing problem". Science in China (Series E), vol. 29, April 1999, pp. 347-353.

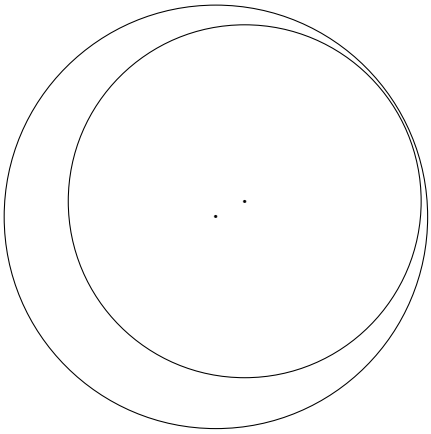
[34] Storn, Rainer and Price, Kenneth. "Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". Journal of Global Optimization, vol. 11, pp. 431–359, 1997.

[35] Engelbrecht, Andries . *Computaional Intelligence: An Introduction*. 2rd edition, Wiley, 2007, pp. 237.

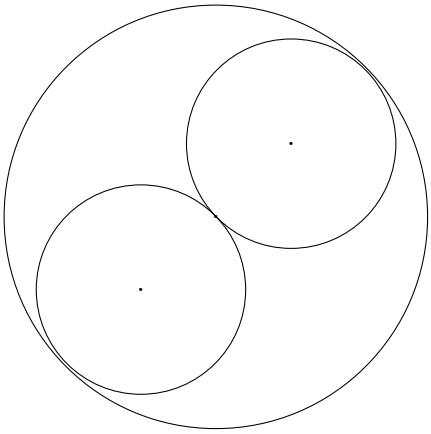
[36] Zhang, De-Fu and Li, Xin. "A Personified Annealing Algorithm for Circles Packing Problem". Acta Automatica Sinica, Vol. 31, No. 4. July 2004.

Appendix A

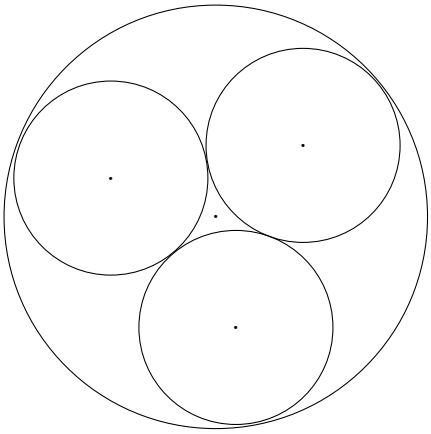
This appendix shows the pictures of the results of Static Quasi-Physical Differential Evolution (SQDE) for all tested instances.



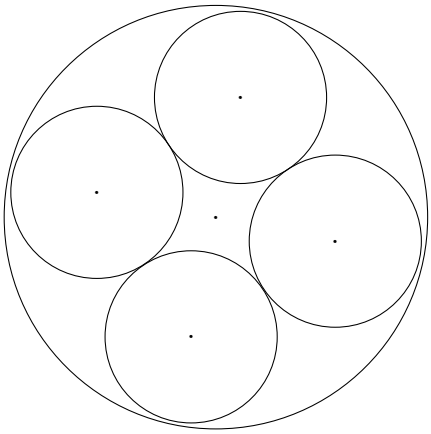
n = 1



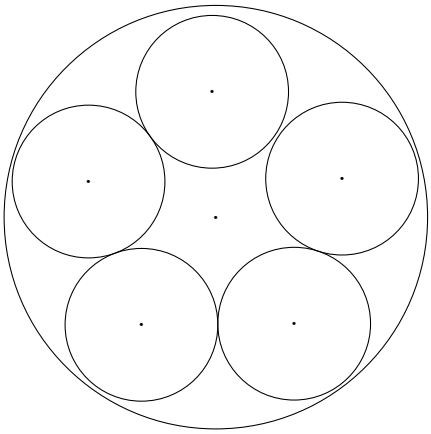
n = 2



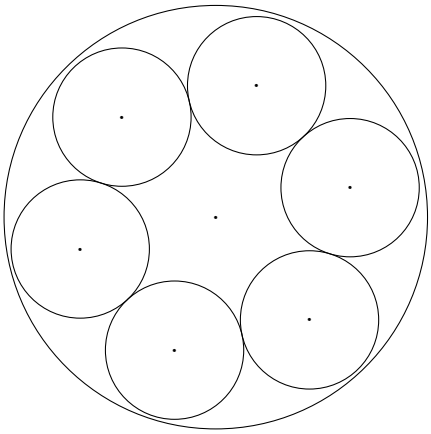
n = 3



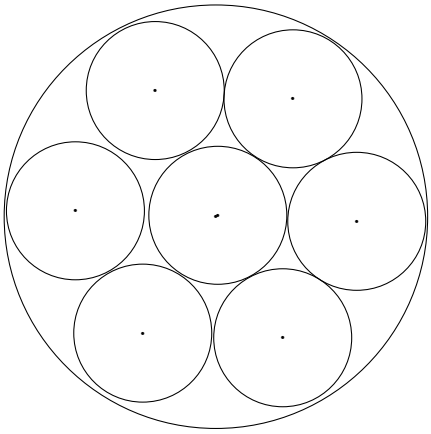
n = 4



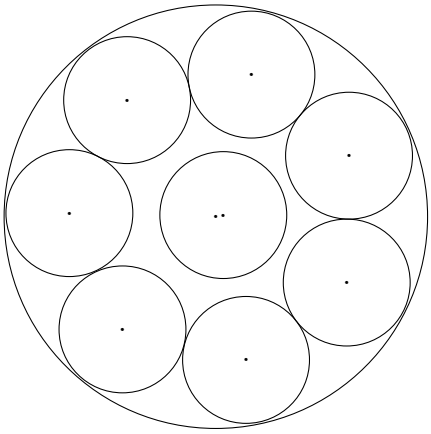
n = 5



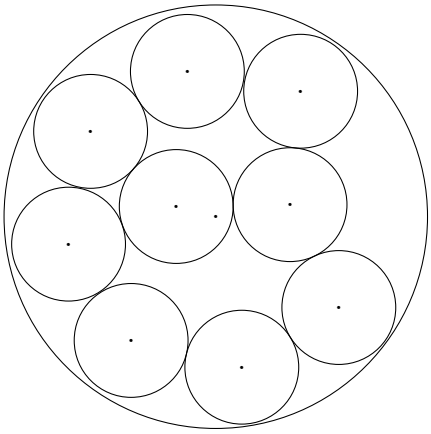
n = 6



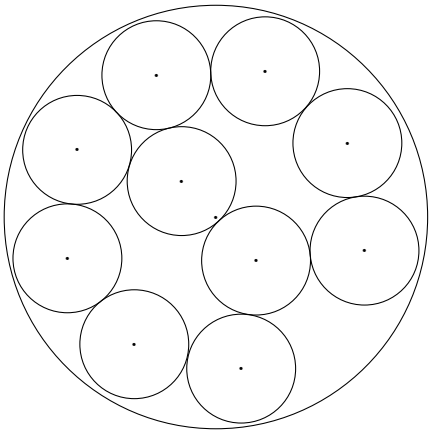
n = 7



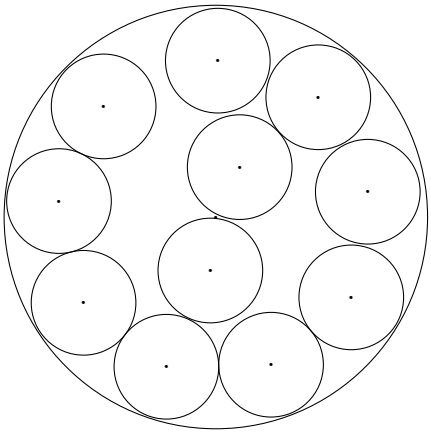
n = 8



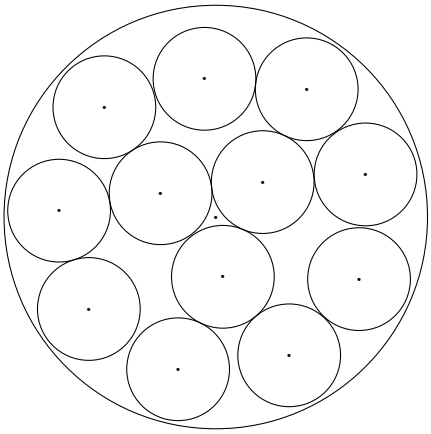
n = 9



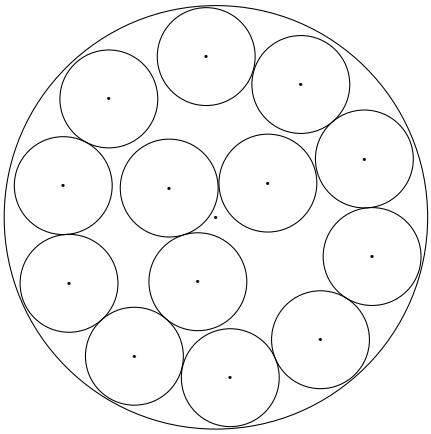
n = 10



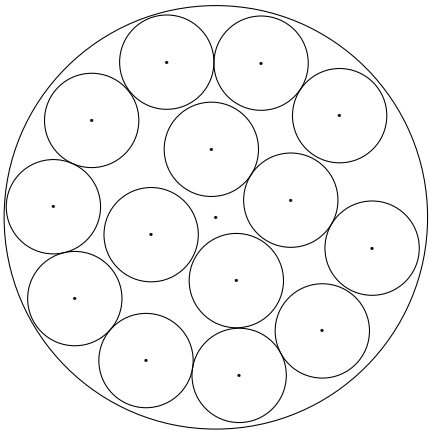
n = 11



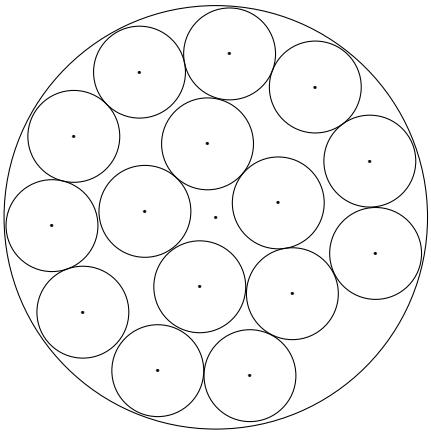
n = 12



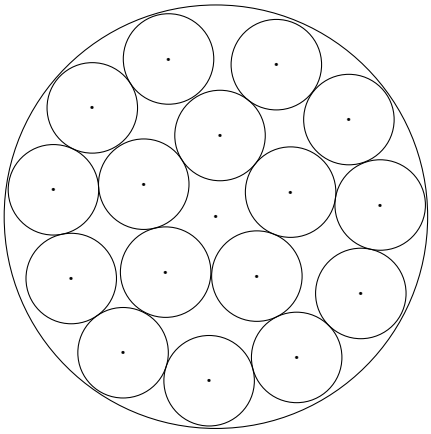
n = 13



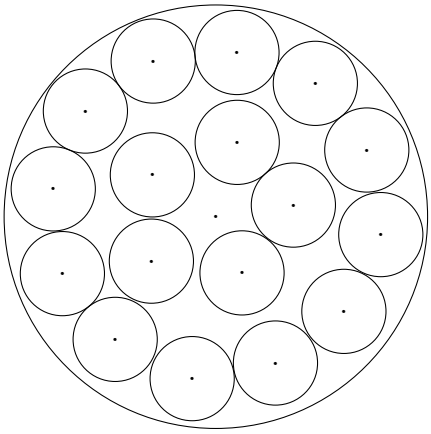
n = 14



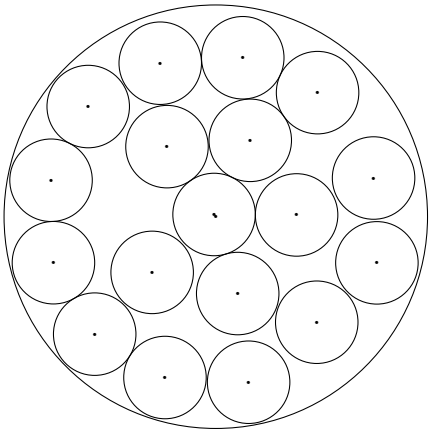
n = 15



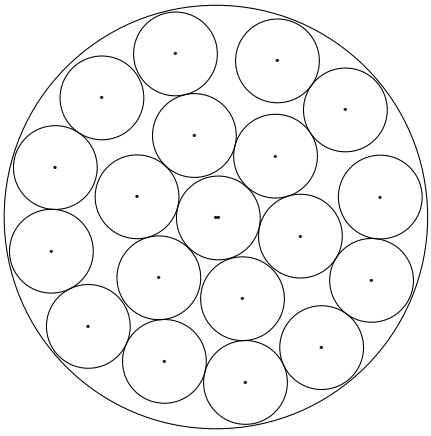
n = 16



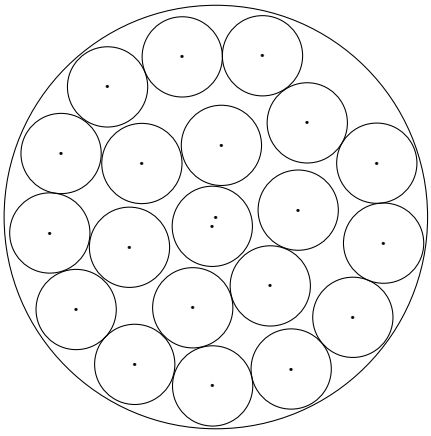
n = 17



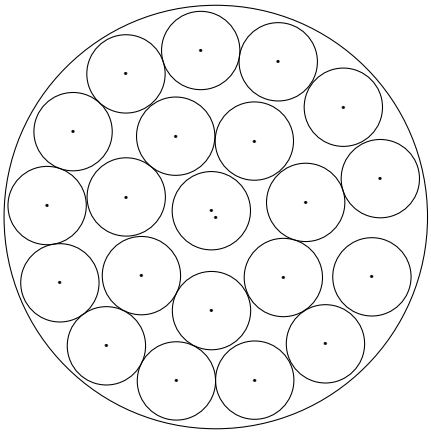
n = 18



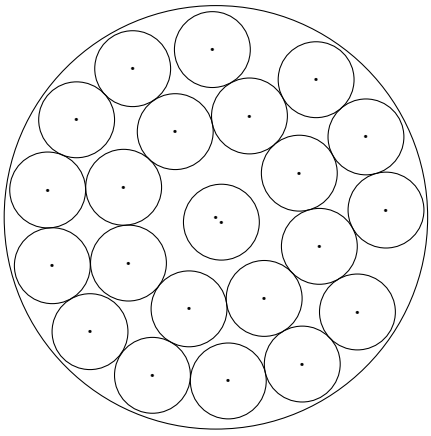
n = 19



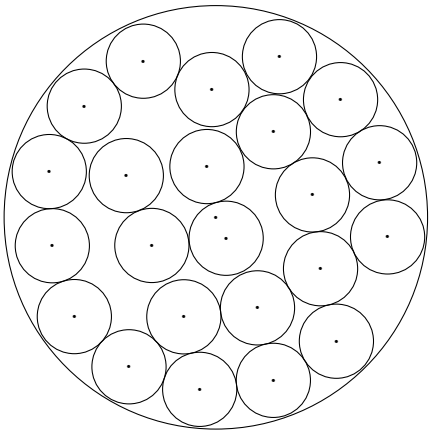
n = 20



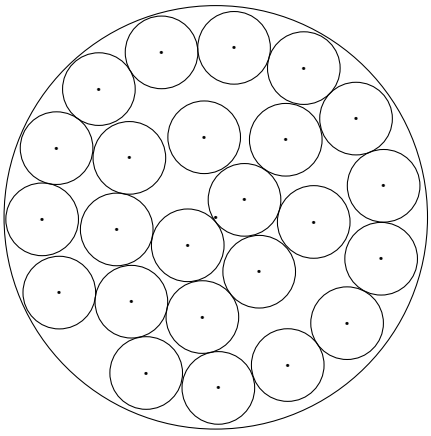
n = 21



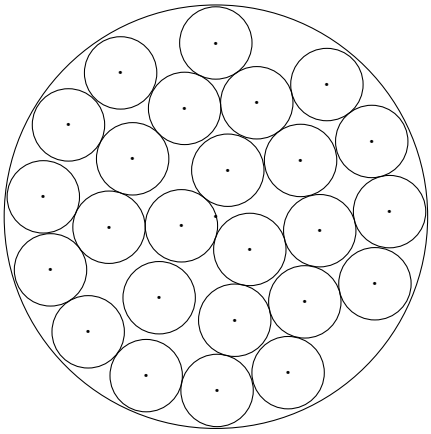
n = 22



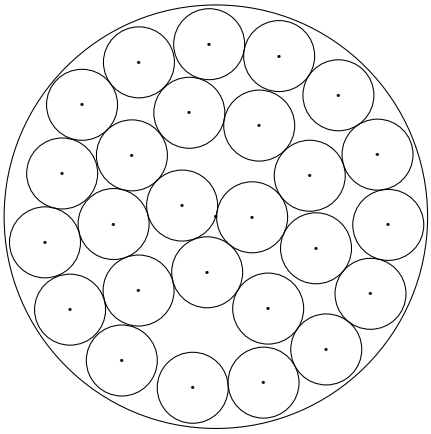
n = 23



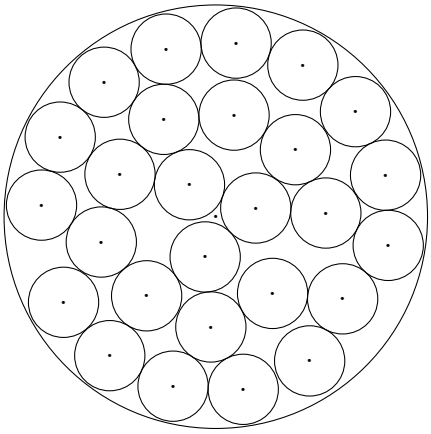
n = 24



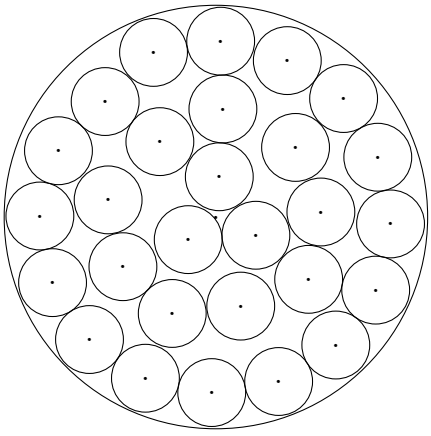
n = 25



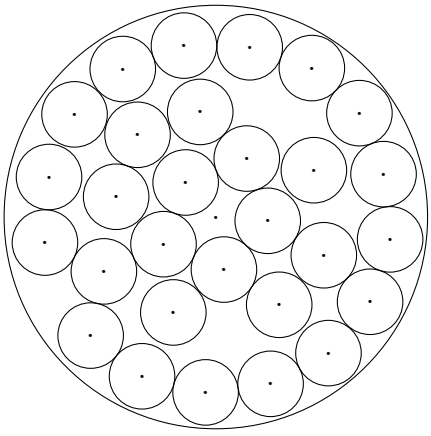
n = 26



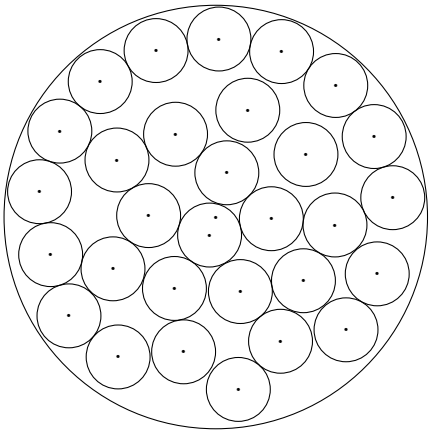
n = 27



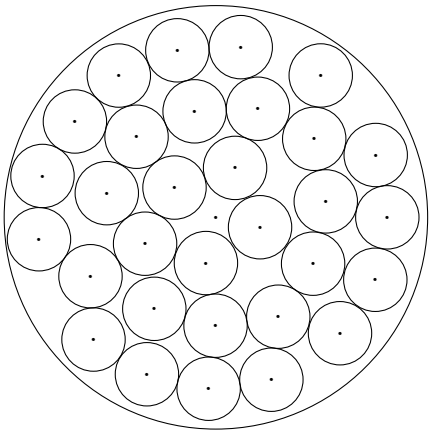
n = 28



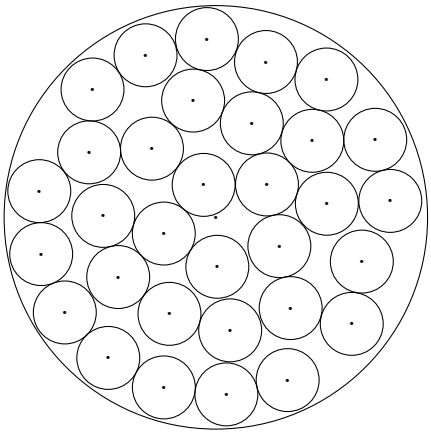
n = 29



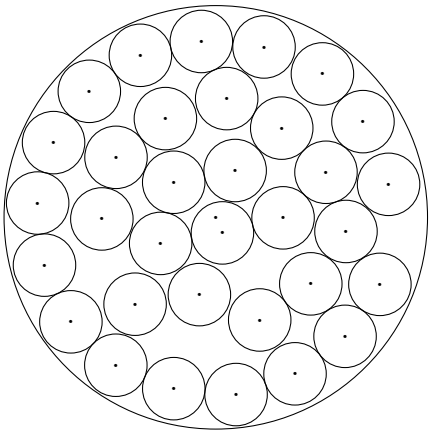
n = 30



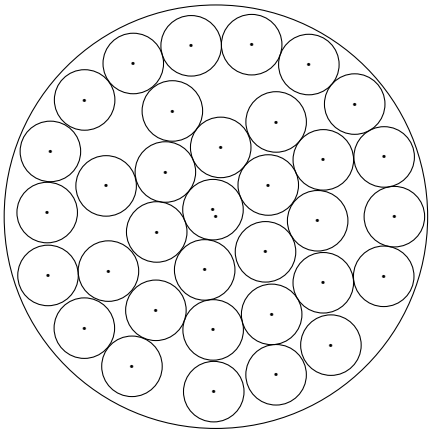
n = 31



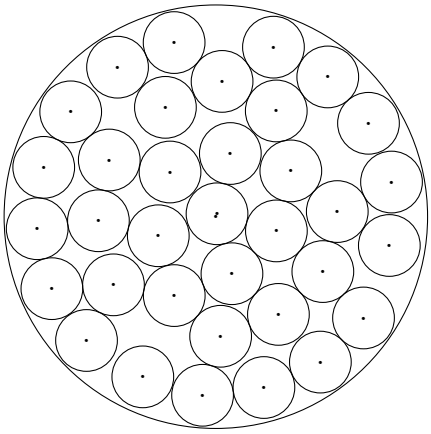
n = 32



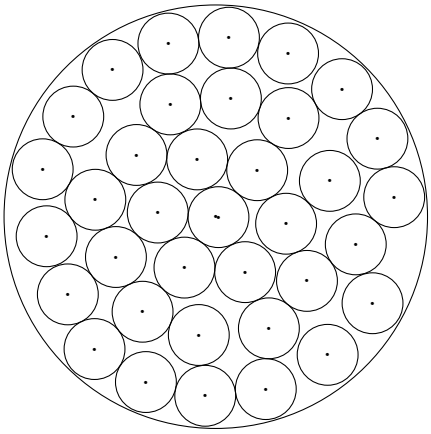
n = 33



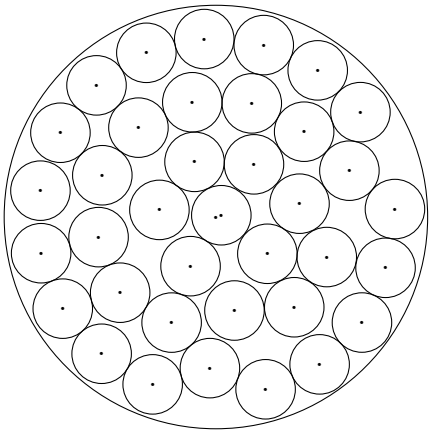
n = 34



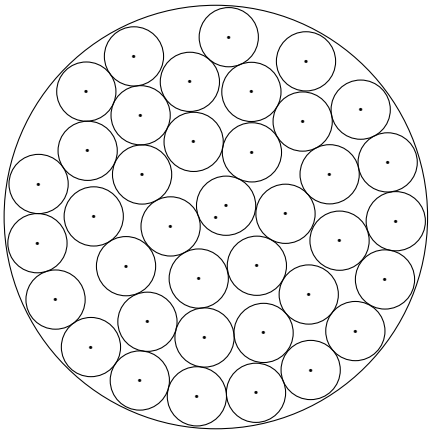
n = 35



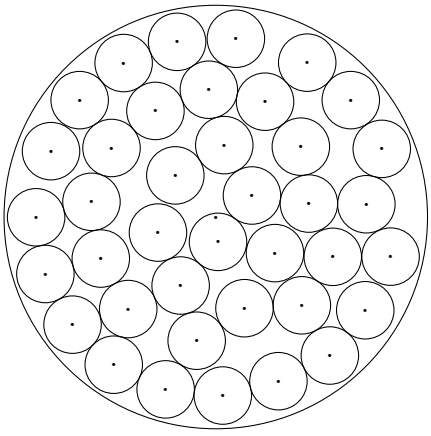
n = 36



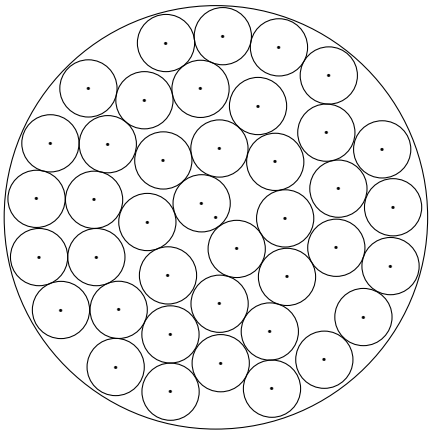
n = 37



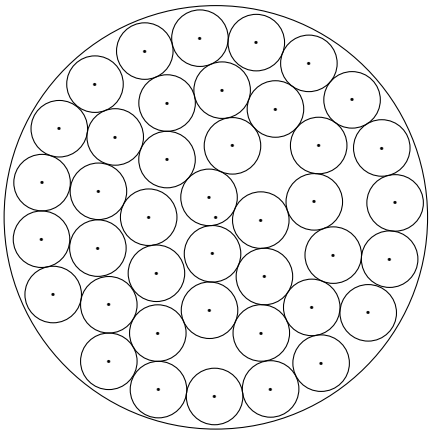
n = 38



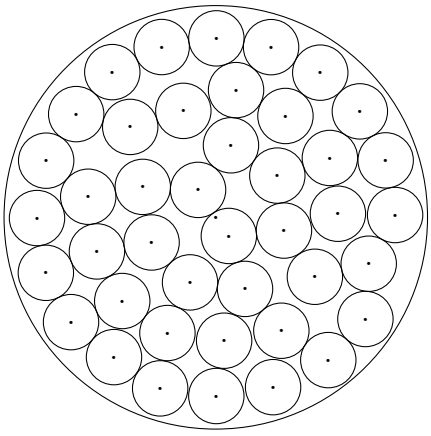
n = 39



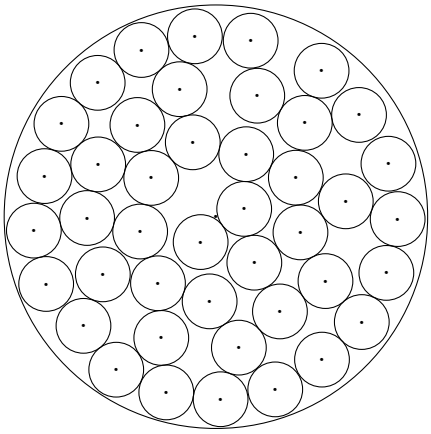
n = 40



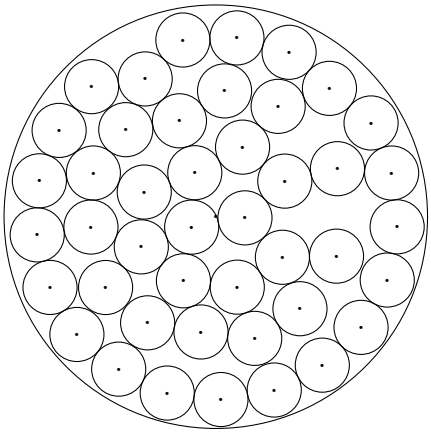
n = 41



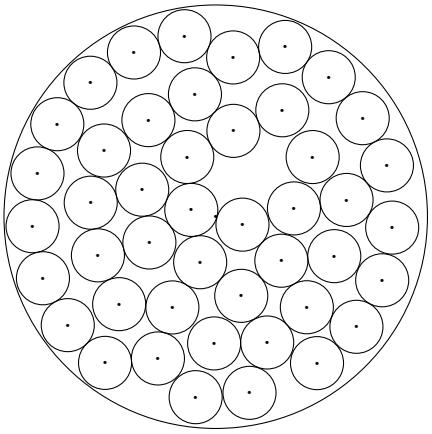
n = 42



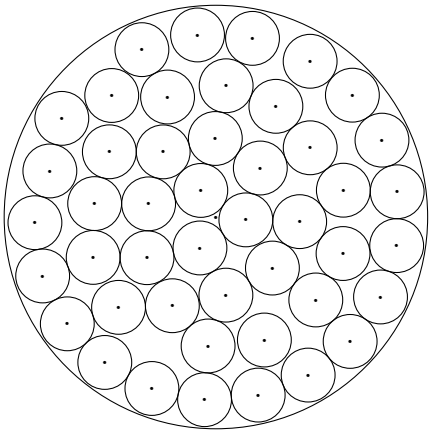
(a) n = 43



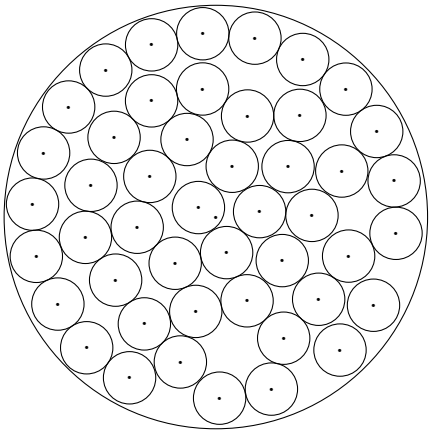
n = 44



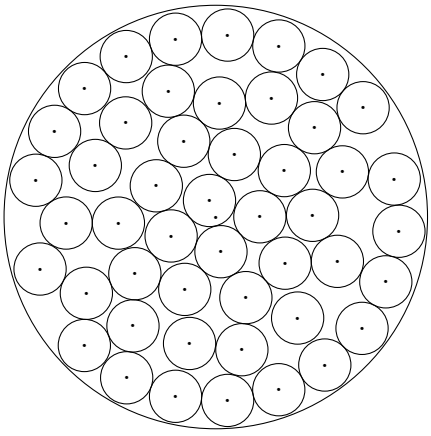
n = 45



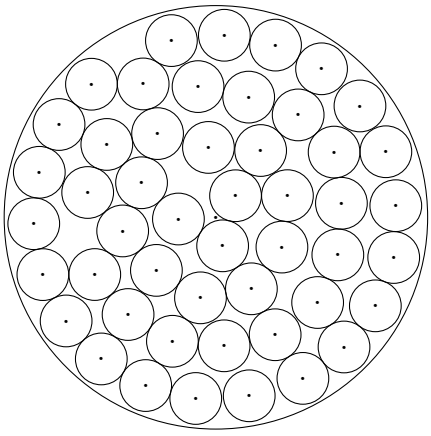
n = 46



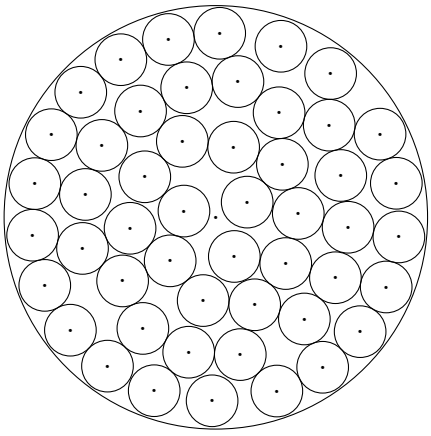
n = 47



n = 48



n = 49



n = 50