Evolutionary Algorithms

2025SEM1_KV6014BNN01

Student: Oliver Winhammar (23056061)

Words: 3,213

Date of Submission: 15/01/2026

GitHub Repository:
https://github.com/0larszl0/Cargo-Loading-with-Evolutionary-Algorithms

How Evolutionary Algorithms can be used to solve Cargo Loading Problems

Introduction

One of the earliest instances where packing has been brought to the limelight was made in the endeavour to better understand the constitution and structure of matter, based upon the patterns observed in nature. More specifically in the 17th century, Johannes Kepler founded a conjecture that questions how an arrangement of (many) spheres can maximise the density in a given space (Weaire, 1999). In the effort of searching for structure and order in solid matter, Kepler inadvertently foreshadowed the modern ruling for packing problems; a way to formulate an optimal solution to maximise the total value or availability of something wherein one or more constraints must be adhered to. During the same century another packing related dilemma began to arise when soldiers had to choose the most essential items to carry, a challenge that wasn't formally explored until the early to mid 20th century – the Knapsack Problem (Esicup, 2018)(Mishra, 2024). In the 20th century, the broader field of combinatorial optimisations had begun to grow colinearly to researchers' investigations in optimisation problems and algorithmic solutions (Mishra, 2024). As a result, numerous packing and cutting problems have emerged, varying in complexity and application. This paper, however, will focus specifically on the constituents involved with packing problems and how evolutionary algorithms have begun to and can be used to solve them.

Literature Review

Nature of the Problem

Among the many regular and irregular shapes available–including those that are hexagonal, oval, or triangular— rectangles are most commonly used in everyday packaging and containerisation. This widespread use is because of several factors, including their efficient utilisation of space and their cost-effectiveness during manufacturing (Piber Plastic, 2023). The former is especially true when considering heavy-duty applications including loading cargo onto massive freighters which require a *Bay Plan* to maximise the safety of crew and cargo, as well as the stability of the vessel (VS&B Containers Group, 2021). Consequently, by addressing the problem in this way, it begins to rationalise the key geometric, weight, and loading constraints that this paper must adhere to (Appendix 1.)(Appendix 2.).

Similar Solutions

One paper, by Birgin et al, treats the cargo loading problem as a decision problem which endeavours to ensure that any circle will at most only touch a neighbouring circle: one intersection occurring between any pair of circles. To solve it, a combination of Newton-like minimisation methods and regularised Hessians were utilised. The gradient computed from the minimisation technique informs the algorithm of the magnitude and direction in which a cylinder's centre should move in, that best fits the objective function. This means that, as the gradient approaches zero, the closer the cylinder's centre arrives at a local optima. However, to prevent getting trapped in local optima, the second derivatives found within the Hessian matrix, helps to refine the magnitude and direction made by the gradient so that the result can be pushed toward the global optimum. In practice, this strategy is visualised as the flowcharts illustrated in Fig. 1. Consequently, Birgin et al's solution can be depicted as a somewhat naive population-based optimisation, wherein it determines whether k number of circles can fit in a given container by iterating through N number of different initial guesses.

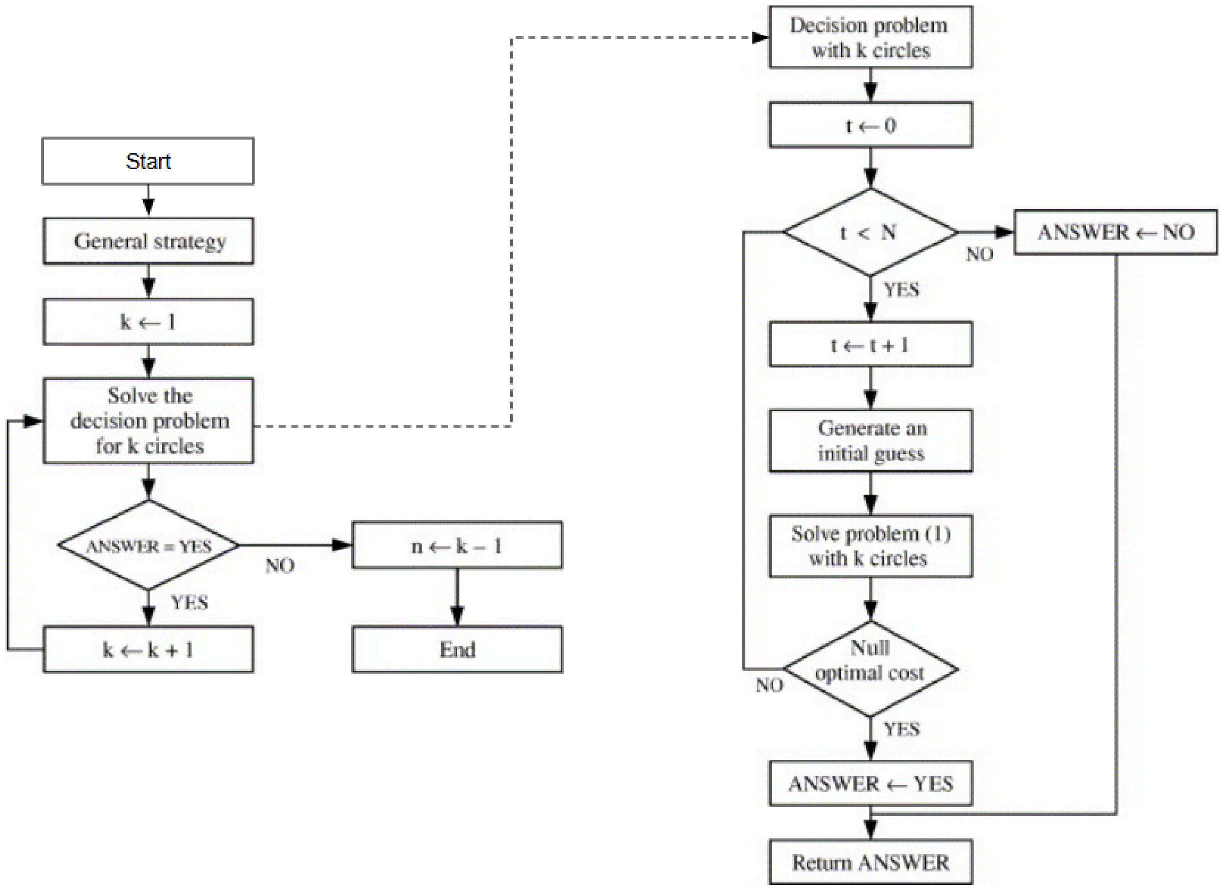


Fig. 1 An amalgamation of two flowcharts: the flowchart on the right determines whether k circles can fit inside the given container, whilst the flowchart on the left utilises the aforementioned one to repeat for differences in k number of circles.

Unlike Birgin et al, a paper by George et al describes several heuristic methods named: Samepack, Wallpack, Stable1, Stable2, Random, alongside a genetic algorithm , to determine the most *stable* solution (Appendix 3.) of circles of varying diameters. The following bullet points outline each of the methods (Appendix 4.).

- **Samepack**: this method focuses on placing circles of similar sizes in close proximity with one another, while applying rules that make the solution somewhat more practical.

- **Wallpack**: this heuristic places circles as close to the edges of the rectangle as possible, whilst also applying the same set of rules as seen in the Samepack method.

- **Stable1**: the method checks whether any feasible location exists in any of the sides in the following order: bottom, left, and right. When no more circles can be packed this way the random method is used to place as many of the remaining circles in the leftover space.

- **Stable2**: the method checks for the best combination of circles alongside the bottom, left, and right sides of the rectangle respectively. Afterwards, the random method is applied in the same way as Stable1.

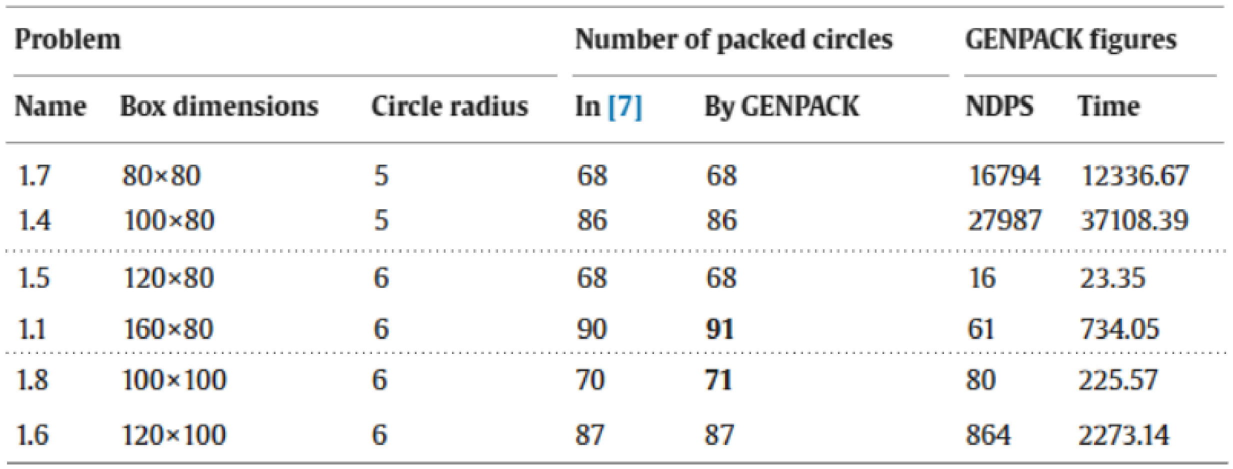
- **Random**: Randomly generates sequences of position numbers and selects the sequence with the best packing configuration.

- **Genetic**: Uses a genetic algorithm approach that uses the random heuristic to fill its initial population. Each string in the population has their fitness measured by how dense their resultant configuration of circles are; this is evaluated by a density formula. The algorithm applies selection, crossover and mutation, to breed a new population until all the strings share the same fitness or if a maximum number of generations has been met.

Analysis

While Birgin et al, describes a very mathematical approach that produces dense packing, something that is important in this project's consideration of weight distribution. Its use of the Hessian matrix, which helps to find an optimal solution, is one drawback because of its computationally expensive and time consuming nature (Chugh, 2025). This is reinforced when observing the method’s performance metrics, where it becomes apparent on how significant time increases in tandem with container dimension. A snippet of several examples are displayed in the following table.

Table 1. A reduced performance table that has spliced rows that shows circles of similar radii getting packed in increasing box dimensions and their effect on compute time.



Problem			Number of packed circles		GENPACK figures	
Name	Box dimensions	Circle radius	In [7]	By GENPACK	NDPS	Time
1.7	80×80	5	68	68	16794	12336.67
1.4	100×80	5	86	86	27987	37108.39
1.5	120×80	6	68	68	16	23.35
1.1	160×80	6	90	**91**	61	734.05
1.8	100×100	6	70	**71**	80	225.57
1.6	120×100	6	87	87	864	2273.14

On the other hand, George et al introduces several heuristic approaches that rely on side and position numbers. While these numbers were devised for the purposes of quantifying *building rules* in order to produce the most physically stable solution (Appendix 3.); the idea has been shown to have decent performance scores alongside genetic algorithm adaptability, which this paper takes a particular focus on. However, during their goal of stability, some of their methods have become too reliant on the sides of the container, which does not align well with this paper’s consideration of weight distribution and the packed items’ centre of mass (COM).

Design

Filtering Literature

After filtering a few different techniques that have been used in the literature (Appendix 5.), there remains one significant approach which will ultimately be adopted; the use of position and side numbers. As described in Appendix 6 the utilisation of both numbering mechanics and starting position are done in a different manner to the original literature portrayed by George et al. A visualisation of such changes can be seen in Fig. 3.

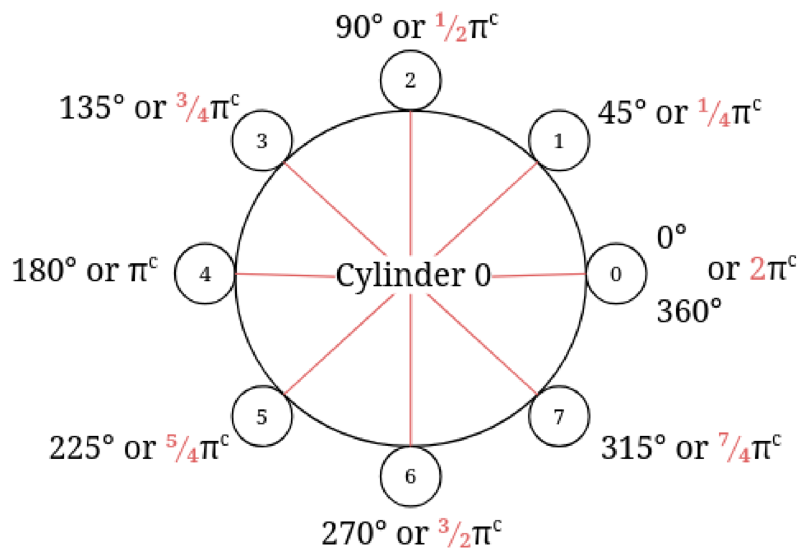


Fig. 3 The new usage of side and position numbers, the figure shows how eight different side positions could be located around a cylinder.

The 8-Sided Decision

As illustrated in Fig. 3, there denotes eight different sides. The choice for this specific number of sides instead of any more or less, is derived from packing cylinders of equal size, but of different quantities, around one central cylinder. Demonstrated in Fig. 4, one can acquire multiple solutions that have reached a perfect COM as a result of using the eight sided method. This 45 degree interval strikes a good balance between performance and execution time, over other intervals and inherently the quantity of side positions (Appendix 7.).

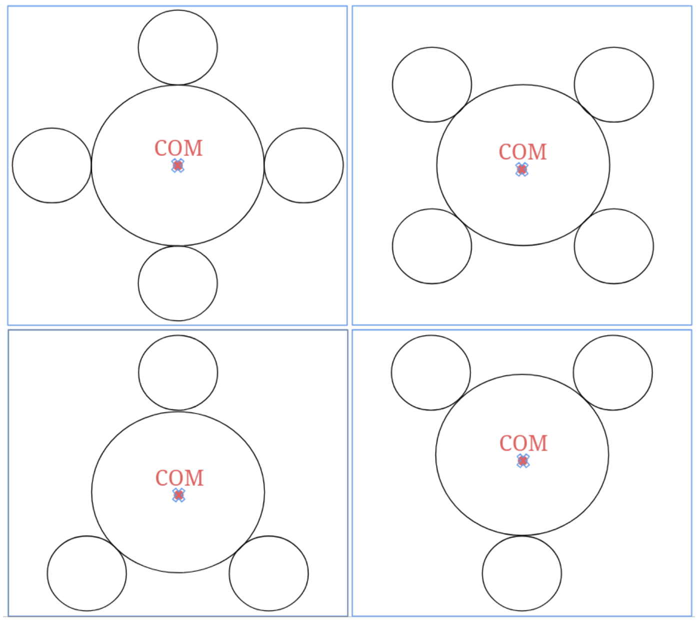


Fig. 4 Examples of perfect solutions of cylinders of equal weight (depicted by the size of their diameter) surrounding a central cylinder., wherein "perfect" is defined as the COM of all collective items to be at the centre of the container.

Objective Function

Realistically, when considering more practical scenarios, not all surrounding cylinders from the first will be of an equal weight, but of a varying kind. As depicted in Fig. 5, the COM is slightly offset from the centre of the container, as a result of the unequal weight between the surrounding cylinders. Consequently, this paper defines the objective function as one that minimises the distance between the group of cylinders' COM and the centre of the container.

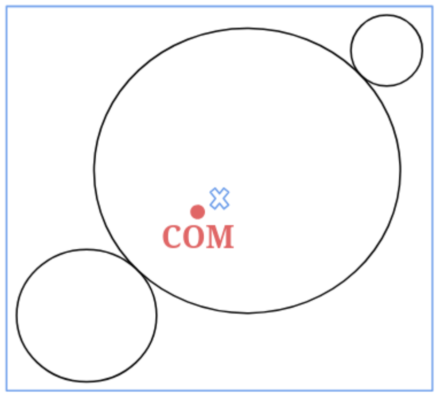


Fig. 5 A solution where cylinders of an unequal weight surround the central cylinder. The cross denotes the centre of the container.

Table 1
Position numbers defined for each circle *i*.

Cylinder *i*	Side *s*							
	s = 0 (right)	s = 1 (top-right)	s = 2 (top)	s = 3 (top-left)	s = 4 (left)	s = 5 (bottom-left)	s = 6 (bottom)	s = 7 (bottom-right)
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31

Side & Position Numbers

The first step to solving the defined objective function is to understand the relationship between position and side numbers within this paper's context. Table 1 defines the first four cylinders that can be used for packing. Each cylinder will have its own appropriate side ranging from one to the maximum number of sides, this paper defaults to eight; refer to Fig. 3 for revision. For this case, the upper range will be denoted as *x*. The position numbers of each cylinder *i* can be found by following a generic function: $f_p = (i * x) + s$, where *i* is the index of the circle, and *s* being the type of side being worked on. Fig. 6 illustrates how these position numbers can be utilised.

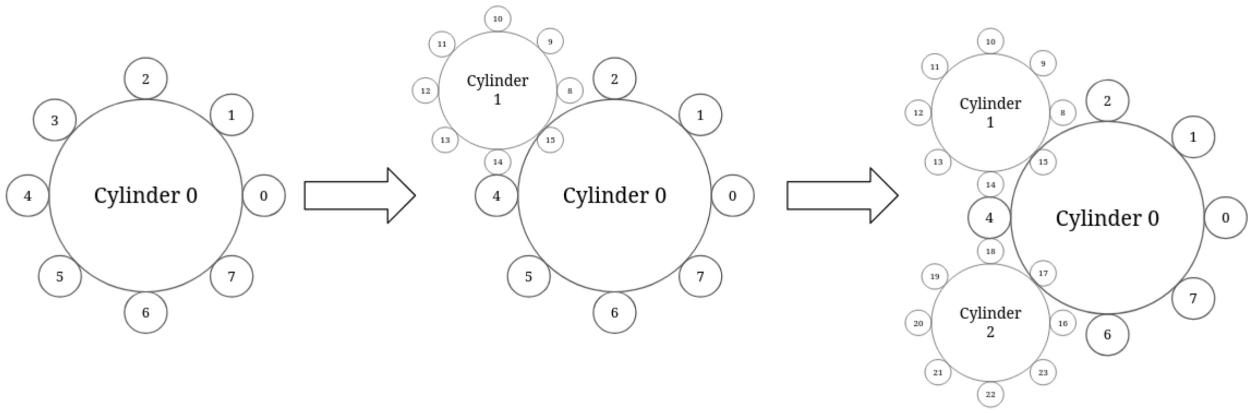


Fig. 6 An example that shows how position numbers can be used to connect different cylinders. Starting with the largest cylinder, Cylinder 0, the figure clearly shows the position numbers it contains. Afterwards, Cylinder 1, is attached to position number 3 otherwise as the top left of Cylinder 0. This is continued with Cylinder 2 being placed at position 5. However, note that both Cylinders 2 & 3 have different position numbers associated with them.

Position Strings

As now evident, position numbers should be used in a way that connects each cylinder together. However, determining the packing configuration of these cylinders is one that becomes a challenge in combinatorial problems; to avoid a naive approach of checking the suitability of every permutative position, another method must be found. Fortunately, the answer already lies within George et al's literature with 'position strings' (Appendix 8.).

Decoding Process

With the philosophy of genetic algorithms in mind, the simplicity in randomising the contents of a position string in mass, makes this approach a suitable encoding for this problem. However, to prevent a position number from being in an illegal position (Appendix 9.), a decoding procedure must occur. Not only does this decoding process prevent illegal positions, but they also check the feasibility of the position: whether the positioned cylinder fits the container and doesn't intersect with any neighbouring cylinders (Appendix 10.). A similar approach to the one described in Appendix 4. under the genetics header will be incorporated.

Fitness

After decoding a population of these position strings, the best grouping within that generation would be stored alongside the centre points each cylinder contains. In other words the group of cylinders with the largest fitness should be recorded. The fitness function can be derived as the inverse of the objective function, or in more granular terms, the inverse of the distance between the COM of the group of cylinders and the centre of the container. Fig. 7, illustrates the formula that can be used to calculate the COM of a group of objects in one dimension.

$$x_{\mathrm{CM}} = \frac{\sum_i m_i x_i}{\sum_i m_i}$$

Fig. 7 The horizontal COM is calculated based on the sum of each cylinder's central x value multiplied by their mass, in which the sum is subsequently divided by the total mass of each circle. To apply this vertically, use the y coordinate value from a cylinder's centre instead.

Genetic Algorithm

At this point, the preceding headings have laid the blueprints for the genetic algorithm approach this paper will follow. As such that following breaks the steps down more algorithmically:

1. Initialise a population of position strings.
2. Decode the population to ensure legality of position numbers within each position string.
3. Determine whether the largest fitness within the generation is greater than the best recorded fitness across any preceding generation.
    a. If so, then record this generation's position string and cylinder positions.
4. Create a new population.
    a. Use a selection method to obtain parents.
    b. Apply a crossover technique to obtain a position string that contains a mix of position numbers from either parent.
    c. Apply a mutation method.
    d. Repeat until the population matches the previous population size.
5. Repeat steps 2-5 until a maximum number of generations has been met.

Results

Before reviewing the results of the test instances that this paper's implementation has computed, it is paramount to understand the structure of which they are shown and of the primary parameters that have been utilised.

To begin with, the following core features were used during the testing of each test instance:

- **Number of Sides per Cylinder**: *8*
- **Population Size**: *50*
- **Max Generations**: *100*
- **Selection**: *Tournament Style*
- **Crossover**: *Single-Point*
- **Mutation Rate**: *0.1*

Additionally, all the visualisations that will be produced and illustrated in the following sections, have the listed components to them:

- **Figure**: *Acts as the container for the cylinders.*
- **Dotted Container**: *Is the central 60% of the container, this is to clearly show whether the grouping's COM is within it, thus adhering to the weight distribution constraint*.
- **'*X*' Marker**
    - **Orange**: *The centre of the container*.
    - **Red**: *The COM for the grouped cylinders*.

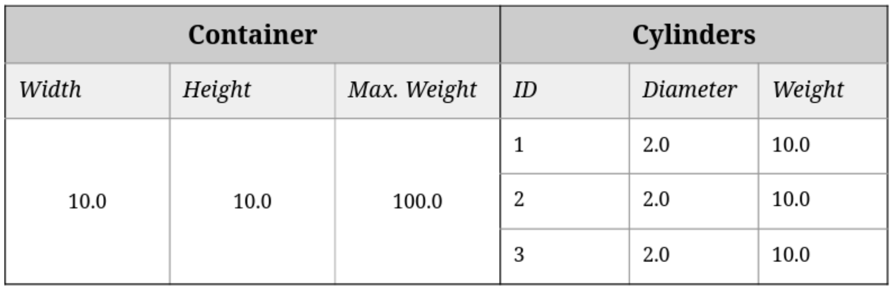
Appendix 11 describes the file structure of the JSON files used to hold more granular information surrounding the results of each test instance.

Basic Test Instances

The first three test instances that this paper is addressing are deemed as basic: cases which don't necessarily stress the limits of the program, but are primarily used as control points that ensure the program behaves in expected ways.

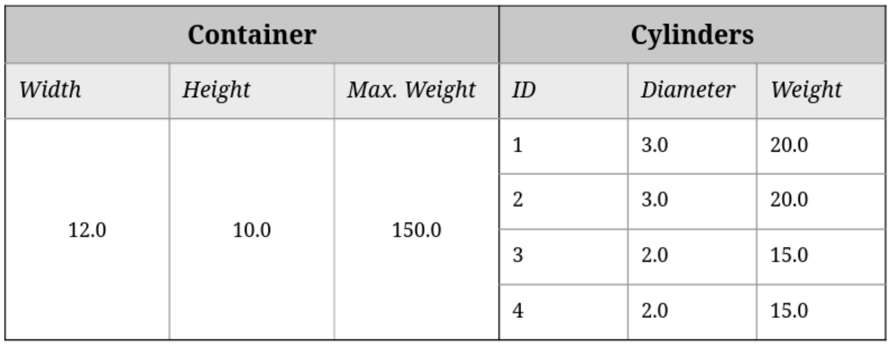
The following tables represent the different test instances within this section:

Table 2
Representing the first test instance: it is a very simple problem that involves three, small, identical cylinders



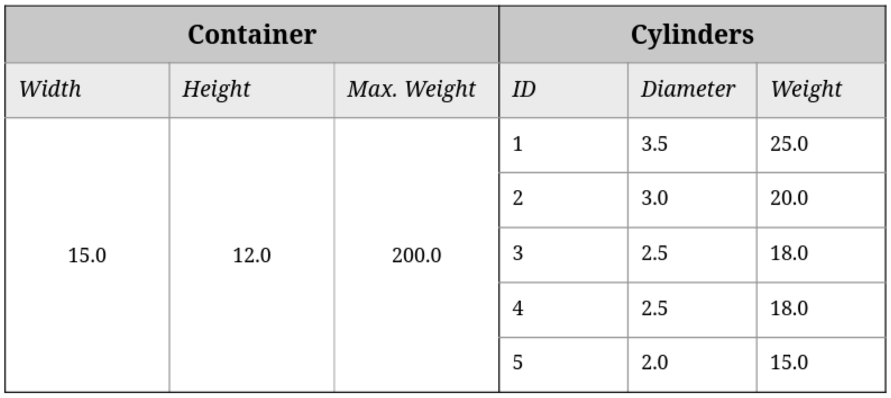
Container			Cylinders		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
10.0	10.0	100.0	1	2.0	10.0
			2	2.0	10.0
			3	2.0	10.0

Table 3
The second test instance increases in complexity with four cylinders of two different sizes.



Container			Cylinders		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
12.0	10.0	150.0	1	3.0	20.0
			2	3.0	20.0
			3	2.0	15.0
			4	2.0	15.0

Table 4
The last of the basic test instances introduces greater variability in cylinder size.



Container			Cylinders		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
15.0	12.0	200.0	1	3.5	25.0
			2	3.0	20.0
			3	2.5	18.0
			4	2.5	18.0
			5	2.0	15.0

Basic Test Results

The following figures represent the results of the genetic approach, with the aforementioned features set. Each figure is in a respective order to their tabulated test case.

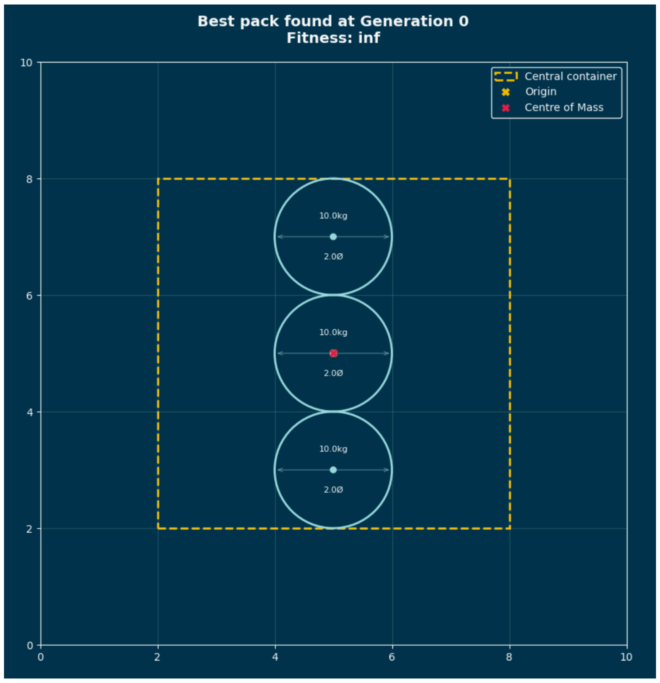


Fig. 8 A visualisation for the result of test instance 1, as described in Table 2. It shows a perfect packing configuration as described by the figure's fitness header as "*inf*".

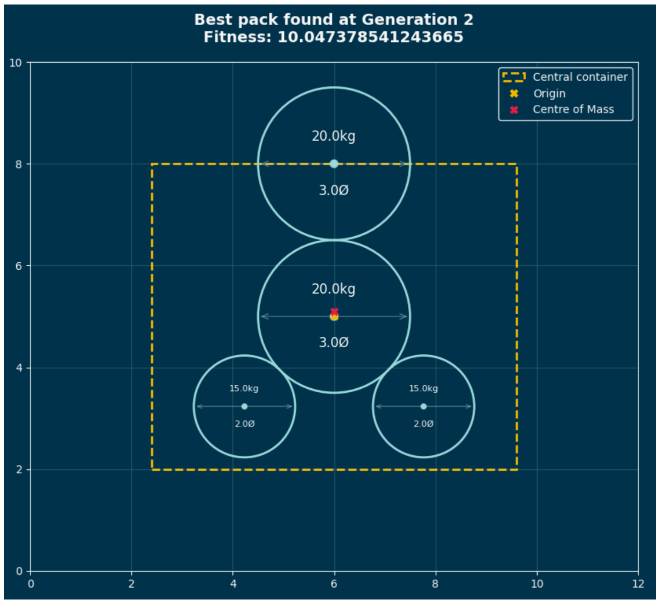


Fig. 9 A visualisation for the result of test instance 2, as described in Table 3. It shows a packing configuration that, albeit is not perfect, has a COM that is well within the central 60% of the container.

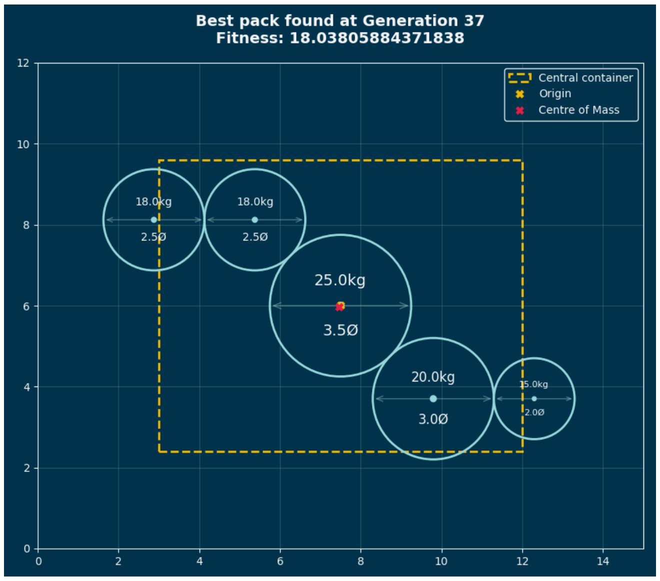


Fig. 10 A visualisation for the result of test instance 3, as described in Table 4.

Advanced Test Instances

The next four tests within this paper apply a greater emphasis on putting an increased strain to the edge-cases of the implementation for this project.

Similarly, to the basic instances, the following tables represent the different test instances within this section.

Table 5
The first of the advanced test instances introduces a problem that requires tight packing, wherein, many increased sized cylinders must fit within a small container.

Container			**Cylinders**		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
15.0	15.0	300.0	1	4.0	35.0
			2	3.5	30.0
			3	3.5	30.0
			4	3.0	25.0
			5	3.0	25.0
			6	2.5	20.0
			7	2.5	20.0
			8	2.0	15.0

Table 6
The next advanced instance tests how well the developed approach does in distributing weights of varying sizes within the container.

Container			**Cylinders**		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
18.0	14.0	400.0	1	3.0	80.0
			2	3.0	80.0
			3	2.5	10.0
			4	2.5	10.0
			5	2.5	10.0
			6	2.5	10.0
			7	3.5	60.0
			8	3.5	60.0

Table 7
The penultimate advanced test instance challenges how well the genetic approach adapts to many small cylinders.

Container			Cylinders		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
20.0	15.0	350.0	1	2.0	15.0
			2	2.0	15.0
			3	2.0	15.0
			4	2.0	15.0
			5	2.0	15.0
			6	2.0	15.0
			7	2.0	15.0
			8	2.0	15.0
			9	2.0	15.0
			10	2.0	15.0
			11	2.0	15.0
			12	2.0	15.0

Table 8
The last advanced test instance contains cylinders of mixed sizes, to put pressure onto the constraints this paper adheres to.

Container			Cylinders		
Width	*Height*	*Max. Weight*	*ID*	*Diameter*	*Weight*
20.0	20.0	500.0	1	5.0	50.0
			2	4.5	45.0
			3	4.0	40.0
			4	3.5	35.0
			5	3.5	35.0
			6	3.0	30.0
			7	3.0	30.0
			8	2.5	25.0
			9	2.5	25.0
			10	2.0	20.0

Advanced Test Results

The following figures show the results of the last four test instances, in their respective order.

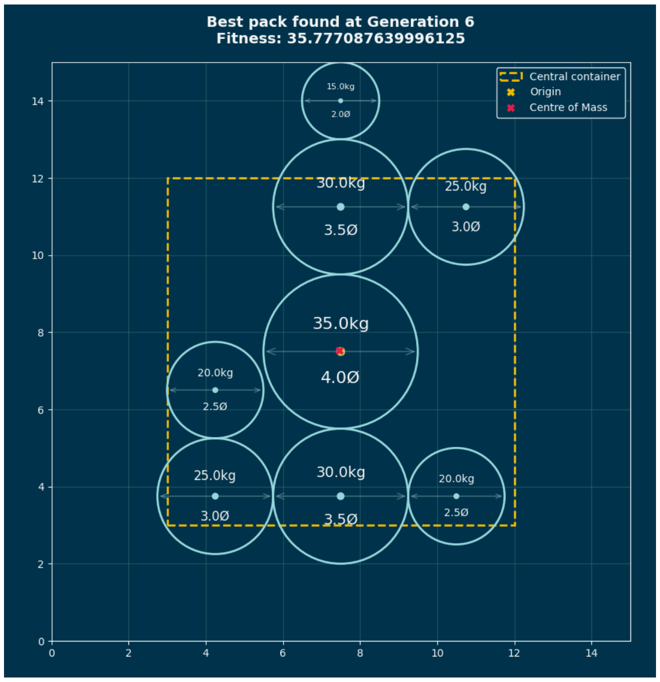


Fig. 11 A visualisation for the result of test instance 4, as described in Table 5.

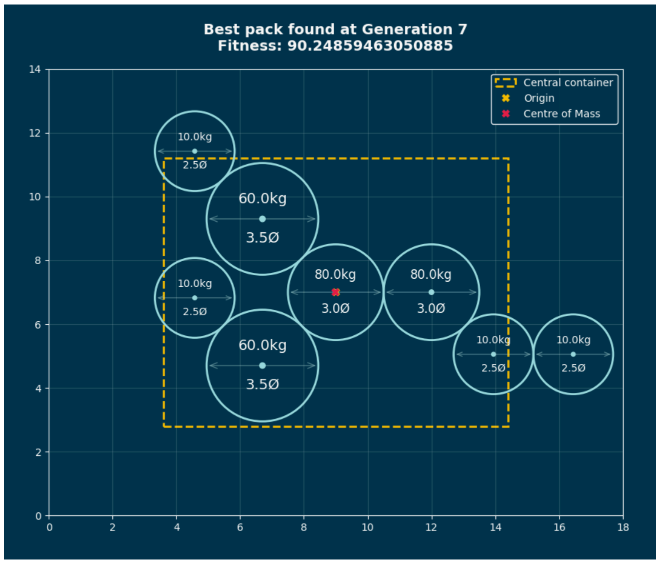


Fig. 12 A visualisation for the result of test instance 5, as described in Table 6.

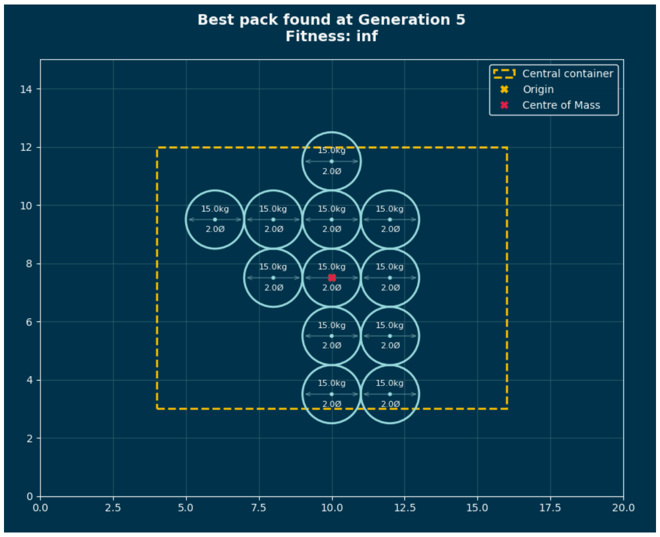


Fig. 13 A visualisation for the result of test instance 6, as described in Table 7.

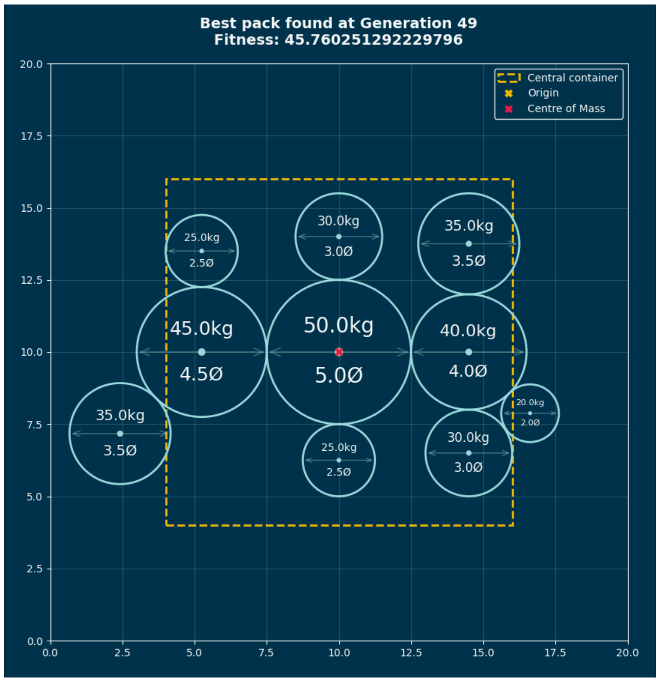


Fig. 14 A visualisation for the result of test instance 7, as described in Table 8.

Reflection

Overall, each of the test instances had passed all of the test instances, which can be rationalised by the following decomposition of each constraint for this problem (Appendix 1.):

- **Geometric Constraint**: Each visualisation for the test instances, figures 8-14, had their cylinders within the boundaries of the figure, which in this implementation acts as the dimensions of the container.

- **Weight Distribution**: Each visualisation of the test instances had their COMs, marked by the red *X*, well within the central 60% of the container, marked by the dotted orange container.

- **Weight Limit**: By analysing the recorded results for each test instance, it can be clearly shown that each of the best cylinder groups have their weights within the maximum weight for that challenge.

- **Loading Order**: The genetic approach doesn't apply any form *spin-out* or *shade-down* rule after the initial algorithm, consequently, each solution naturally follows this constraint by not allowing any cylinders to move after placement.

While each of the produced results succeed in following each constraint whilst creating a strong result and fast execution time, there remains one central area of improvement that can be ascertained. This area of improvement specifically stems from the second basic test instance that has the weakest fitness score throughout all the test instances.

The primary strength of the developed approach revolves around the largest cylinder, with the heaviest mass, being placed at the centre of the container so that the weight distribution constraint would have a higher likelihood of being met. Additional advantages included having several large cylinders of equal mass to eventually converge to a perfect or high-fitness solution, based on the 8-different sides available. However, in the unique case for test instance 2, there were only two large cylinders, which naturally invokes an imbalance within the solution. This imbalance is evident in Fig. 9 where the COM is at a greater lateral position with the algorithm attempting to make up for it by placing the smaller cylinders in the opposing direction. Consequently, a special case should be developed when there are only two large cylinders within the group to produce a solution similar to Fig. 15.

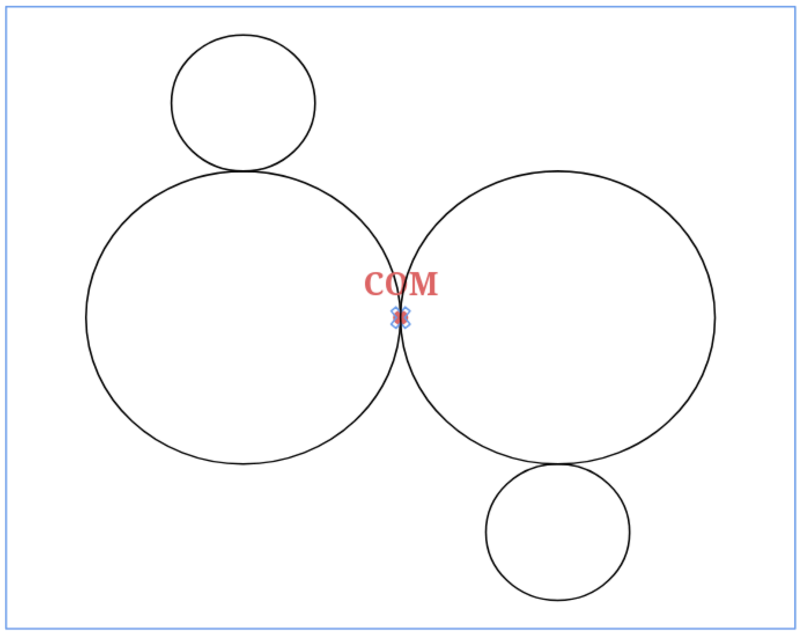


Fig. 15   A special case for two large cylinders within a group (Test Instance 2). Instead of one always at the centre, they instead share the centre, by being a distance equivalent to their radius away from the centre of the container.

Exploration

This section explores the other possible results that can be produced by adapting different values for the core features that were mentioned at the start of the Results section. Additionally, to set as a benchmark, test instance 7 will be utilised across each experiment, whilst the visualisations of each instance can be found under Appendix 12.

To begin with, a series of tests will be conducted singularly to examine the effects of changing only one core feature with another method.

Number of Sides Per Cylinder

Core Features						**Fitness**
Number of Sides Per Cylinder	*Population Size*	*Max. Generations*	*Selection*	*Crossover*	*Mutation Rate*	
60	50	100	Tournament-Style	Single Point	0.1	68.53
70	50	100	Tournament-Style	Single Point	0.1	78.69
80	50	100	Tournament-Style	Single Point	0.1	203.34
90	50	100	Tournament-Style	Single Point	0.1	107.49
100	50	100	Tournament-Style	Single Point	0.1	76.72
120	50	100	Tournament-Style	Single Point	0.1	72.24
240	50	100	Tournament-Style	Single Point	0.1	33.25

Population Size

Core Features						**Fitness**
Number of Sides Per Cylinder	*Population Size*	*Max. Generations*	*Selection*	*Crossover*	*Mutation Rate*	
8	75	100	Tournament-Style	Single Point	0.1	38.92
8	100	100	Tournament-Style	Single Point	0.1	68.80
8	150	100	Tournament-Style	Single Point	0.1	102.81
8	160	100	Tournament-Style	Single Point	0.1	189.27
8	170	100	Tournament-Style	Single Point	0.1	63.78
8	200	100	Tournament-Style	Single Point	0.1	70.53

Max. Generations

No changes in fitness occurred by increasing the maximum number of generations.

Selection Methods

Core Features						**Fitness**
Number of Sides Per Cylinder	*Population Size*	*Max. Generations*	*Selection*	*Crossover*	*Mutation Rate*	
8	50	100	Roulette Wheel	Single Point	0.1	19.55
8	50	100	Rank-based	Single Point	0.1	56.92
8	50	100	Elitist	Single Point	0.1	28.58

Crossover Techniques

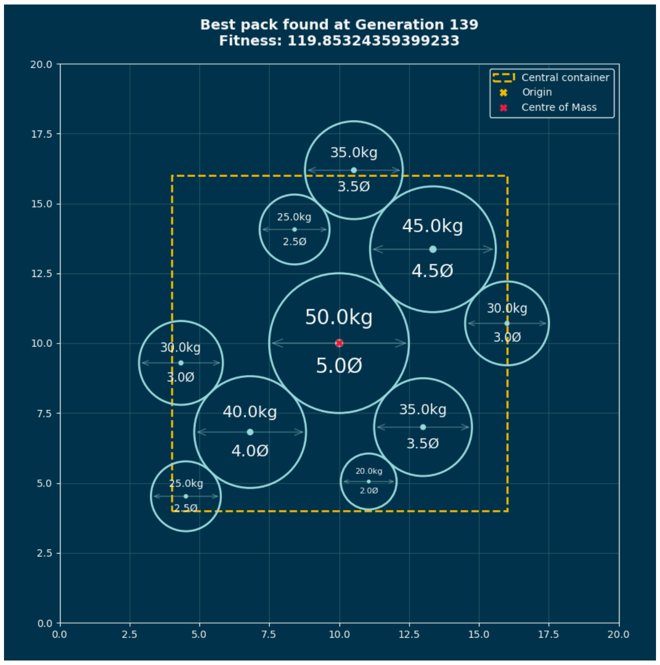
Core Features						**Fitness**
Number of Sides Per Cylinder	*Population Size*	*Max. Generations*	*Selection*	*Crossover*	*Mutation Rate*	
8	50	100	Tournament-Style	Multi Point (2)	0.1	14.17
8	50	100	Tournament-Style	Uniform	0.1	14.08
8	50	100	Tournament-Style	Davis Order	0.1	26.67

Mutation Rate

Core Features						**Fitness**
Number of Sides Per Cylinder	*Population Size*	*Max. Generations*	*Selection*	*Crossover*	*Mutation Rate*	
8	50	100	Tournament-Style	Single Point	0.15	21.88
8	50	100	Tournament-Style	Single Point	0.2	29.07
8	50	100	Tournament-Style	Single Point	0.3	28.34

By combining the best feature of each explored test the following result is produced:

Core Features						**Fitness**
Number of Sides Per Cylinder	*Population Size*	*Max. Generations*	*Selection*	*Crossover*	*Mutation Rate*	
80	160	100	Tournament-Style	Single Point	0.1	119.85



As a result of this, it can be therefore deduced that this problem exhibits strong variable interdependence: improvements in individual variables do not necessarily translate into the best overall result.

References

Journals

E.G. Birgin, J.M. Martinez, D.P. Ronconi. (01 January 2005). Optimizing the packing of cylinders into a rectangular container: A nonlinear approach. *European Journal of Operational Research*. 160(1), pp.19-33. [Online]. Available at: https://doi.org/10.1016/j.ejor.2003.06.018 [Accessed 25 October 2025].

John A. George, Jennifer M. George, Bruce W. Lamar. (03 August 1995). Packing different-sized circles into a rectangular container. *European Journal of Operational Research*. 84(3), pp.693-712. [Online]. Available at: https://doi.org/10.1016/0377-2217(95)00032-L [Accessed 27 October 2025].

Websites

Vidhi Chugh. (2025). *Hessian Matrix: A Guide to Second-Order Derivatives in Optimization and Beyond*. [Online]. datacamp.com. Last Updated: 16 June 2025. Available at: https://www.datacamp.com/tutorial/hessian-matrix [Accessed 05 November 2025].

Esicup. (2018). *History*. [Online]. euro-online.org. Last Updated: October 2018. Available at: https://www.euro-online.org/websites/esicup/history/ [Accessed 15 October 2025].

Isaac Science. (unknown). *Centre of Mass*. [Online]. isaacscience.org. Last Updated: unknown. Available at: https://isaacscience.org/concepts/cp_centre_mass [Accessed 15 November 2025].

Vishwamitra Mishra. (2024). *All about the 0/1 Knapsack Problem Algorithm*. [Online]. medium.com. Last Updated: 20 January 2024. Available at: https://vmlogger.medium.com/all-about-0-1-knapsack-problem-solving-algorithm-96b19be291e6 [Accessed 15 October 2025].

Piber Plastic. (2023). *3 Reasons You Should Use Rectangular Packaging*. [Online]. piberplastics.com.au. Last Updated: 02 January 2023. Available at: https://www.piberplastics.com.au/reasons-to-use-rectangular-packaging/ [Accessed 19 October 2025].

VS&B Containers Group. (2021). *Container Stowage Planning and its Importance*. [Online]. vsnb.com. Last Updated: 15 April 2021. Available at:

https://www.vsnb.com/container-stowage-planning-and-its-importance [Accessed 05 January 2026].

Denis Weaire. (1999). *A Short History of Packing Problems*. [Online]. academia.edu. Last Updated: 15 December 1999. Available at: https://www.academia.edu/65464018/A_Short_History_of_Packing_Problems [Accessed 15 October 2025].

Appendix

1. Problem Constraints

The problem constraints this paper focuses on is described as follows:

- **Geometric constraint**: *All cylinders must fit within the rectangular container boundaries.*

- **Weight distribution**: *The centre of mass of all loaded items must fall within the central 60% of the container (to prevent tipping during transport).*

- **Weight limit**: *Total weight cannot exceed the container's maximum capacity.*

- **Loading order**: *Cylinders are loaded from the rear and cannot be moved once placed.*

2. Similarity to Kepler's Conjecture

It is interesting to mention that when viewing this project's problem in the second dimension, where cylinders are circles and containers being rectangles, it begins to resemble Kepler's Conjecture, as mentioned in the Introduction.

3. Stabilised Packing

George et al considers *stability* as the pursuit of creating a solution that is more realistic and consequently more practical. The three rules of stability is defined as a packing configuration where each circle is either:

(i) touching the bottom of the rectangle; or
(ii) touching the left or right sides and resting on top of a circle at least as big as itself; or
(iii) resting on top of two supporting circles

If one or more of the above points are not met, the packed solution is deemed unstable.

4. Packing Methods

Please note that all the methods, excluding the genetic approach, requires the circles to be sorted in descending order first.

Samepack

The Samepack method focuses on placing circles of similar sizes in close proximity with one another. This is exploited into two phases. The first phase begins by grouping same-sized circles together as well as placing the largest-sized group of circles to the bottom of the container, denoted as side two in the literature. After this the next group which is now incrementally getting smaller, along with another side of the container, is selected. This phase repeatedly places circles of a given group as close as possible to a given side until no further circles can be placed in the rectangle. Once we have reached the terminating condition, the second phase starts. This phase simply applies two rules respectively: a “spin-out” rule that shifts circles closer to their respective edge of the cylinder, followed by a “shade-down” rule that attempts to shift circles down to the bottom of the rectangle without any intersections.

Wallpack

The Wallpack heuristic does a similar two phase approach. The first, places the four largest circles in each corner and then iterates through all the available circles to determine the best set of configurations for each side of the rectangle. Once completed, the next largest circle is chosen from those that remain, and is temporarily placed to the centre of the rectangle before being shifted as far as possible to one side of the rectangle. This process continues until there are no more circles left to pack or till no more circles can be placed on any side of the rectangle. When this end condition is met, it enters the second phase which is identical to Samepack; spin-out and then shade-down to improve the packing configuration.

Stable1

Utilising both side and position numbers, it determines where the next circle will be placed. The order of configuration is as follows: bottom (positions 2, 4, 7, 11…), left (positions 3, 6, 10…), and right (positions 5, 8, 12…). These position values are calculated from the formula below.

$$p = \tfrac{1}{2}(i+1)(i+2) + s - 1.$$

The algorithm starts at the bottom-left corner of the rectangle ($p_k = 1$). Circles are then taken in the order of configurations as described above; successively adding a circle in feasible locations along the bottom, then the left side, and finally the right side. For example, the algorithm checks if the next circle can fit along the bottom; if not, it checks the left side, and if it can’t fit the left side either then try the right side of the rectangle. When no more circles can be placed on any side of the rectangle, the algorithm resorts to the random heuristic approach to attempt to pack the remaining circles.

Stable2

This method starts similarly to Stable1 where the largest circle will be placed to the bottom left of the rectangle, and if possible place the second to largest circle to the bottom right corner. After this initialisation, a similar process to wallpack begins by determining the best combination of circles which best fit the remaining spaces along the bottom, then the left, and then the right sides of the rectangle, regardless of circle size. Once this has been completed, the algorithm applies the random heuristic approach to try to pack the remainder of the circles.

Random

Rather than randomising the placement of circles within the rectangle, a series of sequences of randomly chosen position numbers are generated. Once each of which is “decoded”, the best determined configuration is selected. This procedure also provides a mechanism for generating the initial population of strings used in the genetic algorithm.

Genetic

The idea behind this genetic algorithm approach is to combine position strings that are not normally found together. This does mean that the population would consist of a predetermined position order for each circle, some of which may yet be defined or are in feasible locations; but provides new packing configurations that could result in a better solution found in comparison to the previous methods.

To begin with, a population of random strings that utilise the random heuristic is made. Once these strings are generated, a “decoding” procedure is used to determine the location in which a circle should be placed. The following demonstrates an example of the decoding process.

Consider the string:

1, 2, 14, 8, 3, … 25 , where k = 4, $p_k = 8$, and $f_k = 14$

Or in a less mathematical sense, where we are looking at circle 4 (k = 4), which starts at position number 8 ($p_k = 8$), in which we have a total of 14 different positions to select from ($f_k = 14$). Note that this total is determined by the following function:

$$f_k = \begin{cases} k & \text{if } k = 1, \\ \frac{1}{2}(k^2 + 3k) & \text{if } k \geq 2. \end{cases}$$

With this in mind, we then check if the current position number is a feasible location; based on constraints such as whether it does not intersect with another circle or is within the container's bounds. If the position is feasible, then move onto the next circle, alongside its new position number and total. However, if not, then we check for position 9 (p_k+1), then 10 (p_k+2) if the previous position was infeasible, and this incrementation continues to f_k before wrapping to position 1 and then to p_k-1 (7). Therefore, the procedure checks for a feasible location using the following sequence of position numbers: 8, 9, 10, …, 14, 1, …, 7. If none of which are feasible, the algorithm discards that circle. Note that for k=3, as its position (14) is out of the range of total possible positions (f_k=9), we iterate from position 1, and end at f_k.

Subsequently strings are then selected based on their fitness in proportion to the sum of all fitness values. Afterwards, a single-point crossover is applied between the chosen strings to breed new packing configurations, with the addition of a mutation that is applied to a random cell in the new string to ensure variance. This process is repeated until all the strings have the same fitness value or if a maximum number of generations has been reached.

5. Filtering Design Approaches

From the above literature alongside this paper's objective to use a genetic algorithm to solve this cargo loading problem which includes several constraints, a few concepts of the literature can be made redundant. To begin with, Birgin et al's mathematical approach unfortunately falters at the constraint surrounding loading order. This constraint prohibits any cylinders to be moved once placed, which directly opposes the Hessian approach which dynamically shifts cylinders around to optimise space availability. This constraint not only makes the Hessian matrix unusable for this solution, but also removes any application of rules such as spin-out and shade-down.

6. New Utilisation of Position & Side Numbers

The application of these terms will serve different purposes to those from the original literature. Normally, side numbers from George et al would denote the edges of the container, but due to the weight distribution constraint, which requires the centre of mass of all loaded items to be within the central 60% of the container, giving the opportunity for a circle to be placed near an edge would be disadvantageous. This is similar to the starting position. Instead of initially placing the first cylinder on the bottom left corner of the container, a cylinder should be placed in the centre of the container. By applying a descending order sort, it means that the largest cylinder will always start at the centre, which in average cases would put a large portion of the mass at the centre of the container. Consequently, it will begin to produce a solution that will have a strong chance of passing the weight distribution constraint.

7. Varying Side Positions

If the quantity of side positions are too few, the algorithm may not perform as well, but may carry a faster execution speed. Consider having four sides allocated to each cylinder, and there are four cylinders of equal weight to pack. As one cylinder is always at the centre, three cylinders remain to be placed around it, which will always produce an imperfect result. On the other hand, if the quantity of side positions per cylinder are too large, the greater the chance for producing a denser and better packing configuration, but at the consequence of an increased computing requirement and execution time.

8. Position Strings

Position strings are a list of position numbers that are in a range from 0 to a value equal to (n * s) - 1, where:

- *n* is the total number of cylinders.
- s is the total number of sides.

Consider four cylinders with a position string: [1, 2, 9]. Note that the largest cylinder will always be at the centre of the container, hence it does not have its own position number.

Within the position string, each position value describes where other cylinders should be placed. The first position number within the given string (1), signifies that the second largest cylinder will be positioned to Cylinder 0, the largest cylinder. In general, determining the cylinder another cylinder can attach to can be shown within Table 1, or can be mathematically computed by doing something similar to: *t // p*, where:

- *p* is the position number being investigated.
- *t* being the total number of position number; equivalent to *n * s*.

Once the cylinder to attach onto has been determined, another calculation must be performed to actively place the circle at its rightful position. To begin with, the cylinder will be positioned to the right, or the equivalent of side 0, to the cylinder it is being attached to. Subsequently, the centre of the cylinder will then be rotated relatively to the centre of the cylinder to attach onto. So, for position number 1, Cylinder 1, will be rotated to the top-right position of Cylinder 0.

This process repeats for each other position number within the position string. With a position number of 2, the third cylinder (Cylinder 2), will be attached onto Cylinder 0, at the top point. On the other hand, position number 9, will instead attach the last Cylinder (Cylinder 3) onto Cylinder 1. To determine the side this position number will be rotated to, the formula: *p % s* will be applied.

9. Illegal Position Numbers

Illegal position numbers are those position numbers within a position string that exceed the feasible range of numbers possible for its current index position.

Consider the illegal position string: [12, 1, 2].
The first position number is 12, using Appendix 8's *t // p* formula to determine the cylinder the second largest cylinder (Cylinder 1) should be placed onto. However, as the result of that equation would be equal to 1, it is essentially saying to place Cylinder 1 onto Cylinder 1, which is simply not possible, hence the position number 12 being an illegal value for its position.

To address this, a conditional such as *p >* (*i + 1) * s* can be applied to prevent a position number existing at an infeasible location. The components of the formula are as follows:

- *i* is the current index in the position string. In this case, indices start at 0.
- *s* is the total number of sides.

If an illegal position exists, a similar approach to Appendix 4's Genetic position string problem, can be utilised. This is such that, the position number for that index, would start from the first position number in the possible range (0), and then perform feasibility checks to determine whether the new position is viable for the current cylinder. If it is, then update the position number for that index, and place the cylinder accordingly before moving to the next position number. However, if the new position is not feasible, then continue incrementing till the *(n * s) - 1* position number. If there are no positions that work for that particular cylinder, it will be discarded.

10. Determining Intersections

One can determine the intersection between two circles, aliased as cylinders for this project, by calculating the Euclidean distance between the centre of any two circles and checking whether this distance is greater than the sum of the radii of the two circles.

11. Result File Structure

Each compute of a test instance has been recorded within a JSON file of the following name structure: *TEST_Instance[{EXECUTE_TEST_CASE}]-SMU[{smu}]-CTU[{ctu}]-MR[{mut_rate}].json*, where:

- **EXECUTE_TEST_CASE**, *is the test instance being investigated.*
- **smu**: The *Selection Method Used*, i.e. tournament selection.
- **ctu**: The *Crossover Technique Used*, i.e. single-point crossover.
- **mut_rate**: The *Mutation Rate* that had been applied in the process.

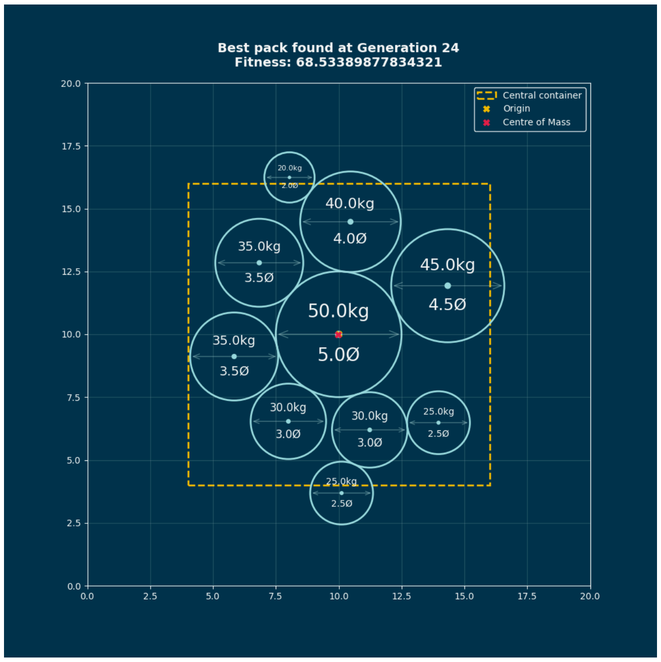
The contents of each file has a similar layout to the following bullet pointed structure:

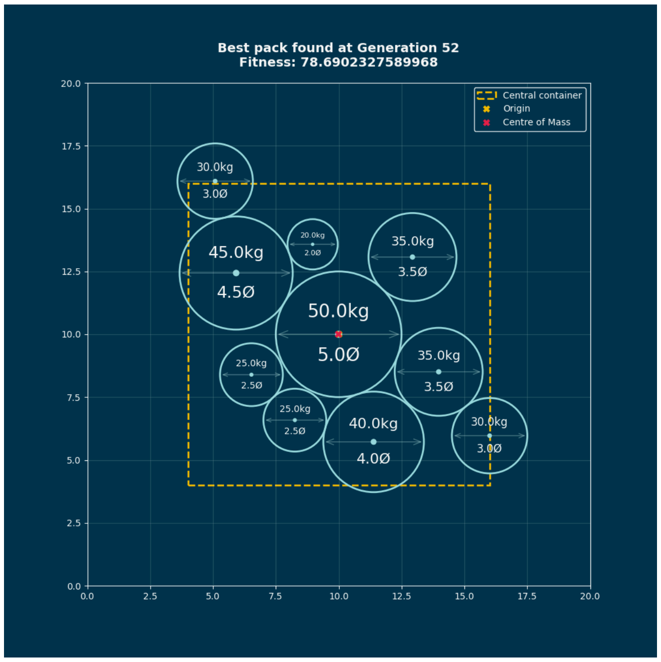
- **Bin X**: *The "bin" of cylinders that homes the following information.*
    - **Compute Time**: *The time taken for the genetic algorithm to complete*
    - **Population Size**: *The size of the population used during the evolutionary process*.
    - **Binned Cylinders**: *A list of all cylinders associated with this bin*.
    - **Max Weight**: *The maximum weight of this bin/container*.
    - **Best Cylinder Group**: *Holds information regarding the best overall packing configuration*.
        - **Weight**: *The weight of the produced group*.
        - **Fitness**: *The best fitness overall*.
        - **Cylinder Positions**: *The positions of each cylinder's centre point*.
    - **Key Generations**: *A list of the generations which left an impressionable fitness compared to each preceding generation.*
    - **Fitness History**: *A list of the fitnesses respective to each key generation.*
    - **Selection Method Used**
    - **Crossover Technique Used**
    - **Mutation Rate**

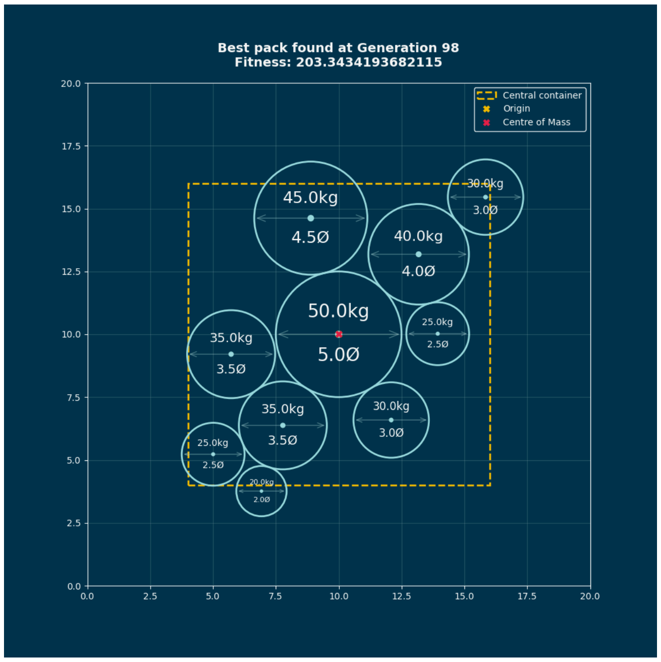
12. Explored Visualisations

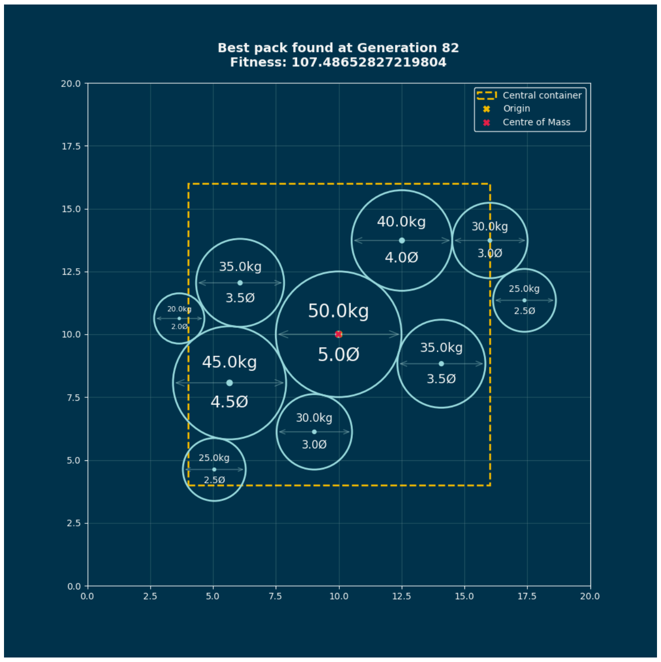
Each visualisation under each header is in the respective order of the rows of each corresponding table within the Exploration header.

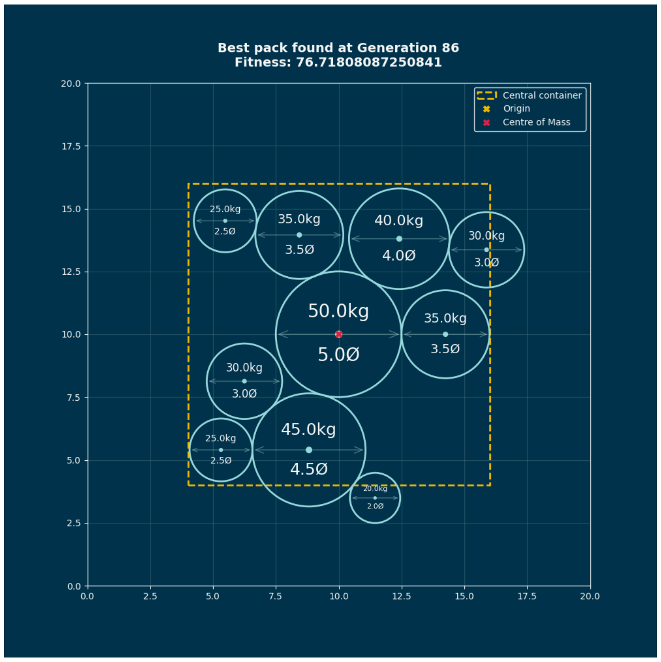
Number of Sides Per Cylinder

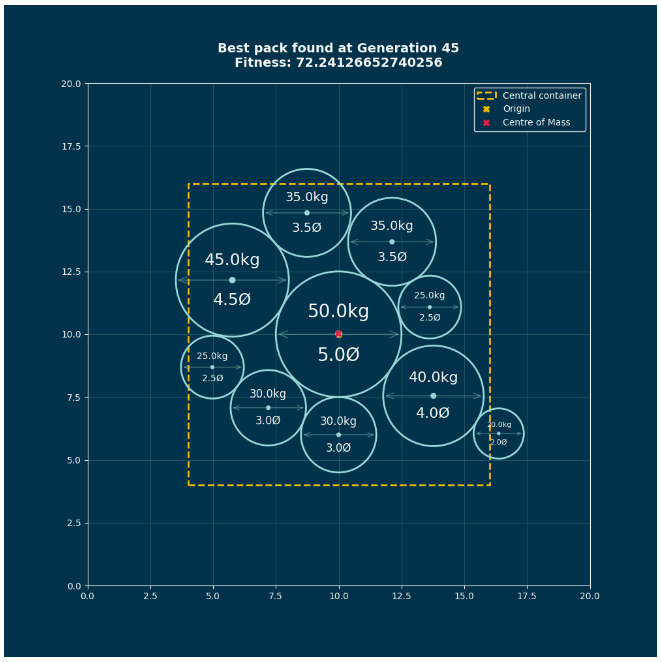


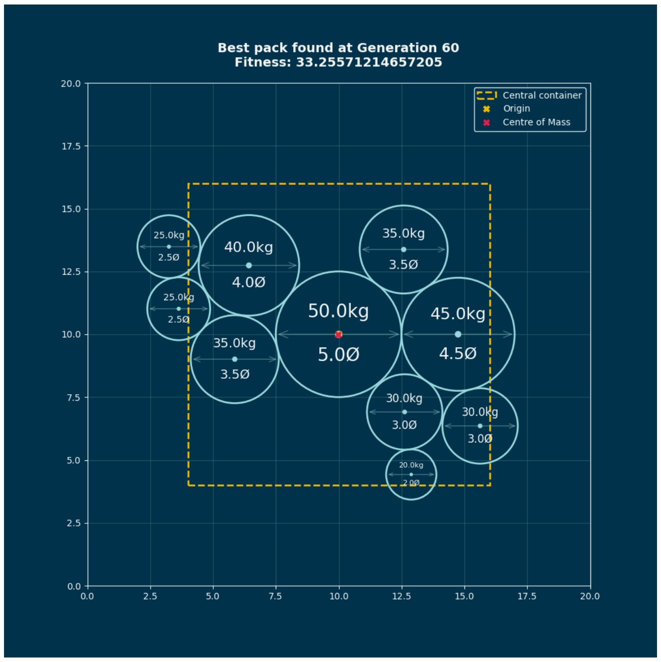



Best pack found at Generation 98
Fitness: 203.3434193682115
Central container
Origin
Centre of Mass
45.0kg
4.5Ø
30.0kg
3.0Ø
40.0kg
4.0Ø
50.0kg
5.0Ø
25.0kg
2.5Ø
35.0kg
3.5Ø
35.0kg
3.5Ø
30.0kg
3.0Ø
25.0kg
2.5Ø
20.0kg
2.0Ø
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

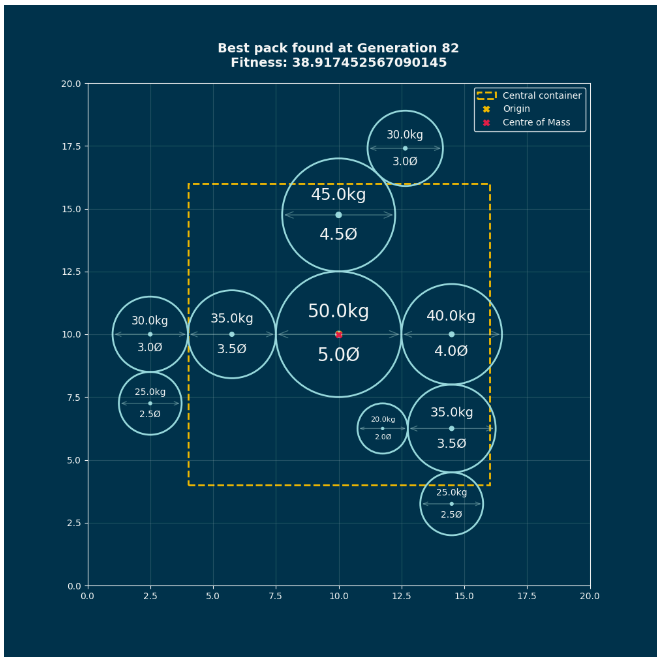

Best pack found at Generation 82
Fitness: 107.48652827219804
Central container
Origin
Centre of Mass
40.0kg
4.0Ø
30.0kg
3.0Ø
35.0kg
3.5Ø
25.0kg
2.5Ø
20.0kg
2.0Ø
50.0kg
5.0Ø
45.0kg
4.5Ø
35.0kg
3.5Ø
30.0kg
3.0Ø
25.0kg
2.5Ø
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

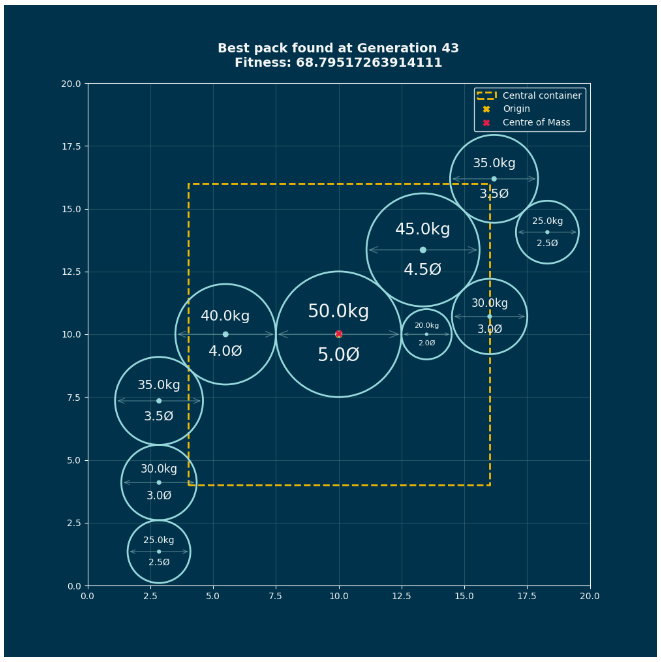

Best pack found at Generation 86
Fitness: 76.71808087250841
Central container
Origin
Centre of Mass
25.0kg
2.5Ø
35.0kg
3.5Ø
40.0kg
4.0Ø
30.0kg
3.0Ø
50.0kg
5.0Ø
35.0kg
3.5Ø
30.0kg
3.0Ø
25.0kg
2.5Ø
45.0kg
4.5Ø
20.0kg
2.0Ø

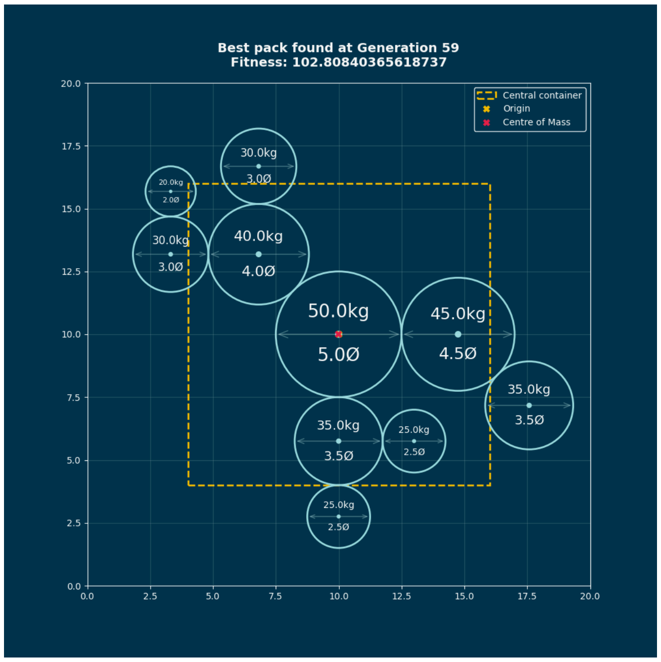

Best pack found at Generation 45
Fitness: 72.24126652740256
Central container
Origin
Centre of Mass
35.0kg 3.5Ø
35.0kg 3.5Ø
45.0kg 4.5Ø
50.0kg 5.0Ø
25.0kg 2.5Ø
25.0kg 2.5Ø
40.0kg 4.0Ø
30.0kg 3.0Ø
30.0kg 3.0Ø
20.0kg 2.0Ø

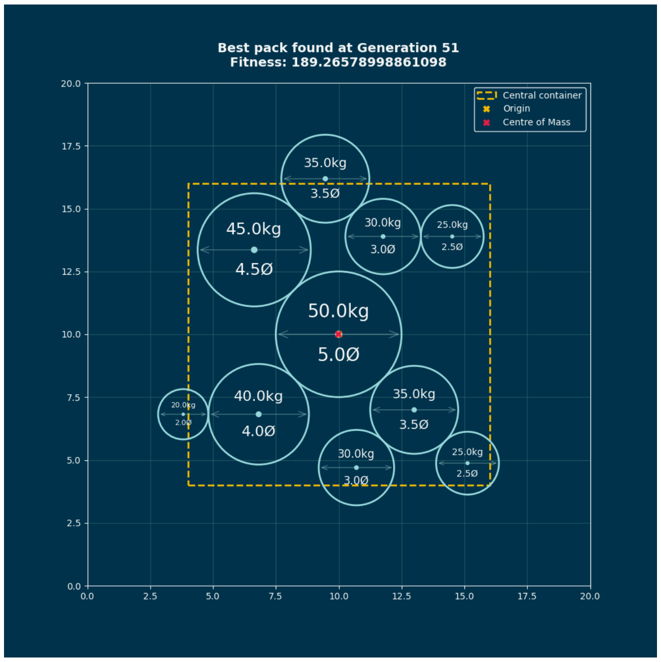

Best pack found at Generation 60
Fitness: 33.25571214657205
Central container
Origin
Centre of Mass
25.0kg 2.5Ø
40.0kg 4.0Ø
35.0kg 3.5Ø
25.0kg 2.5Ø
50.0kg 5.0Ø
45.0kg 4.5Ø
35.0kg 3.5Ø
30.0kg 3.0Ø
30.0kg 3.0Ø
20.0kg 2.0Ø

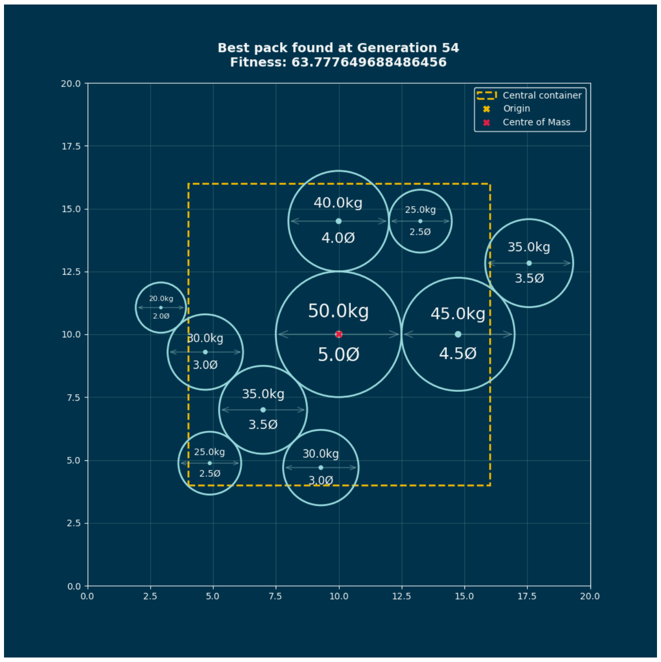
Population Size

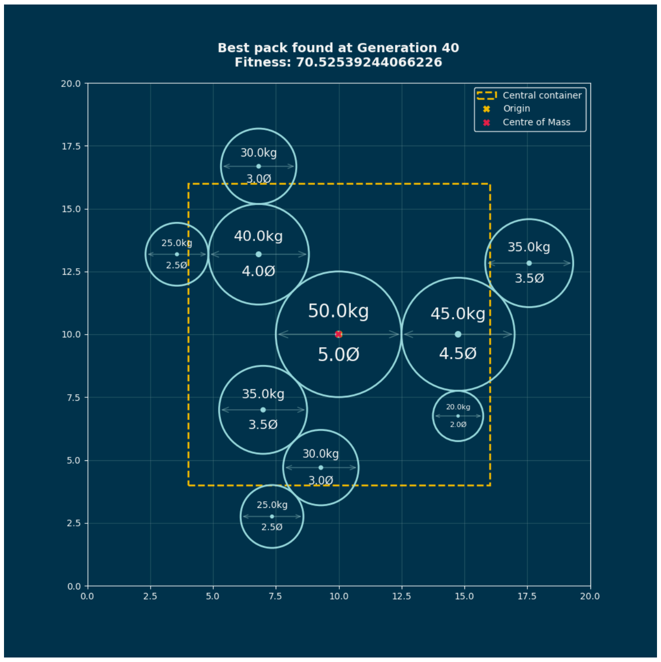




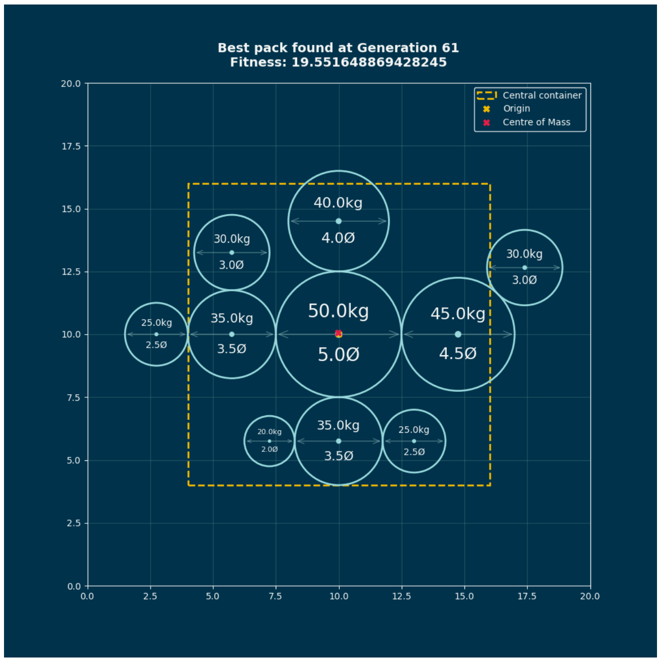

Best pack found at Generation 59
Fitness: 102.80840365618737
Central container
Origin
Centre of Mass
50.0kg 5.0Ø
45.0kg 4.5Ø
40.0kg 4.0Ø
35.0kg 3.5Ø
35.0kg 3.5Ø
30.0kg 3.0Ø
30.0kg 3.0Ø
25.0kg 2.5Ø
25.0kg 2.5Ø
20.0kg 2.0Ø

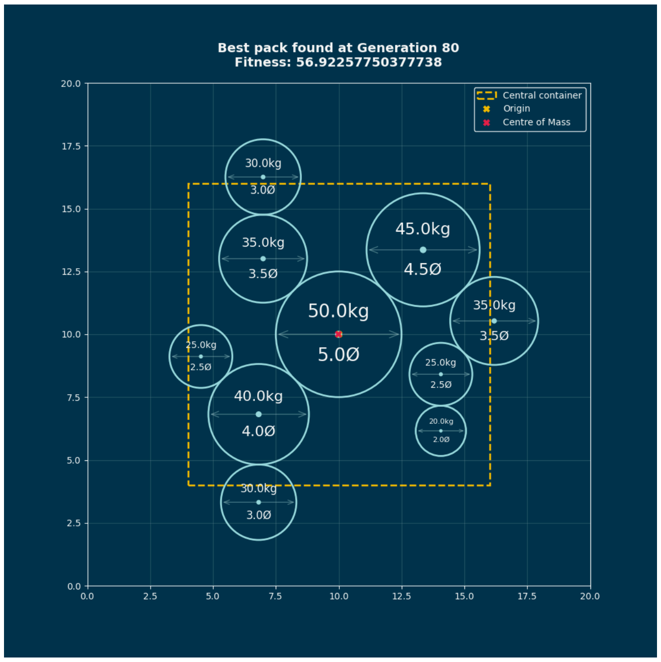

Best pack found at Generation 51
Fitness: 189.26578998861098
Central container
Origin
Centre of Mass
50.0kg 5.0Ø
45.0kg 4.5Ø
40.0kg 4.0Ø
35.0kg 3.5Ø
35.0kg 3.5Ø
30.0kg 3.0Ø
30.0kg 3.0Ø
25.0kg 2.5Ø
25.0kg 2.5Ø
20.0kg 2.0Ø

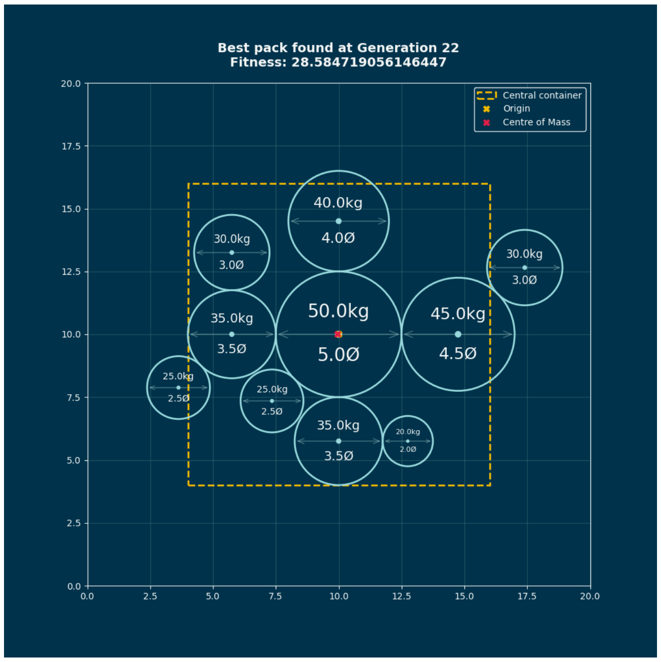

Best pack found at Generation 54
Fitness: 63.777649688486456
Central container
Origin
Centre of Mass
40.0kg 4.0Ø
25.0kg 2.5Ø
35.0kg 3.5Ø
20.0kg 2.0Ø
50.0kg 5.0Ø
45.0kg 4.5Ø
30.0kg 3.0Ø
35.0kg 3.5Ø
25.0kg 2.5Ø
30.0kg 3.0Ø


Best pack found at Generation 40
Fitness: 70.52539244066226
Central container
Origin
Centre of Mass
30.0kg 3.0Ø
25.0kg 2.5Ø
40.0kg 4.0Ø
35.0kg 3.5Ø
50.0kg 5.0Ø
45.0kg 4.5Ø
35.0kg 3.5Ø
20.0kg 2.0Ø
30.0kg 3.0Ø
25.0kg 2.5Ø

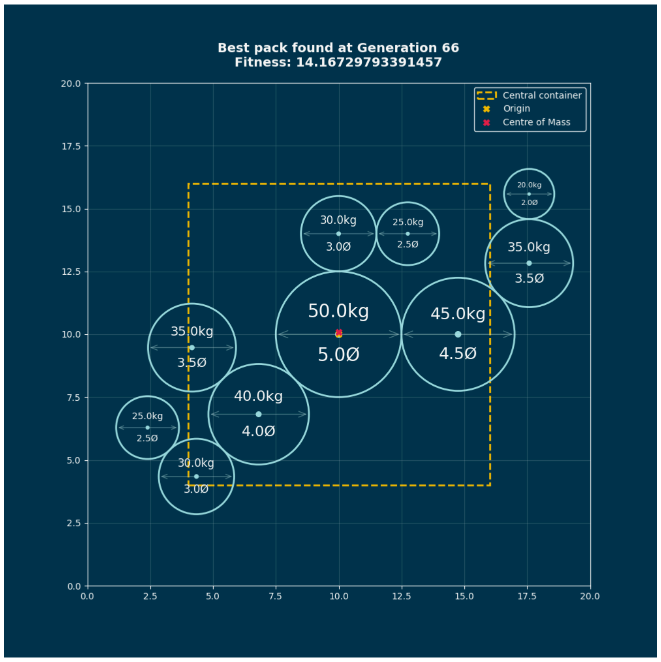
Selection Methods


Best pack found at Generation 61
Fitness: 19.551648869428245
Central container
Origin
Centre of Mass
40.0kg
4.0Ø
30.0kg
3.0Ø
30.0kg
3.0Ø
25.0kg
2.5Ø
35.0kg
3.5Ø
50.0kg
5.0Ø
45.0kg
4.5Ø
20.0kg
2.0Ø
35.0kg
3.5Ø
25.0kg
2.5Ø

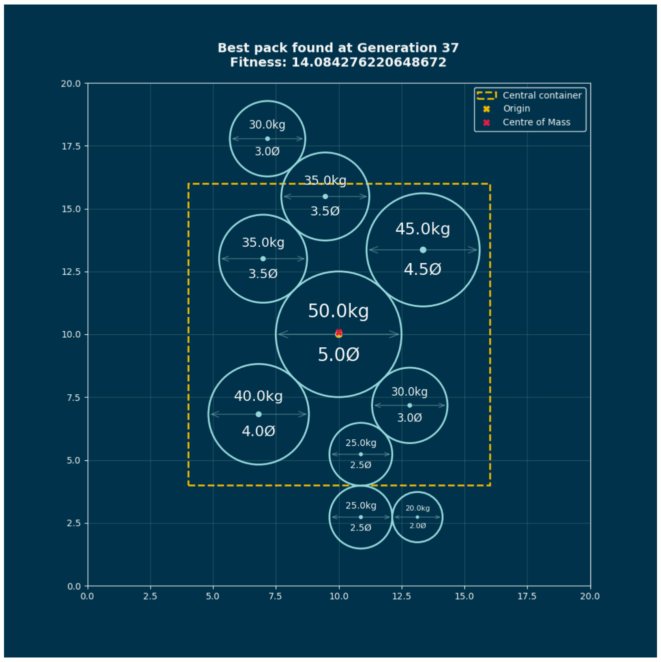

Best pack found at Generation 80
Fitness: 56.92257750377738
Central container
Origin
Centre of Mass
30.0kg
3.0Ø
35.0kg
3.5Ø
45.0kg
4.5Ø
50.0kg
5.0Ø
35.0kg
3.5Ø
25.0kg
2.5Ø
25.0kg
2.5Ø
40.0kg
4.0Ø
20.0kg
2.0Ø
30.0kg
3.0Ø



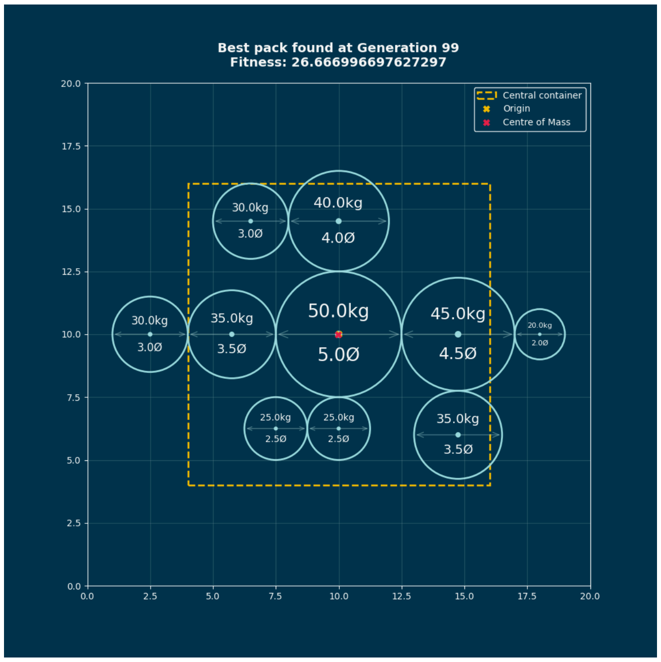
Crossover Techniques



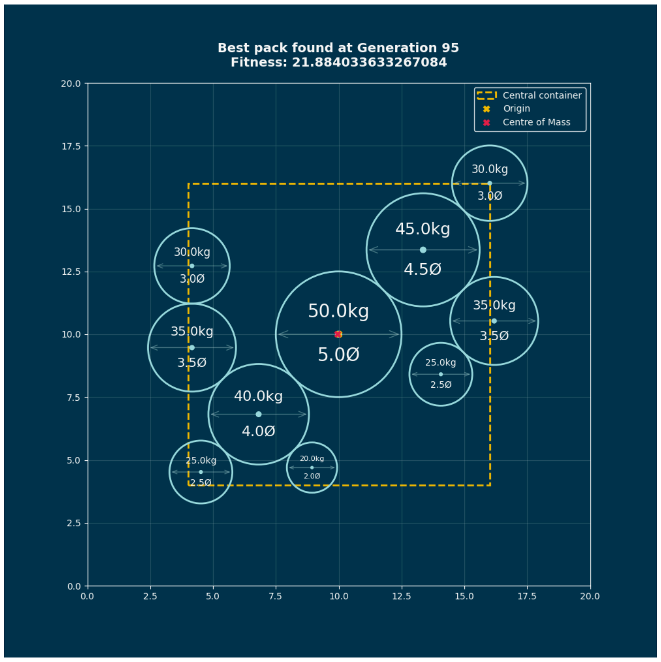
Best pack found at Generation 66
Fitness: 14.16729793391457
Central container
Origin
Centre of Mass
20.0kg 2.0Ø
30.0kg 3.0Ø
25.0kg 2.5Ø
35.0kg 3.5Ø
50.0kg 5.0Ø
45.0kg 4.5Ø
35.0kg 3.5Ø
40.0kg 4.0Ø
25.0kg 2.5Ø
30.0kg 3.0Ø

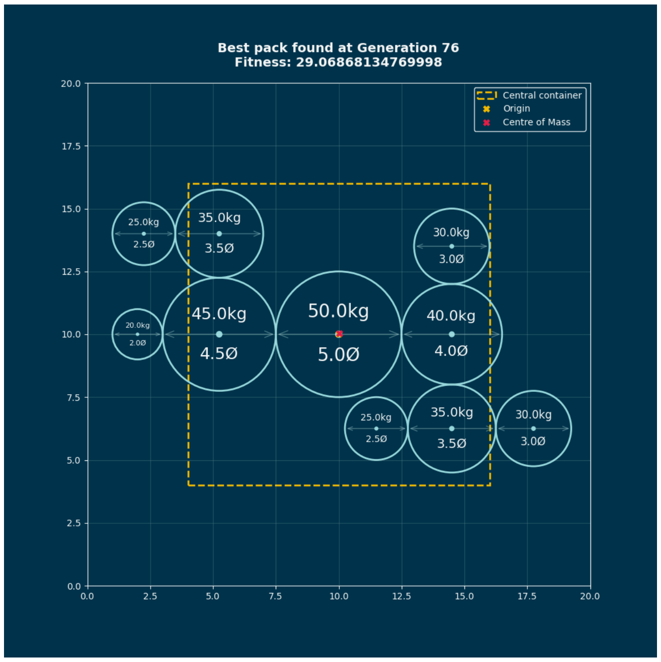


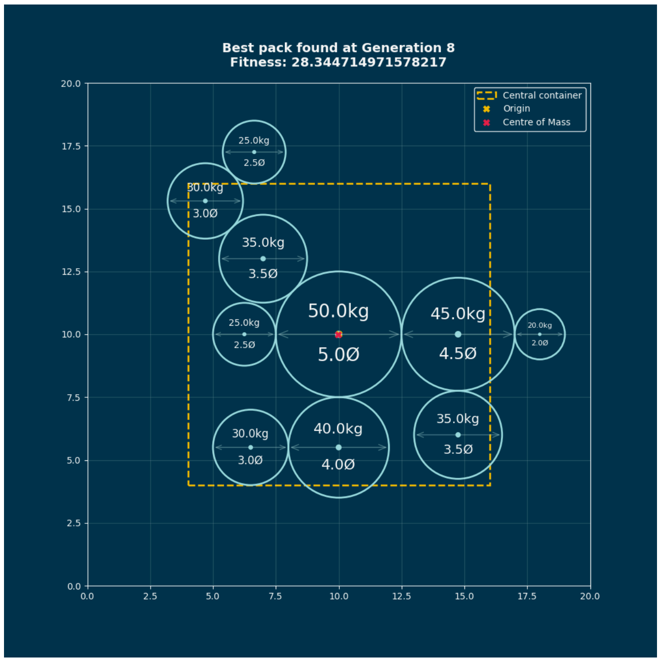
Best pack found at Generation 37
Fitness: 14.084276220648672
Central container
Origin
Centre of Mass
30.0kg 3.0Ø
35.0kg 3.5Ø
35.0kg 3.5Ø
45.0kg 4.5Ø
50.0kg 5.0Ø
40.0kg 4.0Ø
30.0kg 3.0Ø
25.0kg 2.5Ø
25.0kg 2.5Ø
20.0kg 2.0Ø



Mutation Rate




Best pack found at Generation 76
Fitness: 29.06868134769998
Central container
Origin
Centre of Mass
25.0kg 2.5Ø
35.0kg 3.5Ø
30.0kg 3.0Ø
20.0kg 2.0Ø
45.0kg 4.5Ø
50.0kg 5.0Ø
40.0kg 4.0Ø
25.0kg 2.5Ø
35.0kg 3.5Ø
30.0kg 3.0Ø


Best pack found at Generation 8
Fitness: 28.344714971578217
Central container
Origin
Centre of Mass
25.0kg 2.5Ø
30.0kg 3.0Ø
35.0kg 3.5Ø
25.0kg 2.5Ø
50.0kg 5.0Ø
45.0kg 4.5Ø
20.0kg 2.0Ø
30.0kg 3.0Ø
40.0kg 4.0Ø
35.0kg 3.5Ø