## 

## **Software Engineering Department**

## **Optimization of Two-Dimensional Polygon Packing Using Genetic Algorithms and Advanced Heuristic Methods**

### **Capstone Project Phase B** Software Engineering (Course 61999) Karmiel – Sep 2024

### **Author** Coral Harel 316328749

### **Supervisor** Dr. Miri Weiss Cohen

# 

## 

## **Table of Contents**

[**Abstract** 2](#_tvsudgw90s2a)

[**Introduction** 2](#_degqbnb5pr5h)

[**Literature Review 3**](#_kg6r1ukwcl2h)

[The Two-Dimensional Polygon Packing Problem 3](#_nt1y5158zbu)

[The Complexity of NP-Hard Problems 3](#_4yrf3nl2d3wd)

[Genetic Algorithms in Optimization 3](#_jnuko55iocqk)

[Heuristic Methods for Polygon Packing 4](#_w7tqlpux15ku)

[Hybrid Approaches in Optimization 4](#_6xfx23pvjy1j)

[**Genetic Algorithm 5**](#_5y91l53ykdkk)

[Stages of Natural Selection 5](#_ovbs60yybahz)

[Crossover and Mutation Design 5](#_7wyd68jqz3er)

[Unique Considerations for Polygon Packing 6](#_rj5sc1by5rg9)

[The Bottom-Left-Fill Heuristic 7](#_ivtfpxkluz9o)

[Hybridization with Other Optimization Techniques 7](#_fu26jmpim1e5)

[**Algorithm Implementation 8**](#_ovxi7dhkkkoh)

[Generating Random Polygons 8](#_udeosrcx9th5)

[Sorting Polygons Using DJD Heuristic Methods 8](#_z24e57tb90nd)

[Placement of Polygons 9](#_o7hhnhvwsqhm)

[Fitness Calculation 9](#_637d8njxxdcx)

[Mutation and Crossover 9](#_w1hbkz67xhb5)

[Penalization for Overlaps 10](#_47lsaiyutfe8)

[Evolutionary Process 10](#_vr08vkoc4bv)

[**Experiments and Results 11**](#_cn06yqsf6q5g)

[**Analysis of Results 14**](#_fes7aa62jgpi)

[**Future Improvements 15**](#_w64go4gotbq5)

**Conclusion 15**

[**Bibliography 16**](#_36q4de3g72lu)

## 

## 

## 

## 

### **Abstract**

This research addresses the two-dimensional polygon packing problem, an NP-hard problem, by utilizing genetic algorithms combined with heuristic placement strategies. The genetic algorithm evolves a population of potential solutions across multiple generations using natural selection, crossover, and mutation operations. Key strategies such as the *Center-Focus* method are employed to optimize the placement of polygons, ensuring efficient use of space within predefined bins while minimizing overlaps. The fitness function evaluates each solution based on both space utilization and proximity to the center of the bin, prioritizing compact and balanced arrangements.

The algorithm incorporates various mutation techniques to introduce diversity in polygon orientations and positions, enhancing the search for optimal configurations. Experiments conducted on different polygon sets demonstrate that the genetic algorithm, when integrated with heuristic methods, achieves high-quality packing solutions, balancing computational efficiency with optimal space usage. This hybrid approach offers a robust framework for solving complex polygon packing problems in practical applications.

### **Introduction**

The task of arranging geometric shapes, such as polygons, within a constrained area is a well-known problem in geometric optimization, commonly referred to as the two-dimensional polygon packing problem. This problem has significant practical relevance across various industries, including manufacturing, logistics, and material cutting, where efficient space utilization is crucial for reducing waste and improving resource management. Unlike simpler packing problems involving rectangles or regular shapes, polygon packing is considerably more complex due to the irregularity of the polygons, which vary in size, shape, and orientation.

This research aims to address the polygon packing problem by employing genetic algorithms (GAs), which are well-suited for solving NP-hard optimization problems due to their ability to explore large solution spaces efficiently. The genetic algorithm used in this study evolves a population of solutions through selection, crossover, and mutation, ensuring diversity and allowing for exploration of different polygon arrangements over successive generations. A key feature of this approach is the incorporation of heuristic strategies, such as the Bottom-Left and Center-Focus method, to guide the placement of polygons and enhance space utilization.

The combination of genetic algorithms with heuristic methods provides a powerful framework for solving the polygon packing problem. By leveraging the global search capabilities of GAs and the local optimization provided by heuristic techniques, this project demonstrates an effective method for minimizing overlaps, improving space utilization, and achieving high-quality packing solutions. The focus is on balancing computational efficiency with solution quality, ensuring the approach is practical for real-world applications.

**Literature Review**

### **The Two-Dimensional Polygon Packing Problem**

The two-dimensional polygon packing problem (2DPP) deals with arranging polygons of various shapes, sizes, and orientations within a confined space or bin while minimizing unused space and avoiding overlaps. Unlike the simpler rectangular bin packing problems, polygon packing is particularly challenging because of the irregularity and variability in polygon shapes. Each polygon can differ in the number of vertices, angles, and dimensions, making it harder to predict how polygons will fit together in a given space.

Practical applications of this problem are common in manufacturing, logistics, and material cutting, where the goal is to optimize the arrangement of materials to reduce waste. For instance, in the manufacturing industry, companies aim to cut shapes from raw materials (like fabric or metal sheets) as efficiently as possible. Similarly, in logistics, polygons can represent irregularly shaped objects that need to be packed efficiently into shipping containers. In all these applications, minimizing wasted space can translate to cost savings and improved resource utilization .

### **The Complexity of NP-Hard Problems**

The polygon packing problem is NP-hard, which means that the complexity of finding an exact solution grows exponentially as the number of polygons increases. In other words, there is no known algorithm that can solve all instances of this problem in polynomial time. This complexity arises from the need to consider both the placement and orientation of each polygon while ensuring no overlaps occur.

Given this, heuristic and metaheuristic algorithms are often used as an alternative to exact methods. These algorithms, such as genetic algorithms (GAs), simulated annealing, and tabu search, do not guarantee an optimal solution but can find high-quality solutions in a feasible amount of time. The challenge is to balance computational efficiency (how fast the solution is found) and solution quality (how close the solution is to the optimal arrangement). Hybrid methods that combine different approaches are particularly useful because they can exploit the strengths of multiple optimization techniques .

### **Genetic Algorithms in Optimization**

**Genetic algorithms (GAs)** are evolutionary optimization techniques inspired by natural selection. GAs evolve a population of solutions over multiple generations, with each solution represented as a **chromosome** (or individual) within the population. The core operations in GAs include:

* **Selection**: Solutions with better performance (higher fitness) are selected for reproduction.
* **Crossover**: Parts of two solutions are combined to create new offspring.
* **Mutation**: Random changes are applied to some solutions to introduce variation.

In the context of polygon packing, each individual solution represents a specific arrangement of polygons within a bin. The fitness function evaluates each solution by measuring its space utilization and ensuring no overlaps occur. Over successive generations, weaker solutions are eliminated, and stronger solutions are refined, allowing the GA to explore a wide range of possibilities and escape local optima.

One key advantage of GAs in solving NP-hard problems is their ability to explore complex solution spaces and avoid getting stuck in suboptimal solutions. By evolving multiple solutions simultaneously, GAs maintain population diversity and balance between global search and local refinement .

### **Heuristic Methods for Polygon Packing**

Heuristic methods are designed to provide quick, approximate solutions to complex problems. In polygon packing, one well-known heuristic is Bottom-Left-Fill (BLF), which systematically places polygons in the bottom-left corner of the bin and fills the remaining space in an orderly manner. This heuristic reduces gaps between polygons and minimizes wasted space. While it doesn’t guarantee an optimal solution, it is computationally efficient and serves as an excellent foundation for more advanced algorithms.

When combined with GAs, heuristics like Bottom-Left-Fill act as a form of local optimization, refining solutions generated by the GA. The hybridization of these techniques allows the GA to benefit from the global search capabilities while utilizing heuristic methods for fine-tuning the placement of polygons. This combination is particularly useful in solving polygon packing problems because it enables faster convergence toward high-quality solutions .

### **Hybrid Approaches in Optimization**

A key advancement in solving the polygon packing problem is the use of hybrid optimization approaches. These methods combine the strengths of different optimization techniques to achieve better results. For example, hybridizing genetic algorithms with heuristics like Bottom-Left-Fill allows for a balance between exploring the solution space (via GA) and refining solutions locally (via the heuristic). This hybrid approach has been shown to significantly improve performance, especially when handling irregularly shaped polygons, where simple placement strategies often fail .

In conclusion, the integration of genetic algorithms, heuristic methods, and hybrid optimization techniques offers a powerful approach to solving the complex, NP-hard problem of polygon packing. These methods allow for efficient exploration of potential solutions while ensuring that the solutions are feasible and close to optimal in terms of space utilization.

## 

## **Genetic Algorithm**

### **Stages of Natural Selection**

The genetic algorithm in this research mirrors natural selection, progressing through several important stages:

* **Initial Population Creation**: The algorithm starts by generating a diverse set of random solutions. Each solution (chromosome) represents a specific arrangement of polygons within the bin. Randomness at this stage ensures a broad exploration of potential solutions.
* **Fitness Calculation**: The fitness function evaluates each solution based on key criteria such as space utilization (how efficiently the polygons are packed) and the absence of overlaps. Higher fitness scores correspond to better solutions, with more efficient use of space and fewer overlaps.
* **Selection**: Solutions with higher fitness scores are more likely to be selected for reproduction, mimicking natural selection. Selection methods such as roulette wheel or tournament selection give preference to stronger solutions, increasing the probability of passing their traits (polygon arrangements) to the next generation.
* **Crossover**: In this stage, two selected parent solutions are combined to create new offspring. The crossover process swaps sections of the polygon layout between parents, mixing their traits to create potentially better offspring. This promotes diversity in the population, enabling the exploration of new solutions.
* **Mutation**: A small percentage of the population undergoes mutations, introducing random changes to the polygon layout, such as rotating polygons, shifting their position within the bin, or reordering their placement. This helps maintain variability and prevents the algorithm from stagnating at suboptimal solutions.
* **Repetition**: The cycle of selection, crossover, and mutation repeats across multiple generations. The population evolves, improving over time, until a stopping criterion is met (such as reaching a certain number of generations or a solution of sufficient fitness).

### **Crossover and Mutation Design**

#### **Crossover**

Crossover is a key operation that combines traits from two parent solutions to generate offspring. In polygon packing, this involves exchanging sections of the layout between parents. The offspring inherits parts of both layouts, which may lead to more efficient packing configurations.

Different crossover methods offer varying benefits:

* **Single-Point Crossover**: A single crossover point is selected, and polygons are swapped before and after this point. While simple, it may not always produce optimal solutions.
* **Multi-Point Crossover**: Multiple crossover points are used, allowing for more complex exchanges and the combination of diverse traits from both parents.
* **Uniform Crossover**: Each polygon is randomly chosen from one of the two parents, introducing high variability. This method increases diversity but may disrupt effective layouts.

Each method has trade-offs: single- and multi-point crossovers provide more control but may be too rigid, while uniform crossover introduces flexibility but can result in excessive randomness.

#### **Mutation**

Mutation introduces random changes to maintain diversity within the population and explore new possibilities. In polygon packing, common mutations include:

* **Polygon Rotation**: Rotating polygons by small, random angles to explore new configurations that may fit better within the bin.
* **Position Shifting**: Slightly moving polygons horizontally or vertically within the bin to refine their placement.
* **Reordering**: Changing the order in which polygons are placed to see if alternative arrangements improve space utilization.

Mutations are applied sparingly to avoid disrupting strong solutions while ensuring enough diversity to avoid premature convergence.

### **Unique Considerations for Polygon Packing**

Polygon packing presents specific challenges that require careful handling by the genetic algorithm:

* **Shape Irregularity**: Polygons vary in size, shape, and the number of vertices, making it difficult to predict how they will fit together.
* **Rotation and Orientation**: Slight changes in the orientation of polygons, especially irregular ones, can significantly impact how efficiently they fit in the bin.
* **Overlap Avoidance**: Unlike simpler packing problems, polygon packing requires complex calculations to prevent overlaps, which must account for both shape and position.
* **Bin Boundaries**: Ensuring that polygons stay within the bin boundaries requires additional checks, especially during mutation and crossover.

Addressing these challenges ensures that the genetic algorithm produces feasible and efficient solutions that maximize space usage while avoiding overlaps.

### 

### **The Bottom-Left-Fill Heuristic**

The Bottom-Left-Fill heuristic is a simple yet powerful method for organizing polygons within a bin. It works by placing polygons as far into the bottom-left corner as possible and then filling the remaining space systematically.

#### **Implementation within the Genetic Algorithm:**

* **Initial Placement**: During mutation, polygons are placed according to the Bottom-Left-Fill rule, ensuring efficient initial placement relative to previously placed polygons.
* **Post-Crossover Adjustment**: After crossover, the heuristic can be applied to rearrange polygons for better space utilization, compensating for any inefficiencies introduced during recombination.
* **Local Optimization**: As solutions improve over generations, the heuristic serves as a fine-tuning tool to eliminate gaps and maximize space usage.

By combining the global search capabilities of the genetic algorithm with the local optimization strengths of Bottom-Left-Fill, the algorithm produces more efficient, organized solutions.

#### **Advantages:**

* **Simplicity**: Bottom-Left-Fill is computationally efficient and easy to implement.
* **Effective Space Utilization**: It minimizes gaps between polygons, leading to better space usage.
* **Overlap Avoidance**: The heuristic helps naturally avoid overlaps through its structured placement approach.

#### **Limitations:**

* **Limited Flexibility**: It struggles with irregular polygons and situations where non-standard rotations are required.
* **Suboptimal for Complex Arrangements**: While effective for quick solutions, it may not find the global optimal arrangement.

### **Hybridization with Other Optimization Techniques**

While Bottom-Left-Fill is effective, it works best when combined with other optimization techniques to handle more complex scenarios:

* **Simulated Annealing**: This technique introduces controlled randomness to explore a broader range of solutions, helping avoid local optima.
* **Tabu Search**: Prevents the algorithm from revisiting suboptimal solutions, encouraging exploration of new layouts.
* **Penalization Techniques**: Dynamic penalties for overlaps and inefficient space usage guide the algorithm toward better solutions, especially as it approaches convergence.

Hybridizing these techniques with the Bottom-Left-Fill heuristic results in a more robust framework capable of solving even complex polygon packing challenges.

## **Algorithm Implementation**

The algorithm developed in this research is based on the combination of genetic algorithms and heuristic methods, specifically designed to address the challenges of two-dimensional polygon packing. The implementation follows a structured process that allows for the efficient placement of polygons in bins while minimizing overlaps and maximizing space utilization. Below is a detailed description of each part of the algorithm:

### **Generating Random Polygons**

To ensure diversity in the initial population of the genetic algorithm, random polygons are generated with varying shapes and sizes. This process is crucial for the algorithm’s ability to explore a wide range of potential solutions. The generation process involves the following key elements:

1. **Number of Vertices**: Each polygon is assigned a random number of vertices, typically ranging between 3 and 8. This introduces a mixture of simple shapes (like triangles and quadrilaterals) and more complex shapes (such as hexagons or octagons), ensuring a varied population of polygons.
2. **Radius and Angle Calculation**: The vertices of the polygon are randomly placed around a central point, with varying radii to create polygons of different sizes. The angles between the vertices are randomized to generate irregular, non-symmetrical polygons, further enhancing the complexity of the shapes.
3. **Irregularity and Spikiness**: These two parameters control the randomness of the polygon’s shape. Irregularity affects the variation in the angles between vertices, while spikiness controls how much the edges of the polygon vary in length. These factors allow for the creation of highly irregular polygons that challenge the packing algorithm to find the most efficient arrangement.
4. **Random Rotation**: Each polygon is assigned a random initial rotation, adding another layer of complexity to the packing process. This forces the algorithm to consider both the position and orientation of the polygons during the optimization process.

### **Sorting Polygons Using DJD Heuristic Methods**

Before attempting to pack the polygons into bins, they are sorted using the **DJD heuristic**, which orders polygons by their area and perimeter. This heuristic prioritizes larger and more complex polygons for early placement, which helps reduce wasted space and simplifies the subsequent placement of smaller polygons. This approach is particularly effective in minimizing gaps that could arise if smaller polygons are placed first.

### **Placement of Polygons**

The core challenge of polygon packing is finding the most efficient way to place polygons inside the bins while preventing overlaps and keeping all polygons within the bin boundaries. The algorithm utilizes the Bottom-Left-Fill heuristic to achieve this. The heuristic works as follows:

1. **Initial Placement Strategy**: Polygons are positioned as close as possible to the bottom-left corner of the bin. This ensures that polygons are placed in a compact arrangement, minimizing the gaps between them. As each polygon is placed, the algorithm attempts to find the most efficient orientation by rotating and translating the polygon within the bin.
2. **Rotation and Translation**: For each polygon, multiple rotations are attempted at fixed intervals (e.g., every 15 degrees), followed by small translations within the bin. This allows the algorithm to explore different orientations and positions, seeking the one that best fits the current state of the bin without overlaps.
3. **Overlap Prevention**: Overlaps between polygons are a significant concern in polygon packing. The algorithm includes a sophisticated overlap detection mechanism that checks whether a newly placed polygon overlaps with any previously placed polygons. If an overlap is detected, the polygon is repositioned or rotated until a valid placement is found.
4. **Boundary Checks**: In addition to avoiding overlaps, the algorithm ensures that all polygons remain within the bin’s boundaries. If a polygon extends beyond the bin, it is either repositioned or rejected. This constraint ensures that no part of a polygon falls outside the bin.

### **Fitness Calculation**

Once all polygons have been placed within the bin, the quality of the solution is evaluated using a fitness function. The fitness function considers two key factors:

1. **Space Utilization**: The primary goal is to maximize the use of available space within the bin. The algorithm calculates the total area occupied by the polygons and compares it to the bin’s total area. A higher ratio of polygon area to bin area indicates a more efficient packing arrangement.
2. **Proximity to the Bin Center**: In some cases, proximity to the center of the bin is also considered in the fitness function. This ensures that polygons are not only packed efficiently but also positioned in a balanced and compact manner, which can be useful for applications requiring optimal load distribution.

### **Mutation and Crossover**

The genetic algorithm includes standard operations of mutation and crossover to introduce variability and refine the solutions across generations:

1. **Crossover**: The crossover operation combines parts of two parent solutions to create a new solution. In the context of polygon packing, this involves swapping sections of the layout between two parent arrangements. The offspring inherits characteristics from both parents, allowing for the exploration of new packing configurations.
2. **Mutation**: Mutation introduces small, random changes to the polygons’ positions or orientations within the bin. The mutation process typically involves rotating a polygon by a small random angle or slightly shifting its position. This randomness helps prevent the algorithm from getting stuck in local optima, ensuring that a broader range of solutions is explored.
3. **Dynamic Mutation Rates**: To improve the algorithm's performance, mutation rates are adjusted dynamically throughout the evolutionary process. In early generations, higher mutation rates encourage exploration, while in later generations, mutation rates are reduced to focus on refining the most promising solutions.

### **Penalization for Overlaps**

To further ensure valid packing configurations, the algorithm employs a penalization mechanism. If a solution contains overlapping polygons, a penalty is applied to the fitness score. This incentivizes the algorithm to favor solutions where all polygons are placed without overlaps, effectively guiding the optimization process toward feasible solutions.

### **Evolutionary Process**

The algorithm iterates over multiple generations, with each generation representing a new population of solutions. Throughout this process:

* **Selection**: Individuals (polygon arrangements) with higher fitness scores are more likely to be selected for reproduction in the next generation.
* **Recombination and Mutation**: New solutions are generated via crossover and mutation, with a small percentage of the population undergoing random changes to ensure diversity.
* **Convergence**: Over time, the population converges toward high-quality solutions, where polygons are efficiently packed with minimal overlap and maximum space utilization.

## 

## 

## 

## 

## **Experiments and Results**

### **Experiment 1: Initial Population without Improvements**

In this experiment, the algorithm was run with randomly generated polygons placed in the bins without any optimization techniques. This served as a baseline for comparing later improvements.

#### **Methodology:**

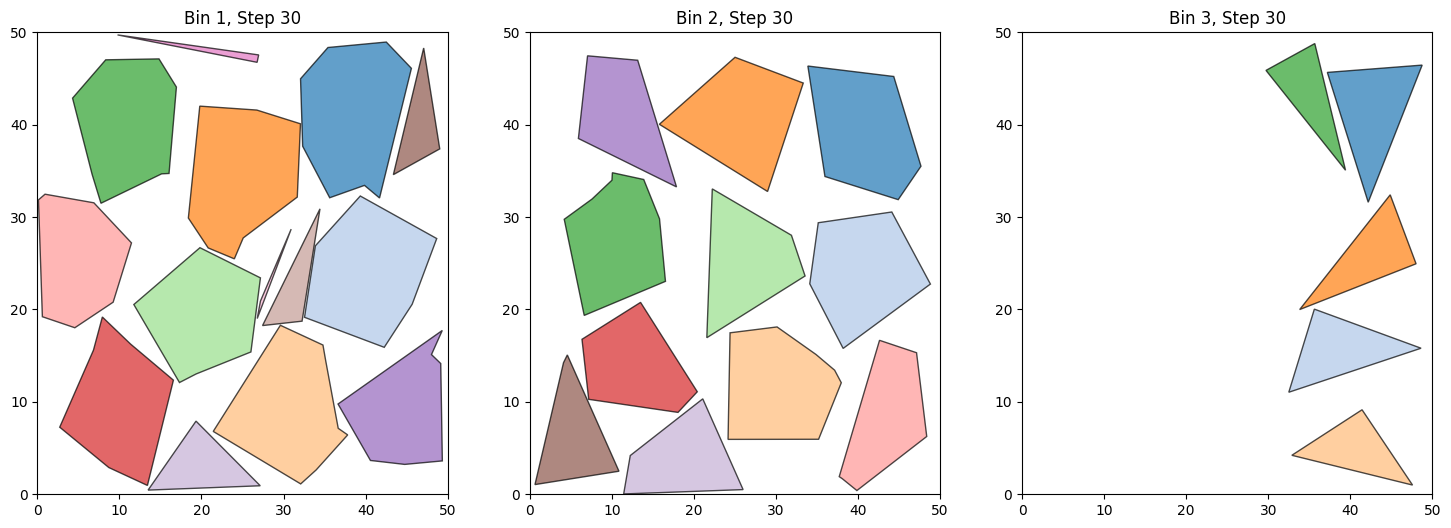
* **Random Placement**: Polygons were placed randomly in the bin, with no strategy for preventing overlaps or optimizing space utilization.
* **Evaluation**: Space utilization and the amount of overlap between polygons were the key metrics used.

#### **Results:**

* **Low Space Utilization**: The arrangement left large gaps, resulting in inefficient use of the bin area.
* **Low Fitness Scores**: Fitness scores were low due to overlaps and poor space utilization.

#### **Conclusion:**

Random placement alone was insufficient for effective polygon packing. The results demonstrated the need for more structured placement strategies to improve both space utilization and solution validity.



### 

### 

### 

### 

### **Experiment 2: Applying the Bottom-Left-Fill Heuristic**

In the second experiment, the Bottom-Left-Fill heuristic was applied to organize polygon placement by systematically pushing them toward the bottom-left corner of the bin.

#### **Methodology:**

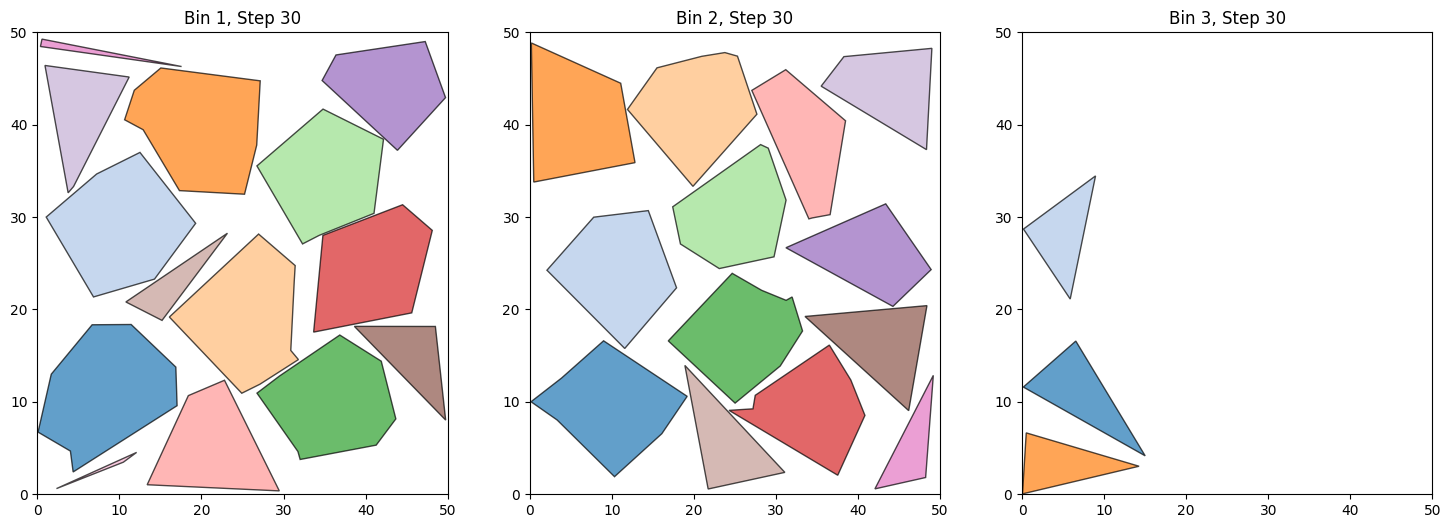
* **Heuristic Placement**: Polygons were rotated and placed in an orderly manner, with larger polygons placed first to optimize space.
* **Overlap Prevention**: Basic checks were introduced to avoid overlaps during placement.

#### **Results:**

* **Reduction in Overlaps**: The heuristic significantly reduced overlaps by placing polygons more tightly and systematically.
* **Improved Space Utilization**: The bin space was used more efficiently, with fewer large gaps between polygons.
* **Higher Fitness Scores**: Fitness scores improved due to better space utilization and reduced overlaps.

#### **Conclusion:**

The Bottom-Left-Fill heuristic proved effective in organizing the placement of polygons, leading to better space utilization and more valid solutions compared to random placement.



### 

### 

### 

### 

### 

### **Experiment 3: Combining Hybrid Techniques**

In the final experiment, additional optimization techniques were combined with the Bottom-Left-Fill heuristic, including penalization for overlaps and dynamic mutation rates.

#### **Methodology:**

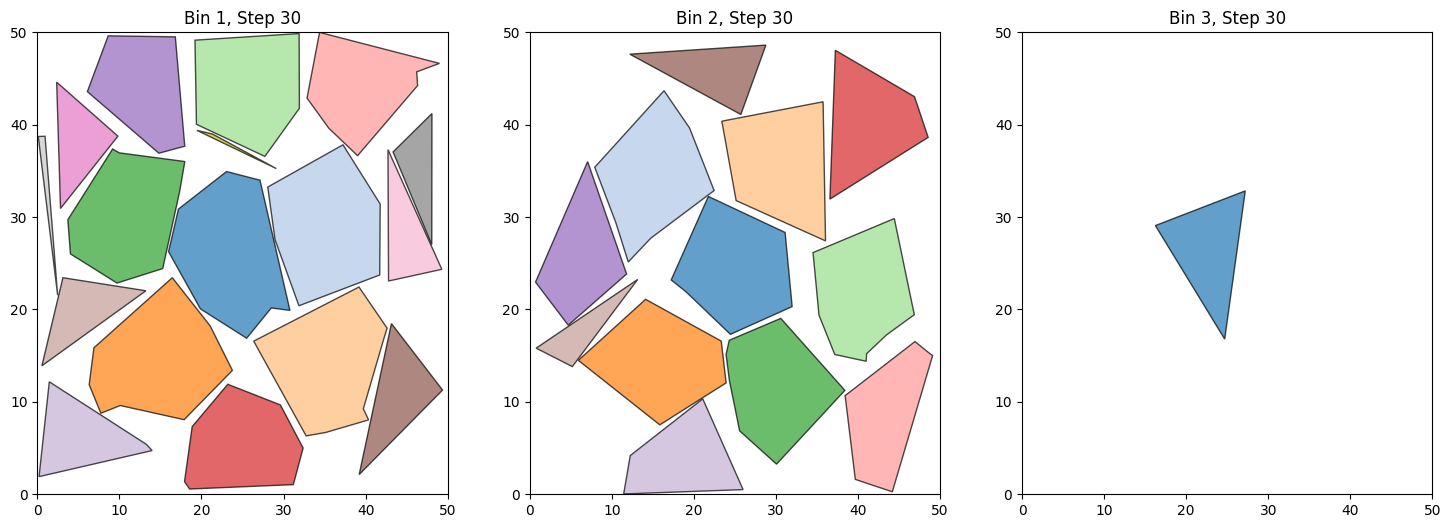
* **Penalization for Overlaps**: Overlapping polygons were penalized in the fitness function to discourage invalid solutions.
* **Dynamic Mutation**: Mutation rates were adjusted dynamically—higher in the early stages for exploration, and lower in later stages for fine-tuning.

#### **Results:**

* **Near-Optimal Solutions**: The hybrid techniques allowed the algorithm to achieve near-optimal packing solutions, with minimal overlaps and wasted space.
* **Maximized Space Utilization**: The bin area was packed more efficiently than in previous experiments.
* **Highest Fitness Scores**: Fitness scores peaked in this experiment, reflecting both efficient packing and valid, overlap-free solutions.

#### **Conclusion:**

The combination of heuristics, penalization, and dynamic mutation rates resulted in high-quality solutions. This hybrid approach effectively balanced local optimization with global search, achieving the best results in terms of space utilization and solution validity.



## 

## 

## **Analysis of Results**

### **Impact of Hybrid Methods**

The results of the experiments clearly demonstrate the benefits of using hybrid methods that combine genetic algorithms with heuristic techniques. The Bottom-Left-Fill heuristic significantly improved the initial placement of polygons by systematically placing them in the bottom-left corner of the bin, leading to more compact and efficient layouts. This heuristic helped reduce the number of large gaps and minimized overlaps, compared to the random placement strategy.

The introduction of penalization techniques for overlaps proved to be essential in guiding the genetic algorithm toward valid solutions. By penalizing overlapping polygons, the algorithm favored solutions that not only maximized space utilization but also adhered to the constraint of no overlaps. This allowed the algorithm to focus on refining the placement of polygons in a way that both optimized space and maintained validity.

The use of dynamic mutation rates played a crucial role in maintaining a balance between exploration and exploitation. In the early stages of the genetic algorithm, a higher mutation rate encouraged exploration of different packing configurations, helping the algorithm avoid getting stuck in local optima. As the algorithm progressed, the mutation rate was reduced, allowing for more precise adjustments in the final generations, which improved the overall solution quality.

The combination of local optimization (using the heuristic) and global search (through the genetic algorithm) provided a powerful synergy. The heuristic offered an efficient starting point by arranging polygons in a structured manner, while the genetic algorithm enhanced these arrangements through crossover and mutation across generations. This hybrid approach consistently delivered solutions that were both space-efficient and overlap-free.

Overall, the hybrid methods proved effective in achieving high-quality solutions. The interplay between heuristic-driven placement and the genetic algorithm’s exploration capabilities allowed the algorithm to achieve near-optimal packing configurations. The results show that this approach is suitable for real-world applications, where minimizing space wastage and preventing overlaps are critical.

## 

## 

## 

## 

## **Future Improvements**

Several potential improvements can be explored to further enhance the performance of the algorithm:

1. **Enhanced Fitness Function**: The current fitness function focuses on space utilization and overlap prevention. Future improvements could introduce additional factors, such as polygon proximity to the bin’s center or edge, or penalties for polygons placed far from the optimal packing configuration.
2. **Population Diversity**: Maintaining diversity in the population is critical to avoiding premature convergence. Future work could explore strategies for introducing new genetic material into the population, such as reintroducing mutated individuals or using niche strategies to maintain diverse solutions throughout the optimization process.
3. **Incorporating Advanced Search Techniques**: Techniques like Simulated **Annealing** or **Tabu Search** could be integrated into the genetic algorithm to improve the exploration of the solution space, especially in later generations when the algorithm might get stuck in local optima.
4. **Optimized Crossover and Mutation**: More sophisticated crossover and mutation strategies could be developed to handle complex polygon shapes. For example, mutations could be adapted to the specific geometry of the polygons, allowing for more precise adjustments in orientation and position.
5. **Penalization Adjustments**: Penalization strategies could be dynamically adjusted based on the progress of the algorithm. For example, penalization for overlaps could increase as the algorithm approaches the later stages, ensuring that the final solutions are free of any errors.

## **Conclusion**

This project investigated the application of genetic algorithms (GAs) combined with heuristic methods to address the two-dimensional polygon packing problem, an NP-hard problem with significant real-world relevance. By integrating the Bottom-Left-Fill heuristic into the genetic algorithm framework, substantial improvements were achieved in terms of space utilization and overlap prevention. The experimental results demonstrated that the hybrid approach is capable of producing high-quality solutions within a reasonable computational timeframe.

The work presented here provides a solid foundation for further advancements in the optimization of polygon packing algorithms. Future research could focus on several key areas: refining the fitness function to better capture complex placement constraints, improving population diversity to avoid premature convergence, and incorporating more advanced hybrid techniques such as simulated annealing or tabu search. These enhancements could further improve the algorithm's performance, particularly in handling larger and more complex polygon packing scenarios.

## **Bibliography**

1. **Heuristics for Evolutionary Optimization for the Centered Bin Packing Problem** - Luke de Jeu1 and Anil Yaman1
2. **An effective heuristic for the two-dimensional irregular bin packing problem** - Eunice Lopez, Gabriela Ochoa and Hugo Terashima-Marín- July 2013
3. Holland, J. H., **Adaptation in Natural and Artificial Systems**, University of Michigan Press, 1975.
4. Goldberg, D. E., **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesley, 1989.
5. Smith, T., & Eiben, A. E., **Introduction to Evolutionary Computing**, Springer, 2003.