Evolutionary Computing
COMP 5660-001/6660-001/6660-D01 – Auburn University
Fall 2025 – Assignment Series 1
Evolutionary Algorithms for the Cutting Stock Problem

Current contributors: Braden Tisdale; Sean Harris; James Browning; Daniel Tauritz, Ph.D.
Previous contributors: Deacon Seals

August 19, 2025

Synopsis

The goal of this assignment set is for you to become familiarized with (I) representing problems in mathematically precise terms, (II) implementing an Evolutionary Algorithm (EA) to solve a problem, (III) conducting scientific experiments involving EAs, (IV) statistically analyzing experimental results from stochastic algorithms, and (V) writing proper technical reports.

The Cutting Stock Problem [`https://en.wikipedia.org/wiki/Cutting_stock_problem`], also known as the Offline 2D Bin Packing Problem, is an extremely important real-world industrial problem belonging to the NP-hard complexity class. Real-world applications include aerospace (for instance, at Boeing's Sheet Metal Fabrication Center), shipbuilding, VLSI design, and manufacturing of shoes, clothing, and furniture.

While highly efficient heuristics have been developed for several variations of the Cutting Stock Problem, particularly hard variations such as those involving irregular shapes have no known heuristics with low approximation ratios (the ratio between the generated solution quality and the optimum). Also, atypical variations sometimes require custom heuristics. One approach to address this is the use of a meta-heuristic such as an Evolutionary Algorithm to directly solve a given variation. Another approach is the use of a hyper-heuristic to automate the design of a custom heuristic. This assignment series takes the former approach. A more in-depth explanation of the problem and our implementation is provided in the accompanying Jupyter notebooks.

These are individual assignments and plagiarism will not be tolerated. You must write your code in Python using the provided assignment framework. You are free to use libraries/toolboxes/etc, except for problem-specific or search/optimization/EA-specific ones. We will allow any standard Python library (e.g., `random` and `json`), in addition to well-known libraries for generic data processing (e.g., `numpy`) or visualization (e.g., `matplotlib`). If you want to use something outside these categories, or anything not provided in the base Conda Linux environment, ask a TA for permission.

Version control requirements

For each assignment you will be given a new repository on [`https://classroom.github.com`]. You will create your repository for each assignment by following a link in the relevant Canvas assignment. **Please view your repository and the README.md file**. It may clear things up after reading this.

Included in your repository is a script named `finalize.sh`, which you will use to indicate which version of your code is the one to be graded. When you are ready to submit your final version, run the command "`chmod 755 finalize.sh && ./finalize.sh`" from your repository then type in your Auburn username.

This will create a text file `readyToSubmit.txt` which lets us know your submission is finished. Commit and push this file to your default branch to submit your assignment. You may commit and push as many times as you like, but your submission will be considered finalized if `readyToSubmit.txt` exists in the default branch after the due date. If you do not plan to submit before the deadline, then you should NOT run the `finalize.sh` script until your final submission is ready. If you accidentally run `finalize.sh` before you are ready to submit, make sure to delete `readyToSubmit.txt` before pushing. Similarly, if it is past the due date and you have already pushed `readyToSubmit.txt`, do not make any further pushes to your repo.

After submission, your latest, pushed, commit to the default branch will be graded if it contains `readyToSubmit.txt`. In order to ensure that the correct version of your code will be used for grading, after pushing your code, examine your repo [`https://github.com`] and verify that you have submitted what you intended to. If for any reason you submit late, then **please notify the TAs when you have submitted.**

Submission, penalties, documents, and bonuses

The penalty for late submission is a 5% deduction for the first 24 hour period and a 10% deduction for every additional 24 hour period. So 1 hour late and 23 hours late both result in a 5% deduction. 25 hours late results in a 15% deduction, etc. Not following submission guidelines can be penalized for up to 5%, which may be in addition to regular deduction due to not following the assignment guidelines.

The code pushed to the default branch after submission will be pulled for grading. Any files created by your assignment must be created in the present working directory or subdirectories within it. All Jupyter notebooks must be completed and submitted with results from running the full notebook. Your submitted code needs to execute as expected, within the EC-env Conda Linux environment, without error. The TAs should not have to worry about any external dependencies or environments. Grading will be based on what can be verified to work correctly as well as on the quality of your source code. You must follow the coding requirements as stated in the syllabus. Always remember that the TAs will thoroughly examine everything by hand, and that your code being easy to read and understand is a substantial part of your grade (*and their sanity*).

Documents are required to be in PDF format; you are encouraged (but not required) to employ LaTeX for typesetting.

Deliverable Categories

There are three deliverable categories, namely:

GREEN Required for all students in all sections.

YELLOW Required for students in the 6000-level sections, bonus for the students in the 5000-level section.

RED Bonus for all students in all sections.

Note that the max grade for the average of all assignments in Assignment Series 1, including bonus points, is capped at 100%. That is, if you received 100%, 100%, 90%, and 120% on the individual assignments, you will receive a 100% for Assignment Series 1.

Assignment 1a: Random Search

In this assignment, you must implement a random search algorithm which generates solutions by uniform random placement of shapes within a predefined rectangular area. Since this is a random search, rather than a more intelligent heuristic search, it is expected to produce poor results. It will serve as a baseline to compare your future EAs against.

You must complete the Jupyter notebook `1a_notebook.ipynb`, a partial Python class implementation, and a report. The notebook will guide you through implementation where you will perform the experiments necessary to create the report. While implementing the specifications in the notebook, think about what data you will need to record in order to write the report described below.

Once you've finished the notebook, you need to write a report. This report should include the following:

- A stairstep plot showing the best fitness progression of the run which resulted in the most-fit solution.
- A histogram showing the distribution of fitness values encountered over the experiment.
- A visualization of the best solution discovered over your 30 runs.
- Statistical analysis comparing the best fitness obtained by each run to the provided mystery data. This should include the sample size, sample means, standard deviations, the test's p-value, alpha, and a brief discussion interpreting the results of the statistical test.

The deliverables of this assignment are:

GREEN 1 your source code and completed notebook

GREEN 2 a PDF document headed by your name, AU E-mail address, and the string "COMP x660 Fall 2025 Assignment 1a", where x reflects the section you are enrolled in, containing your report, including statistical analysis and plot(s)

GREEN 3 files containing any data you analyzed to write your report or generate your plot(s) should be saved to the `data` directory of your repo

Submit all files via GitHub, by *pushing* your latest commit to the default branch, including `readyToSubmit.txt`. The due date for this assignment is 10:00 PM on Sunday, September 7, 2025.

Grading

The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green
Algorithmic	35%
Programming practices, readability, and implementation	35%
Report and plot(s)	20%
Statistical analysis	5%
Notebook questions	5%

Assignment 1b: Evolutionary Algorithm Search

In this assignment, you will implement an EA to search for solutions to the cutting-stock problem. This assignment will utilize the same framework as Assignment 1a, and builds on some of the code you produced in that assignment. Treating the problem of cutting stock as a black-box problem, your EA must generate shape placements and use their evaluated fitness to search for higher-fitness solutions.

In this assignment, you are asked to complete the Jupyter notebook `1b_notebook.ipynb`, several functions outside the notebook which will be reused in later assignments, and a report. The notebook will guide you through implementation of your EA, and will explain how to perform the experiments necessary to create the report. While completing the notebook, think about what data you will need to record in order to write the report described below.

Once you've finished the notebook, you need to write a report. This report should include the following:

- A table of every EA parameter used in your experiment.
- An evals-vs-fitness plot showing the progress of evolution averaged over 30 runs (different from the stairstep plots from 1a).
- A histogram showing the distribution of fitness values encountered over the experiment.
- A visualization of the best solution discovered over your 30 runs.
- Statistical analysis comparing the best fitness obtained by each run to data generated by the algorithm you implemented during 1a. This should include the sample size, sample means, standard deviations, the test's p-value, alpha, and a brief discussion interpreting the results of the statistical test.

This assignment also includes a unit testing suite provided for your benefit. The unit tests will be called by a cell in the assignment notebook, and when a unit test fails, you are expected to perform basic troubleshooting **before trying to contact a TA**. The unit tests are designed so you can determine the cause of failure and remedy it on your own. Note that, due to the stochastic nature of EAs, the unit tests have a small chance of creating false negatives **or** false positives. When in doubt, run them several times to increase your confidence in their accuracy. **Passing these unit tests is part of your assignment grade** – your submission must pass all tests for full points (we will run them several times to ensure accurate testing). You may not make any changes to the unit tests that changes their outcome (pass/fail). You may, however, include things such as `print` statements in them (then run `!pytest -rx` to see their output), **but you must mention this in your report**.

The deliverables of this assignment are:

GREEN 1 your source code and completed notebook

GREEN 2 a PDF document headed by your name, AU E-mail address, and the string "COMP x660 Fall 2025 Assignment 1b", where x needs to reflect the section you are enrolled in, containing your report, including statistical analysis and plot(s)

GREEN 3 files containing any data you analyzed to write your report or generate your plot(s) should be saved to the `data` directory of your repo

YELLOW 1 up to 10% (bonus for COMP 5660 students, required for COMP 6660 students) for an implementation of Stochastic Universal Sampling (see Figure 5.2, page 84 in the textbook) with accompanying report, including all the same components required for the GREEN experiment's report, except that the analysis compares performance against the GREEN experiment. You should make `data/1b/easy_yellow` and `data/1b/hard_yellow` subdirectories and log data there as you did in the GREEN experiment.

RED 1 up to 15% bonus for an implementation of an extra variation (recombination or mutation) operator meaningfully different from the prescribed method(s). What does that mean? You tell us! Any correct algorithm that reasonably fits this definition can be submitted. You must also submit a report including all the same components required for the GREEN experiment's report, except that the analysis compares performance against the GREEN experiment. You should make `data/1b/easy_red` and `data/1b/hard_red` subdirectories and log data there as you did in the GREEN experiment.

Submit all files via GitHub, by *pushing* your latest commit to the default branch, including readyToSubmit.txt. The due date for this assignment is 10:00 PM on Sunday September 21, 2025.

Grading

The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green	Yellow	Red
Algorithmic	30%	60%	60%
Tuning	10%	0%	0%
Passing unit tests	10%	0%	0%
Programming practices, readability, and implementation	20%	20%	20%
Report and plot(s)	10%	15%	15%
Statistical analysis	5%	5%	5%
Notebook questions	15%	0%	0%