VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF MATHEMATICAL COMPUTER SCIENCE

Evolutionary optimisation algorithms: Particle Swarm Optimisation alogorithm

Course work

Author: Bioinformatikos 3 kurso studentas
Kazimieras Jasaitis

Supervisor: Lekt. Irus Grinis

Vilnius – 2023

TABLE OF CONTENTS

Introduction

In contemporary society, technology permeates every aspect of life, with computational devices ubiquitously performing tasks with high precision. Generally, computations in computers are deterministic, yielding predictable outcomes by following a preordained set of logical steps. This implies that for the same input, the output is invariably identical and is produced by what is termed as deterministic algorithms [AB09].

However, there are computationally intensive problems where deterministic algorithms are insufficient. One of the classifications of these problems is the NP (nondeterministic polynomial) problems. According to [Gol10], NP problems encompass decision problems where, if the answer is affirmative, a solution can be verified in polynomial time. The concept of nondeterminism is crucial for understanding NP problems. Imagine a computational model capable of making multiple choices at each step, exploring various possibilities. This, in essence, defines nondeterminism. In the context of NP problems, a nondeterministic algorithm can hypothetically guess a solution and then verify it within polynomial time. Nevertheless, the physical computers available are fundamentally deterministic [AB09; Gol10].

An alternative approach to solving complex optimization problems is through evolutionary algorithms. One such algorithm is the Particle Swarm Optimization (PSO), which has been widely adopted in applications like antenna design, biomedical applications, image processing, and robotics among others [Pol08; Sim13].

Collage creation is a representative task that seems trivial but escalates in complexity, particularly when numerous images of varying dimensions are to be placed on a canvas with predefined dimensions, with constraints like maintaining relative sizes, avoiding cropping, overlaps, or leaving any space unoccupied. This task necessitates the optimal and proportional resizing and positioning of images, which is computationally demanding and classified as an NP-Hard problem.

The problem of optimally placing images on a canvas is analogous to the 2D bin packing problem. The 2D bin packing problem entails fitting a set of 2D objects into a larger 2D container, making it pertinent to image placement where images are to be fitted on a canvas. Efficient solutions to the 2D bin packing problem can, therefore, be adapted for collage creation [WQZ15].

Particle Swarm Optimization (PSO) algorithm has been utilized to tackle the challenge of collage creation. The PSO algorithm exploits a population of candidate solutions that traverse the search space to locate an optimal or near-optimal solution for the collage creation problem, thereby providing an efficient and practical approach [Pol08; WQZ15].

Objectives:

1. Provide a detailed analysis of the Particle Swarm Optimization algorithm, elucidating its underlying principles and mechanics.

2. Develop a unit test generator that segments an image into n pieces, where n is an integer ranging from 2 to 5, to facilitate the evaluation of the PSO algorithm.

3. Implement the Particle Swarm Optimization algorithm and apply it to generate solutions for placing images on a canvas of dimensions $100px \times 100px$.

4. Execute the generated tests to evaluate the performance of the algorithm with varying numbers of image pieces.

5. Critically analyze the performance of the algorithm, considering metrics such as execution time, quality of solutions, and scalability.

6. Draw conclusions based on the experimental results and propose potential enhancements to the algorithm for future work.

1. Evolutionary optimisation algorithms

1.1. Introduction to Ant Colony Optimization

1.1.1. Behaviour of Ant Colonies

Ants are small insects, which individually possess limited capabilities. However, when they operate collectively within colonies, which sometimes consist of millions of ants, they exhibit highly sophisticated behaviors and can accomplish complex tasks. This success is attributed to their collective intelligence and ability to adapt to various environments. [DS04].

One of the primary communication mechanisms among ants is through pheromones, which are chemicals they secrete. For instance, when ants forage for food, they leave a pheromone trail along their path. Other ants, equipped with antennas sensitive to these chemicals, follow these pheromone trails to locate food sources. As more ants traverse this path, additional pheromones are deposited, reinforcing the trail. Consequently, the shortest path to the food source is often highlighted due to the concentration of pheromones. Over time, as the pheromones on older trails evaporate, the ants adapt by finding and reinforcing new optimal paths to new food sources. [DS04].

1.1.2. Ant Colony Optimization Algorithm

Inspired by the pheromone-depositing behavior of ants, the Ant Colony Optimization (ACO) algorithm was developed. ACO is a population-based metaheuristic that has been effectively applied to solve combinatorial optimization problems. Contrary to Evolutionary Algorithms (EAs), ACO does not involve direct exchange of solution information among candidate solutions. Instead, it simulates the process of ants depositing pheromones on the paths they traverse and subsequent generations of simulated ants using this information to make probabilistic choices in exploring the search space [Sim13].

1.1.3. Application to the Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a classic optimization problem. In the Selective Traveling Salesman Problem variant, which can also be referred to as the Orienteering Problem, a weighted graph with profits associated with vertices is given. The objective is to find a simple circuit with the maximum total profit, subject to the constraint that the circuit's length does not exceed a predetermined bound [LM90].

ACO can be effectively employed to solve the TSP. In ACO, a population of simulated ants is initially positioned. Each ant selects a city to travel to based on probabilistic rules that account for the attractiveness of the path (often influenced by distance) and the intensity of the pheromone trail. As ants traverse between cities, they deposit pheromones, reinforcing paths. Over time, pheromones on less favorable paths evaporate, making them less attractive, while the paths more frequently taken become more attractive due to increased pheromone deposits. Through iterative processing, the algorithm converges towards an optimal or near-optimal solution. [GR04]

1.2. Genetic Algorithms and Their Application to the Traveling Salesman Problem

1.2.1. Introduction to Genetic Algorithms

Genetic Algorithms (GAs) are among the earliest, most well-known, and widely used Evolutionary Algorithms (EAs). GAs are inspired by the process of natural selection and are employed to solve optimization problems. However, it is pertinent to note that GAs encompass a much broader class of systems than mere function optimizers [Sim13].

Natural selection exhibits certain fundamental characteristics, which include:

1. A biological system comprises a population of individuals, many of whom have the ability to reproduce.
2. The individuals have a finite lifespan.
3. Variation exists within the population.
4. Survival capabilities are positively correlated with reproductive abilities.

GAs simulate these features. For an optimization problem, a population of candidate solutions, termed individuals, is created. The individuals with better solutions have higher chances of reproduction, while those with poorer solutions have lower chances. Over generations, as parents produce offspring and are subsequently replaced by them, the population becomes more adapted. Occasionally, highly fit individuals evolve, providing near-optimal solutions.

1.2.2. Application to the Traveling Salesman Problem

Though the Traveling Salesman Problem (TSP) is often solved more efficiently using the Ant Colony Optimization algorithm, Genetic Algorithms can also be devised for this purpose. The GA approach involves the following steps:

1. Initialize essential parameters such as city locations, maximum number of generations, population size, crossover probability, and mutation probability.
2. Generate an initial random population of chromosomes and evaluate the fitness of each chromosome.
3. Transform the population into a new generation using three genetic operators: selection, crossover, and mutation. The selection operator chooses parent chromosomes, which then produce offspring through crossover and/or mutation.
4. Retain a higher proportion of characteristics from the fitter members of the previous generation, facilitating the propagation and mixing of advantageous traits.
5. Produce a new set of chromosomes equal in size to the initial population.

6. Repeat the transformation process until the population converges to an optimal solution or other stopping criteria are met.

The process ensures that with each successive generation, the population evolves towards a more optimal solution, generally culminating when a significant proportion of the population possesses the same optimal chromosome [RG11].

2. Particle Swarm Optimisation

2.1. How does it work?

Most EAs tend to be comparatively static as they simulate the evolution of potential solutions across generations without explicitly modeling the dynamism of their movement through the search space.

In PSO, as an individual progresses through the search space, it possesses inertia, thereby tending to sustain its velocity. Nonetheless, the velocity is subject to alteration owing to two principal factors:

1. The individual retains memory of its optimum position historically and has an inclination to modify its velocity in an effort to revert to that position. This is akin to human propensities to reminisce about the halcyon days and endeavor to rekindle past experiences. Within PSO, as the entity navigates the search space, its position undergoes a transformation from one generation to another. However, it keeps track of its performance through prior generations and recollects the coordinates within the search space where it achieved its peak performance.

2. Each particle is aware of the most favorable positions held by its neighbours during the current generation. This requires that all particles communicate their respective performances.

These dual facets randomly modulate an individual's velocity and bear resemblance to human social dynamics. There are instances when an individual may exhibit a higher degree of resolve, thus not being substantially swayed by its peers. Conversely, at times, an individual may be more influenced by nostalgia, thus being significantly impacted by past achievements.

2.1.1. Inspired by nature

In Particle Swarm Optimizations (PSOs), which take cues from the swarming behavior of birds and schools of fish, numerous basic units known as particles are introduced into the parameter space of a specific problem or function. Each of these particles assesses the quality or fitness of its current position. Subsequently, every particle decides its path through the parameter space by integrating elements of its own historical fitness data with that of one or more fellow particles. This path is navigated at a speed that is dictated by the positions and the evaluated fitness levels of the other particles, coupled with some random variations. The particles with which a given particle can engage are termed its social neighborhood. The aggregate of these social neighborhoods across all particles constitutes the social network of the PSO.

2.1.2. Allusions to human social behaviour

The particle swarm algorithm imitates human social behavior. —James Kennedy and Russell Eberhart [Kennedy and Eberhart, 2001]

Particle swarm optimization (PSO) draws its inspiration from the realization that clusters of entities collaborate to enhance both their joint and individual performance in a given task. The fundamentals of PSO are evidently manifested in the behavior of animals and humans alike. In our

endeavors to better our performance in a particular task, we modify our strategies relying on certain elementary concepts. [Sim13]

2.1.3. Dimensions

As mentioned previously, PSOs have dimensions that define the possible solution space. Figuratively speaking, after each iteration the particles can take up any position in the N-dimensional space, depending on their parameters.

2.1.4. Topology

In Particle Swarm Optimization (PSO), the configuration concerning how each particle is influenced by its neighboring particles is referred to as the swarm's topology. The topology encompasses the arrangement of neighbors that exert influence on a given particle. In certain cases, the neighborhood surrounding each particle evolves with each generation, rendering the topology dynamic. When the neighborhood is localized and does not incorporate the entire swarm, it is referred to as a local or $lbest$ topology.

There are numerous methodologies to establish the neighborhood of each particle. For instance, neighborhoods can be predefined at the onset of the algorithm, creating static topologies that remain constant across generations. Alternatively, if the optimization process reaches stagnation, neighborhoods can be randomly redefined at that juncture. In an extreme scenario, a single neighborhood encompassing the entire swarm can be employed, implying that each particle is influenced by the entire swarm. This is termed the global or $gbest$ topology and is historically significant as it represents the original topology employed during the development of PSO. It continues to be widely utilized. [Sim13]

Apart from the global topology, other common topologies include the ring topology, where each particle is connected to two other particles; the cluster topology, where particles are highly connected within their clusters with sparse connections to other clusters; the wheel topology, in which a central particle is connected to all others while the others are solely connected to the central particle; and the square or von Neumann topology, where each particle is connected to four neighbors, forming a toroidal structure as connections wrap from top to bottom and left to right. [Sim13]

The choice of topology in the social network of a PSO is crucial as it plays a significant role in dictating the behavior and performance of the algorithm, sometimes being as influential as the velocity update equation. There have been numerous variants of PSO and a plethora of topologies that have been developed and evaluated. PSO, with its varied topologies, has been successfully applied across a diverse range of applications. It has demonstrated particular effectiveness in multimodal problems and in scenarios where no specialized method exists or where specialized methods yield unsatisfactory results. [Sim13]

2.1.5. Parameters

As any other evolutionary algorithm the PSO defines certain parameters, some of which are unique to this optimisation algorithm. These parameters can influence the behaviour of the algorithm, exploration and exploitation rates. [Eng07]

2.1.5.1. inertia weight

Inertia weight is a factor that contributes to the momentum of the particles in PSO. Essentially, it determines the extent to which a particle's previous velocity influences its current velocity. It is the component that allows the particle to sustain its directional course through the search space. A high inertia weight encourages exploration by permitting a broader search, which may help in escaping local minima. Conversely, a low inertia weight tends to facilitate exploitation, enabling the particle to conduct a more localized search. This is essential for refining solutions in the vicinity of the perceived optimum. Balancing the inertia weight is vital for ensuring an effective trade-off between exploration and exploitation. [Eng07]

2.1.5.2. cognitive weight

Cognitive weight, often denoted by a constant, reflects the significance given to the individual experiences and memory of particles. It represents how much a particle's velocity is influenced by its personal best position encountered thus far. When a particle moves through the search space, it keeps track of the best position it has ever achieved. The cognitive weight determines how strongly the particle is drawn towards this personal best position. A higher cognitive weight prompts the particle to favor its historical experiences, steering it back towards regions of the search space where it has previously found promising solutions. [Eng07]

2.1.5.3. social weight

Social weight, also commonly referred to as global weight and often represented by another constant, signifies the importance accorded to the collective knowledge of the swarm. It quantifies the degree to which a particle's velocity is influenced by the best position known to the entire swarm. Essentially, it reflects how much a particle is attracted towards the global best position found by any member of the swarm. A high global weight encourages cooperative behavior among particles, driving them towards regions of the search space deemed promising by their peers. [Eng07]

2.1.6. Velocity

In Particle Swarm Optimization, velocity is a fundamental attribute that dictates the movement of particles through the search space. It is a vectorial quantity, dynamically adjusted in each iteration, that influences how a particle approaches potential solutions. The calculation of velocity encompasses various components, including inertia, cognitive velocity, and global velocity, which collectively balance the particle's momentum, individual experiences, and social information. The

judicious modulation of velocity is essential for efficiently navigating the search space, striking a balance between exploration and exploitation in pursuit of optimal solutions. [Eng07]

2.1.6.1. cognitive velocity

Cognitive velocity represents the component of a particle's velocity that is influenced by its own historical best position. This aspect reflects the tendency of a particle to gravitate towards the most favorable position it has encountered in the past. The cognitive velocity is calculated as the product of the cognitive weight (often denoted by a constant) and the difference between the particle's personal best position and its current position. This component essentially quantifies the pull towards the particle's historical achievements, guiding it based on its individual experiences. [Eng07]

2.1.6.2. global velocity

Global velocity, on the other hand, corresponds to the component of a particle's velocity that is influenced by the best position known to the entire swarm, or the global best position. This component reflects the inclination of the particle to move towards the most promising area found by any member of the swarm. The global velocity is computed as the product of the global weight (also often represented by a constant) and the difference between the swarm's global best position and the particle's current position. This component essentially quantifies the pull towards collective knowledge and achievements within the swarm. [Eng07]

2.1.7. Well established applications

The PSO algorithm can be implemented in solving a very broad range of problems. An analysis of the Publications on the Applications of Particle Swarm Optimisation details a broad range of applications. The most prevalent of which were in antenna design, biomedical applications, and modeling.

Antenna Design: The telecommunications industry heavily relies on antennas for the propagation of signals. The performance and efficiency of antennas are crucial for reliable communication. PSO algorithms help in optimizing various attributes of antenna structures. For instance, it can be used for fine-tuning phased array antennas to achieve desired beam direction and shape. Additionally, PSO assists in the design of broadband antennas to ensure efficient transmission over a range of frequencies. The customization of patch antennas, reflector antennas, and the design of miniaturized antennas for portable devices are some other applications where PSO has proven to be extremely beneficial.

Biomedical Applications: The complexity and sensitivity of biomedical data and systems necessitate sophisticated optimization techniques. PSO algorithms have been embraced in the biomedical sector for a plethora of applications. For instance, PSO is employed in the optimization of radiotherapy planning, where the objective is to maximize the dosage to cancerous tissues while minimizing exposure to surrounding healthy tissues. Additionally, PSO algorithms have been

used for gene clustering and the identification of transcription factor binding sites in DNA, which is significant in genomics research.

Modelling: The development of descriptive and predictive models is integral to understanding and solving complex problems across various domains. PSO algorithms are used in creating models that accurately represent phenomena or predict future outcomes. For example, in acoustics, PSO can be utilized for the inversion of underwater acoustic models which is crucial in sonar applications. In customer satisfaction analysis, PSO helps in developing models that can predict customer behavior based on various factors. In systems engineering, PSO is used for model order reduction which simplifies complex systems without significant loss of accuracy.

Apart from these, PSO algorithms are also widely applied in:

Image Processing: PSO is used for tasks like image segmentation, feature selection, and object recognition. It helps in optimizing the features and parameters that are crucial for efficient image analysis.

Robotics: In robotics, PSO can be used for path planning, where the algorithm finds the optimal path for a robot to navigate from a starting point to a destination, avoiding obstacles.

Supply Chain and Logistics: PSO algorithms help in optimizing logistics operations such as vehicle routing and warehouse management to minimize costs and improve efficiency.

Financial Markets: In finance, PSO is used for portfolio optimization and trading strategy development by analyzing patterns and trends in market data.

In conclusion, the versatility and efficiency of Particle Swarm Optimization algorithms have led to their adoption in a myriad of applications across diverse fields, making them an invaluable tool for solving complex optimization problems.

3. PSO to place rectangle images on a square canvas

Collage is a prevalent medium employed for the representation of information and the exhibition of a collection of images. Occasionally, the construction of a collage demands an artistic perspective to effectively arrange the given images in a manner that is aesthetically pleasing. However, frequently, the task becomes unnecessarily convoluted. While it might be relatively straightforward to find an optimal arrangement for a small number of images, the complexity increases substantially when the images have varying dimensions or do not completely fill the designated space. The task can be formulated as follows: Place a set of n images onto a canvas with predetermined dimensions, ensuring that the images are positioned such that there is no overlap or extension beyond the canvas boundaries. Furthermore, the images must be resized proportionally in relation to each other, and the arrangement should aim to minimize the amount of uncovered area on the canvas. It is worth noting that this task is computationally intractable and is classified as NP-Hard.

3.1. Relation to 2D bin packing problem

In the realm of optimization problems, the Two-Dimensional (2D) Packing Problem holds particular significance, especially in industries where efficient space utilization is crucial. The problem involves fitting a collection of 2D rectangular items into a confined area or container, with the objective to maximize the utilization of the available space. This requires careful arrangement of the items such that there is no overlap between them. The 2D Packing Problem finds its applications in various industries including manufacturing, logistics, and shipping. Due to the combinatorial nature of this problem, finding an optimal solution is computationally challenging, especially as the number of items increases. Therefore, various heuristic and meta-heuristic algorithms such as Particle Swarm Optimization, Genetic Algorithms, and Simulated Annealing are often employed to find approximate solutions within a reasonable time frame.

The image placement problem shares similarities with the classical 2D bin packing problem, which can be defined as follows: The task is to pack a collection of two-dimensional rectangles into the minimum number of larger rectangular bins, where each bin has a predetermined width and height. The primary objective is to arrange the rectangles efficiently within the bins, ensuring that each rectangle is entirely contained within a bin, with no overlap and without any rotation.

It is pertinent to recognize the image placement problem defined in this article as a variant of the 2D bin packing problem. The distinction here is that the focus is not on minimizing the quantity of bins but on strategically positioning and scaling a collection of images onto a sheet of paper. The sheet of paper could be analogous to a bin, and the images - the objects awaiting packing. Adhering to the constraints of the 2D bin packing problem, the images must be positioned within the sheet without overlap and must not exceed the boundaries of the paper.

However, the image placement problem introduces some intricacies to the 2D bin packing problem. One such complexity is the necessity to scale images. This adds an additional layer to the problem, which is absent in the conventional 2D bin packing problem. This task encompasses the dual challenge of ascertaining the optimal position for each image coupled with identifying

an appropriate scaling factor. Moreover, this project could extend beyond the confines of packing efficiency. Here, needs of digital designers need to be taken into account. From this, the need for consistency in image scaling, tolerance for overlapping and out of boundary solutions arises.

Therefore, though this project bears semblance to the 2D bin packing problem by focusing on the efficient allocation of 2D objects within a limited space, it incorporates additional elements. This suggests that while algorithms and solutions employed in classical 2D bin packing could provide a rudimentary framework, there is a necessity to adapt and augment these strategies to cater to the distinct nature of this project.

3.2. Solution using PSO

3.2.1. Input

The program performs a scan of the specified folder to identify image files, specifically those with the file extensions .png, .jpg, and .jpeg[1]. Subsequently, it extracts the dimensions of these images. It is noteworthy that the program can be adapted to accept an array of image dimensions as input, offering flexibility for modifications and extensions.

3.2.2. Output

The program provides the solution as a text file containing image names, coordinates and their relative scale respectively. Image coordinates are calculated for the left bottom corner of the image. The scale is relative to the original scale of each individual image regardless of their dimensions. Each image is initialised with a scale of 1.

3.2.3. Fitness

Given the complexity of the problem, it is imperative that the fitness function considers multiple factors to achieve the desired outcome. A solution is deemed perfect if it attains a fitness score of zero. However, owing to the approach adopted for the implementation of this optimization algorithm, it is possible that certain input combinations do not yield a perfect solution.

To ensure the program's effectiveness in scenarios where a perfect solution is unattainable, adjustments to fitness values are necessary, taking into consideration the specific requirements of digital designers. Particularly, the algorithm must impose harsher penalties on solutions that permit overlaps compared to those that leave areas uncovered. This entails the incorporation of several adjustment factors.

In the fitness function of the algorithm, these adjustment factors are integrated as multipliers for the corresponding penalty factors:

- $scalingpenaltyfactor$ = 10,
- $boundarypenaltyfactor$ = 10,

[1]The file extensions considered are .png, .jpg, and .jpeg.

- $overlappenaltyfactor$ = 10.

These values are configured such that penalties associated with leaving areas uncovered are less severe.

3.2.3.1. Resizing

Moreover, solutions are subjected to penalties for substantial deviations in resizing. These penalties are computed in relative terms, as the algorithm aims to cover the plane with the provided images.

For instance, consider a scenario where the program is supplied with four images, each having dimensions of $25px \times 25px$. To fully cover the plane, the images must be scaled up by a factor of two. A desired solution in this case would have the following characteristics:

- Image1: x = 0; y = 0; $scale$ = 2;
- Image2: x = 50; y = 0; $scale$ = 2;
- Image3: x = 0; y = 50; $scale$ = 2;
- Image4: x = 50; y = 50; $scale$ = 2.

For such a solution to attain the perfect fitness score of zero, the scale penalty must be computed relative to each image. Essentially, scaling is a vital component of the solution, and to ensure that the scaling of images does not adversely impact the fitness score, the scale factor for all images must be identical.

3.2.3.2. Overlapping

The algorithm imposes penalties on solutions wherein the images overlap. The objective is to arrange all the images on a designated plane, such as a piece of paper, ensuring that they are all discernible and free of overlaps.

3.2.3.3. Boundaries

The instances where images extend beyond the boundaries are penalized analogously to the overlapping of images. Essentially, cases where an image extends beyond the designated boundaries can be conceptualized as overlapping, where the image in question is obstructed by the confines of the designated plane.

3.2.3.4. Uncovered Area

Certain combinations of inputs may not yield optimal solutions. In such scenarios, it is imperative to discriminate between solutions based on their desirability. In the context of image placement, preventing clipping is often prioritized as it can obfuscate crucial visual information and

necessitate manual adjustments by the editor, rendering the automated solution ineffectual. Additionally, alternative artistic approaches, such as the incorporation of a background or the inclusion of supplementary images to fill the gaps, can be employed. Consequently, the fitness function is calibrated to favor solutions with uncovered areas over those with clipping or overlapping. This adaptation yields more pragmatic results in scenarios where optimal solutions are either unattainable or remain unidentified.

3.2.4. Initialization

3.2.4.1. Position

Each particle is initialized with an x-coordinate and a y-coordinate, both randomly selected within the range [0, 100], and an initial scale factor of 1. This initialization strategy positions the program in close proximity to the optimal solution in scenarios where the images fully cover the plane at their original scale. The x and y boundaries are set within the range that corresponds to feasible solutions.

3.2.4.2. Velocity

Each particle is initialized with a velocity value randomly selected within the range [-1, 1].

3.2.5. Execution

3.2.5.1. Velocity

As previously discussed, velocity plays a crucial role in determining the efficacy of a Particle Swarm Optimization (PSO) algorithm. Precise computation of velocity is indispensable for the success of the algorithm. There are five hyperparameters that aid in the fine-tuning of the algorithm, three of which have a direct impact on velocity computation.

The new velocity for each particle is derived by combining the current velocity, cognitive velocity, and social velocity.

- Current Velocity: It is multiplied by an inertia weight (w) to contribute to the new velocity.
- Cognitive Velocity: It is computed by multiplying the cognitive scaling parameter ($c1$) by a random value ($r1$[2]) and the vector directed towards the particle's personal best position.
- Social Velocity: Similar to cognitive velocity, social velocity is computed by multiplying the social scaling parameter ($c2$) by a random value ($r2$[3]) and the vector directed towards the global best position.

The parameters $r1$ and $r2$ are incorporated to instill stochasticity into the velocity update equation, which fosters diverse exploration of the search space by the particles.

[2] $r1$ and $r2$ introduce stochasticity into the velocity update equation, enabling the particles to explore the search space more diversely.

[3] Refer to the footnote regarding $r1$.

3.2.5.2. Position

The position of each particle is updated by adding its current velocity to its existing position. This process iteratively modifies the positions of the particles in the search space.

3.3. Performance evaluation

3.3.1. Reverse Engineering Benchmarking

How well the program performs will be evaluated by giving it a problem that the solution is known for. In this case, for the evaluation to be quantitative and more accurate it is important for the perfect solution (fitness of 0) to exist for given input. By providing the program with input that we know has at least one perfect solution we can perform benchmarking and objectively gauge the performance of the algorithm.

3.3.2. Unit test generator

A program cuts an image up into n pieces, where n is a natural number $1 > n > 5$. It does so by selecting 1-2 random points on the image and drawing orthogonal cut lines on it. Then The program crops the original image n times thus generating n images that cover the whole plain, given the original image is of the same proportions, as the paper we are trying to cover.

3.3.3. Testing

10 tests were carried out with each of the values of n. The test generator cuts the given image into n pieces 10 times.

To account for the test generator generating problems of different complexities, each test was repeated 10 times. Overall the algorithm was run 100 times for each value of n.

First, tests were made for each value of n only changing the population size intuitively.

The algorithm was set to stop after 1000 iterations without any improvement on the best global position, assuming at this point it has already converged.

This approach allows for determining the convergence rate for the algorithm with each test and different values of n. In addition, if the algorithm does not converge in any of the 10 runs, the best found solution for each test is saved and can be applied or analysed.

The image chosen is a semi-transparent visual in the .png format with the goal to effortlessly detect overlapping.

Fig. 1. Image used for test generation

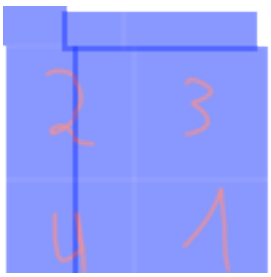

Fig. 2. Solution where pieces of the image overlap

3.3.3.1. Hyper-parameter values

The values for the first run were set to the following:

1. w = 0.7
2. $c1$ = 1
3. $c2$ = 2
4. $desiredfitness$ = 0.005

For these tests a canvas of $100px \times 100px$ was chosen and the image provided for the tests is of the same dimensionality. At first, the program was run from the n of 2 to the n of 5. Tests for n from 2 to 5 were performed with the population size of 500. The population size was increased to 750 and tests were repeated for n = 5. All other parameters remained the same throughout the experiment.

3.3.3.2. What to expect

Since this testing approach provides tasks that have at least one perfect solution, the x and y values should be in the range of 0-100. For the first basic tests of the algorithm, the image provided to the test generator is of the same dimensions as the canvas. Therefore, the scale of each piece, assuming the perfect solution was found, should be exactly 1. Given the nature of particle swarm optimisation algorithms, the algorithm is not expected to converge 100% of the time.

3.3.3.3. n=2

Since the tests provide solutions that cover the whole plain, with the n of, 2 pieces of one image is passed to the program and essentially it only has to place each of the images on the opposite sides of the paper. This, so far, is a trivial task for humans, so this is a rudimentary stage for the algorithm to successfully work, else chances are slim that it would provide any value or serve any useful purpose further down the line.

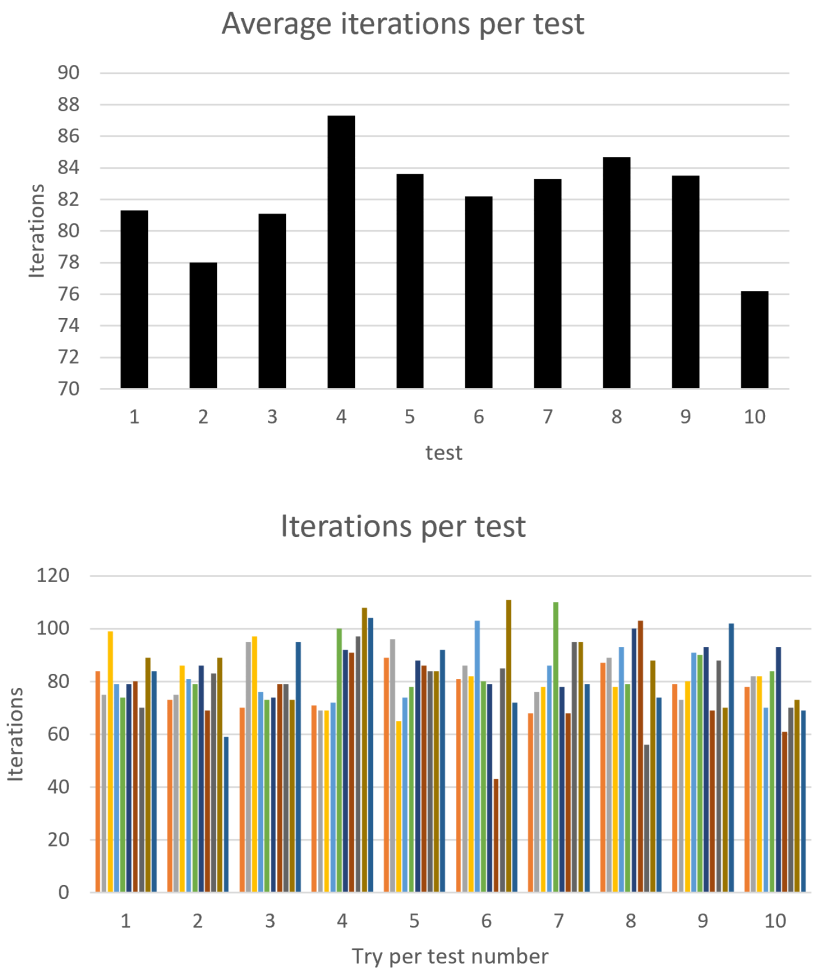

Fig. 3. Results for 98 successful tests run when n=3

3.3.3.4. n=3

With the n of 3, the task is equally as trivial for a human to solve if a simple algorithm of putting each of the images in opposing corners is followed. 10 tests were run 10 times each, resulting in 100 runs. The program successfully found a solution that has a desired fitness 98 times out of 100. On average it took 311 iterations to come up with a satisfactory solution. The median of these values is 296.5.

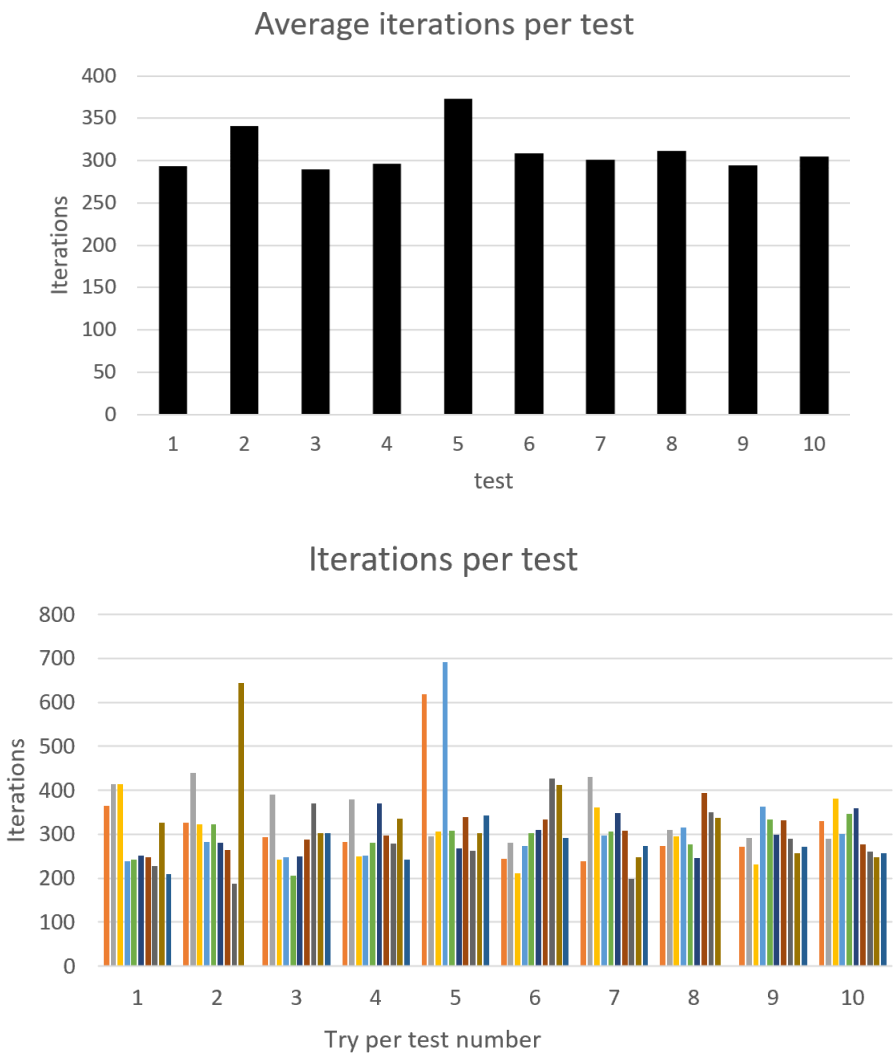

Fig. 4. Results for 98 successful tests run when n=3

A few spikes in iteration count are visible. Such spikes may suggest that some tests were more difficult than others, however the sample size is too low to draw any difinitive conclusions, in additionm besides these spikes, most solution times even for the same tests remain within 100 iterations from the median value.

3.3.3.5. n=4

When n is 4, the task remains trivial for a human as long as the same approach of putting each piece in one of the corners is followed. 10 tests were run 10 times each, resulting in 100 runs. The program was successful at finding a solution with the desired fitness 72/100 times. On average it took the algorithm 1270 iterations to come up with a satisfactory solution. The median for this test was 736 iterations. For tests where n = 4, the number of outliers is a lot bigger. The program became less reliable at finding satisfactory solutions and it took way longer on average.

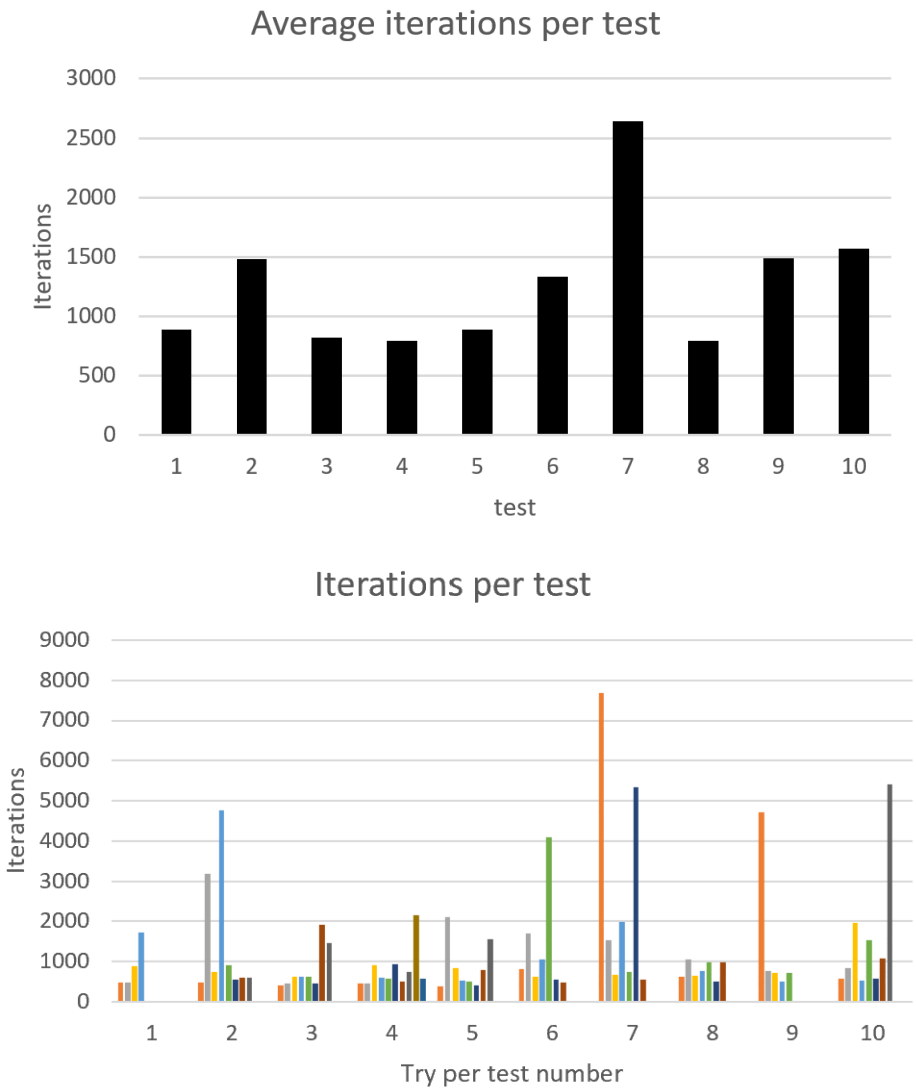

Fig. 5. Results for 72 successful tests run when n = 4

3.3.3.6. n=5

When n takes on a value of 5, the task becomes more complicated for humans. It now may require some rearrangement of the images to achieve the same result. 10 tests were run 10 times each, resulting in 100 runs. The program was successful at finding a solution with the desired fitness only 37/100 times. On average it took the algorithm 3298 iterations to come up with a satisfactory solution. The median for this test was 2625 iterations.

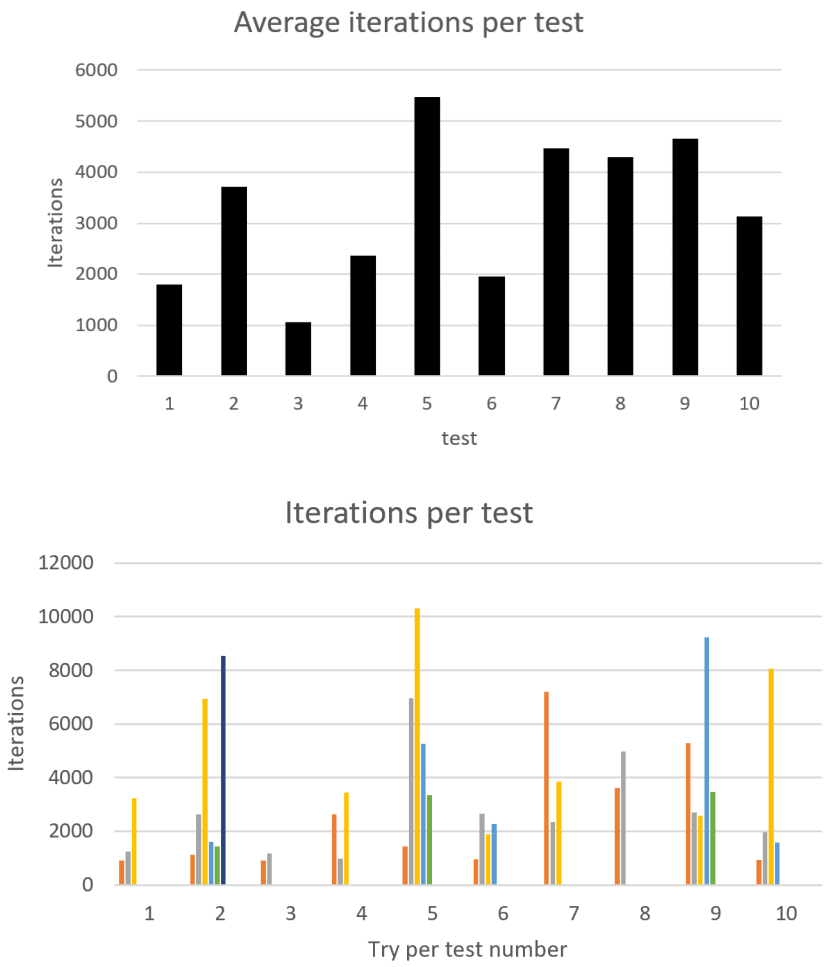

Fig. 6. Results for 37 successful tests run when n = 5

These results are were not satisfactory so the experiment was repeated with a larger population index for the n of 5. The test run with the population size increased showed a substantial improvement. The program was able to find satisfactory solutions times, the average iteration count was and the median run took iterations.

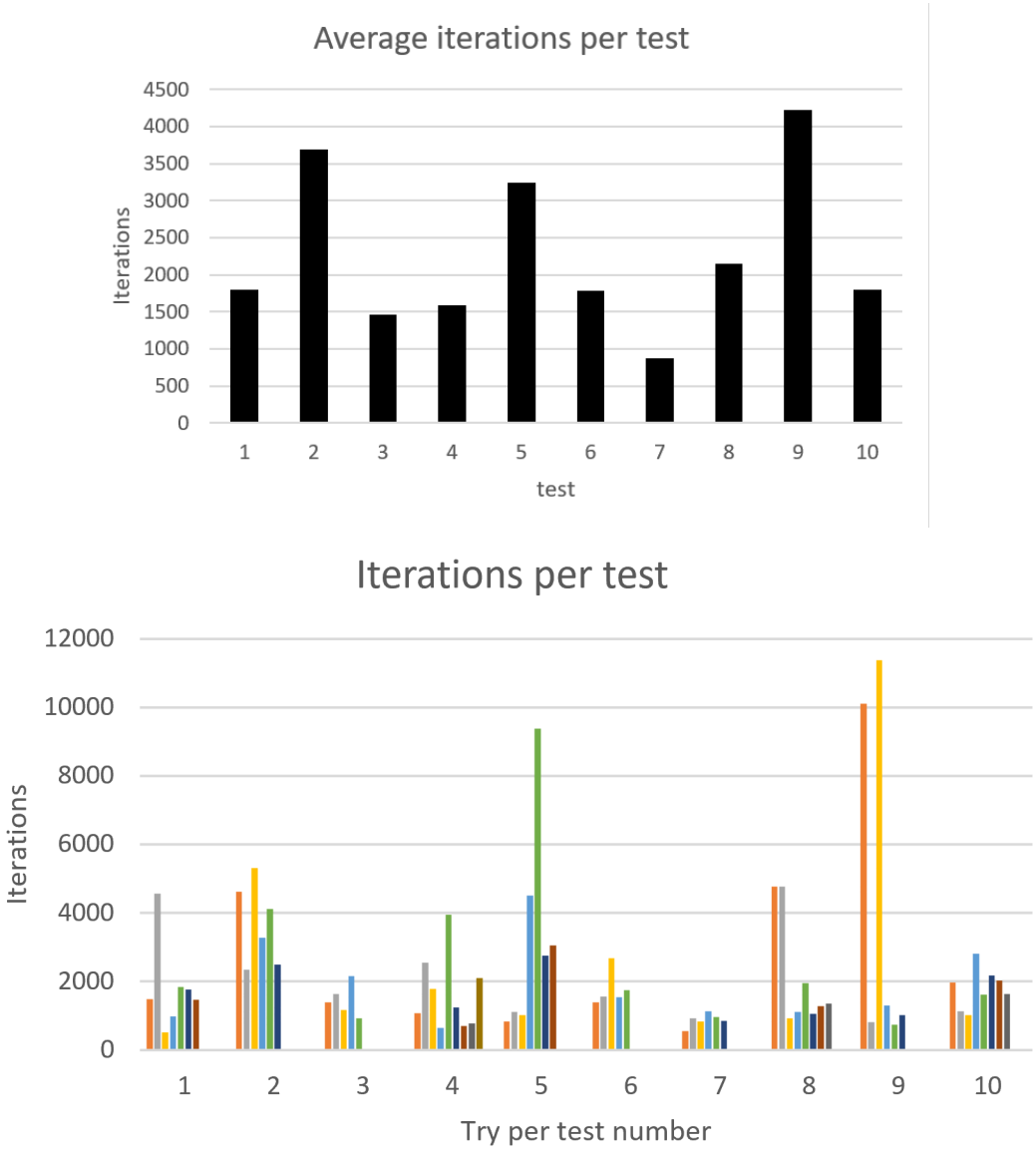

Fig. 7. Results for 37 successful tests run when $n = 5$

3.3.4. Potential Enhancements

Further experimentation could aid in discovering superior hyper-parameter values, potentially enhancing the algorithm's performance—for instance, enabling it to identify solutions in fewer iterations, accelerating convergence rates, and improving the reliability of satisfactory solution identification.

Moreover, the hyper-parameters ought to be tailored more adeptly to align with the problem's specific attributes. For instance, when five images are supplied as input, the choice of hyper-parameters should differ from the scenario in which six images are provided.

The fitness function could be improved by the selection of more precise and pertinent penalty factors, consequently producing results that are more aligned with the program stakeholders' objectives.

The inclusion of diagnostic tests to ascertain the nature of the provided problem could also be beneficial. This could influence the initial positions of the particles and potentially enhance the

performance of the algorithm.

The program might also benefit from the inclusion of dynamic parameters for the computation of the desired solution fitness and particle positions across various dimensions. In the current program iteration, certain distinct values yield identical results due to the image segments aligning to the closest pixel slot on the canvas. Such equivalent solutions should be assigned the same fitness value. A simplification of the program to operate with a precision of one pixel could markedly enhance the algorithm's efficiency.

Finally, it could be beneficial for the program to detect instances when it starts converging towards a non-satisfactory solution, providing an opportunity for timely intervention.

4. Conclusions

Initially, an extensive literature review was conducted to provide an overview and analysis of various evolutionary optimization algorithms. Among these algorithms, the Particle Swarm Optimization (PSO) algorithm was selected for an in-depth examination. This choice was informed by the algorithm's versatility and robustness in solving complex optimization problems.

Furthermore, the thesis presented an investigation into the 2D bin packing problem and established its relevance and similarity to the image placement problem. This correlation was essential in adopting methodologies that are effective in solving the 2D bin packing problem for the image placement task.

Notably, the Particle Swarm Optimization algorithm was implemented and applied to find solutions for the image placement problem. This involved the development of a test generator to facilitate the creation of various scenarios for assessing the algorithm's performance. Through rigorous testing, the algorithm was evaluated for its effectiveness in solving the image placement problem.

The tests and subsequent analysis provided invaluable insights into the performance characteristics of the algorithm. These insights have been instrumental in identifying areas where the algorithm could be improved. Some potential avenues for enhancement include tuning the algorithm's parameters for different problem instances and integrating the algorithm with other heuristics to create hybrid solutions.

For future work, it is recommended to explore additional optimization algorithms and to experiment with hybrid approaches. Moreover, the incorporation of machine learning techniques might prove beneficial in dynamically adapting the algorithm's parameters based on problem characteristics. Lastly, the study could be extended to address variations of the image placement problem, such as allowing image rotations or considering the aesthetic quality of the generated collages.

The accomplishments of this thesis not only contribute to the understanding of evolutionary optimization algorithms, especially Particle Swarm Optimization, but also open avenues for further research and improvements in solving complex optimization problems such as the image placement problem.

Literature

[AB09] Sanjeev Arora ir Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[DS04] Marco Dorigo ir Thomas Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[Eng07] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, 2007.

[Gol10] Oded Goldreich. *P, NP, and NP-Completeness: The basics of computational complexity*. Cambridge University Press, 2010.

[GR04] Daoxiong Gong ir Xiaogang Ruan. A hybrid approach of GA and ACO for TSP. In *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788)*, t. 3, p. 2068–2072. IEEE, 2004.

[LM90] G. Laporte ir S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 1990.

[Pol08] R. Poli. Analysis of the publications on the applications of particle swarm optimisation, 2008.

[RG11] N.M. Razali ir J. Geraghty. Genetic algorithm performance with different selection strategies in solving TSP. In 2011.

[Sim13] Dan Simon. Evolutionary optimization algorithms, 2013.

[WQZ15] J. Wang, Y. Qi, and J. Zhang. Research on pso algorithms for the rectangular packing problem. *Tianjin Key Laboratory of High Speed Cutting and Precision Machining, Tianjin University of Technology and Education*, 2015.

Evolutionary optimisation algorithms: Particle Swarm Optimisation alogorithm

Kazimieras Jasaitis

Abstract

In contemporary society, technology has become integral to everyday life, with computers executing a plethora of tasks with deterministic precision. However, there exists a category of complex, computationally intensive problems for which deterministic algorithms are insufficient. One such problem is collage creation, which involves arranging multiple images on a fixed-size canvas without overlaps or unoccupied space, and is classified as an NP-Hard problem. This thesis explores the application of Particle Swarm Optimization (PSO), an evolutionary optimization algorithm, in solving the collage creation problem. The PSO algorithm is analyzed in-depth, and its application to the problem is investigated through the development of a unit test generator and the execution of performance tests. The unit test generator segments an image into n pieces, and the PSO algorithm is applied to find optimal or near-optimal placements of these pieces on a canvas. The performance of the PSO algorithm is evaluated and analyzed critically, considering various metrics. The study concludes with insights drawn from the results and recommendations for potential enhancements to the PSO algorithm for future research. This work contributes to the understanding of evolutionary optimization algorithms and presents a practical approach to solving complex optimization problems such as collage creation.

Keywords: Particle Swarm Optimisation, optimization, evolutionary methods, PSO.

Additional files

Additional file no. 1

Program files.

https://github.com/KazimierasJasaitis/PSO_for_image_placement