# Evolutionary Computing
# COMP 5660-001/6660-001/6660-D01 – Auburn University
# Fall 2025 – Assignment Series 2
# GPac: A Genetic Programming & Co-evolution Approach to the Game of Pac-Man

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

October 17, 2025

## Synopsis

The goal of this assignment series is for you to become familiarized with (I) unambiguously formulating complex problems in terms of optimization, (II) implementing an Evolutionary Algorithm (EA) of the Genetic Programming (GP) persuasion, (III) conducting scientific experiments involving EAs, (IV) statistically analyzing experimental results from stochastic algorithms, and (V) writing proper technical reports.

This assignment series is based on a custom version of Pac-Man, which we call GPac. The problem you will be solving is to employ GP to first evolve a controller for Pac-Man (also referred to as a Pac-Man agent) and subsequently to co-evolve controllers for Pac-Man with controllers for ghosts. This problem is representative of a large and very important class of problems which require the identification of system models such as controllers, programs, or equations. An example of the latter is symbolic regression which attempts to identify a system model based on a limited number of observations of the system's behavior; classic mathematical techniques for symbolic regression have certain inherent limitations which GP can overcome. Employing GP to evolve a controller for Pac-Man is also a perfect illustration of how GP works, while avoiding many of the complications of evolving full-blown computer programs.

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.

## General implementation requirements

For this assignment series you must implement GPac controllers for Pac-Man. You are provided with an implementation of GPac with proper score calculation, spawn mechanics, game-over identification, and world file generation (called a game log in the code). In theory, the fitness of a controller is its expected performance for an arbitrary game instance (i.e., its performance averaged over all game instances). However, as it is computationally infeasible to evaluate a controller over all possible game instances, for the purpose of this assignment it will be sufficient to play a single game instance to completion to estimate fitness. Your code needs to be compatible with the provided GPac implementation and adhere to the specifications of the individual assignments in this series.

## 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 <u>NOT</u> 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.

## Assignment 2a: Random Search

You must implement a random search through valid parse tree space for Pac-Man controllers in GPac. In this assignment, you are asked to complete the Jupyter notebook `2a_notebook.ipynb` and several other Python files as directed by the notebook. Your submission should also contain a report to document the findings of a 10-run experiment. Your report should include the following:

- A stair-step plot showing evaluations vs the progression of the best score seen so far, for the run which produced the highest score overall.
- A histogram showing the distribution of scores across all runs in the experiment.
- The mean and standard deviation of the best scores seen in each run.
- An informal analysis of your agent's performance from watching the visualization of the highest-scoring game. In this informal analysis, we want you to comment on whether or not you think the agent performs well, as well as note any behavioral quirks.

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 2a", where *x* needs to reflect the section you are enrolled in, containing your report, including all components described above

**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

**RED 1** Up to 10% bonus points can be earned by conducting a random search experiment to search for ghost controllers playing against a Pac-Man agent that makes random decisions. This requires adding a new primitive (M, which returns the distance to Pac-Man) and modifying the G primitive to return the distance to the nearest OTHER ghost. These will be required for Assignment 2c, so consider this an opportunity to get a head start. Ghost controllers should be scored the opposite of Pac-Man, i.e., the best ghost controller is the one which found the lowest game score. This experiment should include all the same components as your GREEN experiment.

**RED 2** Up to 15% bonus points can be earned by investigating the use of a hill climber to optimize Pac-Man controllers by iteratively making small changes to the controller and accepting changes that improve fitness. In order to demonstrate that an EA is a reasonable tool for solving a given problem, it is generally more compelling to compare the EA to a simple optimization algorithm such as a hill climber, rather than random search. Showing that the EA outperforms a hill climber indicates that the problem being solved is probably multimodal, and that evolution allows a more effective exploration of the search space. This bonus investigation needs to be documented, including result plots and a new config file, in a separate section of the required document marked as "Hill Climber". The report should include statistical analysis comparing the performance of this experiment with the GREEN experiment, as well as all other components as required 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 November 2, 2025.

**Grading**
The point distribution is as follows:

| Assessment Rubric \ Deliverable Category | Green | Red 1 | Red 2 |
|---|---|---|---|
| Algorithmic | 50% | 50% | 55% |
| Programming practices, readability, and implementation | 35% | 35% | 30% |
| Report and plot(s) | 12% | 12% | 10% |
| Statistical analysis | 3% | 3% | 5% |