Northumbria University
NEWCASTLE

School of Computer Science

ASSESSMENT BRIEF	
Module Title:	Evolutionary Computation
Module Code:	KV6018
Academic Year / Semester:	2025-26 / Semester 1
Module Tutor / Email (all queries):	Lucas França/lucas.franca@northumbria.ac.uk
% Weighting (to overall module):	100%
Assessment Title:	Cargo Container Loading
Date of Handout to Students:	29/09/25
Mechanism for Handout:	Module Blackboard Site & Seminar in Week 3
Deadline for Attempt Submission by Students:	15/01/26 3:59pm GMT
Mechanism for Submission:	Document upload to Module Blackboard Site
Submission Format / Word Count	**Part 1**: Upload your written report as a single PDF to the TurnItIn portal on Blackboard. **Part 2**: Please upload your written report as a single PDF document containing the link to your code Github repository.
Date by which Work, Feedback and Marks will be returned:	12/02/26
Mechanism for return of Feedback and Marks:	Mark and individual written feedback sheet will be uploaded to the Module Site on Blackboard.

LEARNING OUTCOMES

The learning outcomes (LOs) for this module are:-

LO1: Describe, explain, compare and contrast the features of different Evolutionary Algorithms.
LO2: Design, implement and evaluate an Evolutionary Algorithm for a specific problem.

LO3: Clearly document the process of developing and testing an Evolutionary Algorithm.
LO4: Demonstrate an awareness of the literature on Evolutionary Algorithms, and use this to evaluate different approaches.

This assessment addresses learning outcomes LO1, LO4 (Part 1) and LO2 and LO3 (Part 2).

This assessment concerns a cargo container loading problem: We are given a rectangular cargo container and a set of cylindrical containers (barrels, drums, tanks) of different sizes and weights. Each cylindrical container has a diameter and weight. The problem is to place the cylinders in the container to minimize wasted space while ensuring weight distribution constraints are satisfied.

The container is treated as a two-dimensional rectangular footprint (width × depth) with a limit weight w. Each cylindrical container is modelled as a circle of given diameter placed on the container floor at a specific (x,y) coordinate. The goal is to place all circles subject to constraints.

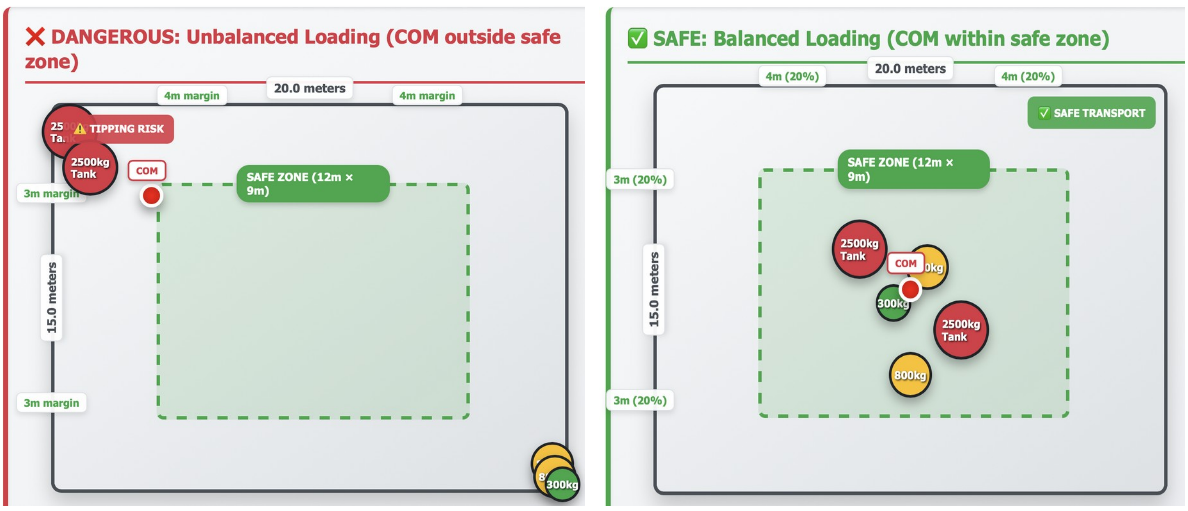
Key Constraints:

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.

Example Scenarios:

Consider loading cylindrical containers of varying diameters and weights into a 20×15 m unit container.

Scenario A: Poor arrangement leads to wasted space and unbalanced weight distribution
Scenario B: Optimised arrangement achieves better space utilization and balanced loading



Heavy Tank (2500kg, Ø2.0m) Medium Drum (800kg, Ø1.5m)

Light Barrel (300kg, Ø1.2m) Center of Mass

This assessment has two parts; an analysis/design section, and an implementation and exploration section. Please note that you will be able to begin work on Part 1 before we have covered all of the material required for Part 2, and we *strongly urge* you to do so.

Part 1 (30%)

Write a short report (no more than 1500 words), made up of the following:

1. *Literature review*: (1) A short history of the rectangle packing problem and related variants, its complexity, and its applications, and (2) a description of at least *three* algorithms for its solution, all with proper citations to academic literature. One of the methods discussed must be an evolutionary algorithm. (750 words, approx.) This section accounts for 10% of the marks available.

2. *Analysis and design*: A description of your own proposed *evolutionary* approach to solving the problem, giving a description of the three main components of your solution: (1) problem encoding, (2) fitness function, and (3) local search. (750 words, approx.) This section accounts for 20% of the marks available.

For both sections, you will be marked on overall presentation, quality of writing and referencing, etc.

Part 2 (70%)

IMPORTANT: To make it easier assess the development of your solution, please store your code on a GitHub repository. I would like to see commits over the semester that demonstrate how your solution evolved. This evidence helps to confirm that your code was developed by you.

The implementation of your code accounts for 40% of the marks available. You must implement your designed solution using an *evolutionary algorithm written in Python Processing library*, a framework for which will be introduced in lectures and labs during the first few weeks of term. You will be supplied with a number of simple *reference* problem instances of the problem on which your code *must* be run in order to benchmark the *effectiveness* of your algorithm(s).

In order to gain a *passing* (40%+) mark for this component, your evolutionary code *must* generate at least one solution that perfectly solves *every* reference instance (i.e., finds a fitness value of zero and perfectly packs every rectangular container with the cylindrical containers). **Your report must clearly state the solution found (the ordering of the crates placed in the container) for each instance.**

For the 60%+ mark band for this component, you must include visualisation of the solutions generated. Please ensure you use the visualisation code supplied in Week 7, as this will ensure consistency, and make it easier for me to verify the correctness of the solutions you generate.

You will also be given a set of more challenging problem instances. For the 70%+ mark band for this component, your code must pass on all these instances (again, finding a perfect packing). **As before, your report must clearly state the solution found for each instance.**

For the higher (80%+) mark band for this component, you must *also* provide implementations of at least *two* other algorithms for the problem.

For the top (90%+) mark band for this component, you should provide the facility to compare the best solutions found by your algorithms by "flipping" through their visualisations.

You will also be marked on the overall quality/presentation of your code (and how it is structured), and how well it corresponds to your design. It must be submitted for verification.

Your write-up accounts for 30% of the marks available. Write a short report (no more than 1500 words), made up of the following:

<u>Results</u>: Demonstrate that your solution works correctly on the reference instances (that is, it gives the expected results, specifying the solutions found), and give your results for the challenging instances (if applicable).

<u>Exploration</u>: Give the result of any investigations you performed with respect to further optimisations of the code, the effect of different parameter settings on the speed/quality of your solution, comparison between your code and other methods, etc (if applicable).

For a mark of 60% in this component you will be expected to include some discussion of the meaning of your results (for example, identifying situations in which your algorithm particularly struggled).

For top marks for this component, you will be expected to perform some experimental comparison of your algorithm with other algorithms for this problem

ASSESSMENT REGULATIONS

You are advised to read the guidance for students regarding assessment policies. They are available online here.

(http://www.northumbria.ac.uk/about-us/university-services/academic-registry/quality-and-teaching-excellence/assessment/guidance-for-students/)

Late submission of work

Where coursework is submitted without approval, after the published hand-in deadline, the following penalties will apply.

For coursework submitted up to 1 working day (24 hours) after the published hand-in deadline without approval, **10% of the total marks available for the assessment** (i.e. 100%) **shall be deducted** from the assessment mark.

Coursework submitted more than 1 day (24 hours) after the published hand-in deadline without approval will be marked as zero, but will be eligible for referral. The full policy can be found here.

Word limits and penalties

If the assignment is within +10% of the stated word limit no penalty will apply.

The word count of your report should be declared on the front page of your assignment

The full Word Limits Policy is available here.

Academic Misconduct

In all assessed work you should take care to ensure that the work you submit is your own. The University takes academic dishonesty and cheating very seriously, and it is your responsibility to ensure that you don't attempt to cheat or become victim to cheating.

There are many different forms of academic misconduct or 'cheating'. Plagiarism is the most common, and both the University library and your academic tutors are able to provide further guidance on proper citation and referencing in your assessed work.

The full Academic Misconduct Policy is available here. Useful guidance for avoiding academic misconduct can be found here.

Both parts will be assessed using the following rubric. The introduction, design, and analysis and exploration components will be assessed by their presentation in your report, while the implementation criteria will consider *both* the relevant section in your report *and* the code that you have submitted.

Section **(with weighting)**	**(0-29)**	**(30-39)**	**(40-49)**	**(50-59)**	**(60-69)**	**(70-79)**	**(80-89)**	**(90-100)**
Lit review (10%)	**No coherent attempt** to describe the problem or existing solutions.	**Limited description** of the problem, no description of existing solutions.	**Basic description** of the problem *and* existing solutions.	**Clear and detailed description** of the problem. Clear and detailed discussion of different solutions.	As for previous band, plus a discussion of several **application domains**.	As for previous band, but also including discussion of the **computational complexity** of the problem.	As for previous band, but including a brief comparative analysis of **different solution methods** (pros and cons).	As for previous band, but showing **exceptional insight** into the nature of the problem and/or its solution.
Analysis/Design (20%)	**No design provided**, or the design presented is significantly incorrect.	**No clear approach, limited design**, lacking significant detail or complexity. Several details of the design may be incorrect or missing.	**Adequate if unambitious approach**, with a reasonable design (expecting diagram only). Some classes may not be correctly structured. Moderately good presentation.	As for previous band, and design is **properly separated out into appropriate classes**.	As for previous band, but with **good pseudo-code presentation** of algorithms implemented.	As for previous band, plus good discussion of any **design decisions** taken.	As for previous band, plus design for **other solution methods** is presented.	An outstanding design, featuring all features needed for previous band, plus **exceptional presentation**.

Section (with weighting)	(0-29)	(30-39)	(40-49)	(50-59)	(60-69)	(70-79)	(80-89)	(90-100)
Implementation (40%) **A case fails if no perfect packing is found.** **Here, "code" refers to the *evolutionary* algorithm produced.**	**Code is missing, fails to run, or is so limited as to provide no clear path to a solution to the problem**. Fails on reference cases.	Code has limited correctness. **Fails on reference cases.** Implementation may not match design.	**Code passes all reference cases.** Implementation broadly matches design. Code presentation adequate.	**Passes all reference cases**, but lacks any visualisation. Code presentation is good. Implementation matches design.	As for previous band, and additionally includes **solution visualisation** (that is, drawing the best solution found). Code presentation is very good.	As for previous band, and code **passes all challenging cases.**	As for previous band, and additionally includes **implementation of at least two other simple algorithms** presented in the lecture (eg. order-based, greedy). Code presentation is excellent.	As for previous band, but the code is presented in such a way that **the user may "flip" between the best solutions found by each method** (perhaps using the cursor keys).
Analysis and exploration. (30%	There has been **no meaningful attempt** to present the results obtained.	There has been ***very limited* attempt** to present the results (perhaps a quick summary).	**Basic presentation of results** of experiments (perhaps simple numerical presentation). Best solutions found for all test cases are clearly stated (for verification).	As for previous band, plus the results of the runs have been **thoroughly presented** (perhaps including graphical plots).	As for previous band, plus results have been **thoroughly *analysed*** (that is, the meaning and implications of the findings have been considered).	As for previous band, plus an investigation made into the **effects (run time, quality, etc.) of varying different parameters**, etc. as specified.	As for previous band, but including a **comparative analysis** with other (non-evolutionary) solution methods.	As for previous band, but with **exceptional presentation and discussion** of ideas.

Northumbria University
NEWCASTLE