Evolutionary Computing
COMP 5660-001/6660-001/6660-D01 – Auburn University
Fall 2024 – Assignment Series 2
GPac: A Genetic Programming & Co-evolution Approach to the Game of Pac-Man

Deacon Seals
Braden Tisdale
Daniel Tauritz, Ph.D.

November 19, 2024

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 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.

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 2024 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 3, 2024.

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%

Assignment 2b: Genetic Programming Search

In this assignment, you will implement a GP search to find high-performance Pac-Man controllers in GPac. You are asked to complete the Jupyter notebook `2b_notebook.ipynb` and several other Python files as directed by the notebook.

You need at minimum to implement support for the following EA configurations, as described in the notebook:

Representation Parse tree

Initialization Ramped half-and-half

Parent Selection Fitness Proportional Selection, k-Tournament Selection with replacement, uniform random selection

Recombination Sub-Tree Crossover

Mutation Sub-Tree Mutation or Point Mutation

Survival Selection Truncation, k-Tournament Selection without replacement

Bloat Control Parsimony Pressure

Termination Number of fitness evaluations

Your submission should also contain a report to document the findings of a 10-run experiment as well as files containing the game log and parse tree from the controller with the highest base fitness from all runs. In your report, include the following:

- A table of every EA parameter used in your experiment.
- A histogram showing the distribution of scores across all runs in the experiment.
- An evals-vs-metrics plot showing the progress of evolution averaged over 10 runs (like assignments 1b-1d).
- Statistical analysis comparing the best fitness obtained by each run to data generated by the random search algorithm you implemented during Assignment 2a. This should include the sample size, sample means, standard deviations, the test's p-value, α, and a brief discussion interpreting the results of the statistical test.
- An informal comparison of the behavior of the best Assignment 2a agent and the best agent from this experiment, by analyzing the visualization of the highest-score game from each algorithm, and by comparing and contrasting their respective highest-scoring parse trees.

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 2024 Assignment 2b", 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) saved to the `data` directory of your repository

YELLOW 1 Up to 20% (bonus for COMP 5660 students, required for COMP 6660 students) can be earned by investigating the impact of casting parsimony pressure as a second objective. That is, instead of using a parsimony pressure penalty function, treat this as a multi-objective problem where your GP search is trying to maximize game score and minimize number of nodes. Combine all Pareto fronts from the final generation of each run into one set of solutions, then use the Pareto front of this set for plotting and behavioral analysis. Plot this Pareto front similarly to Assignment 1d, and conduct an informal analysis of the highest-scoring tree, the smallest tree, and one tree from somewhere in the middle of the Pareto front. You also must conduct a control experiment, which keeps track of the same metrics and produces the same artifacts in your report, but uses the same parsimony pressure mechanism as your GREEN experiment. Finally, you should conduct statistical analysis comparing the global best score found during these two experiments. That is, compare the global best scores, ignoring the parsimony pressure mechanisms from each experiment. Include these results in a separate dedicated section of your report.

RED 1 Up to 25% bonus can be earned by implementing fitness sharing and investigating its impact on population diversity, as measured by phenotypic distance. A distance metric will be provided for you based on the different components of the GPac score function. This experiment should include all the same components as your GREEN experiment, with the following changes: **1)** the evals-vs-metrics plot must *also* visualize the population diversity (calculated as the population's average pairwise distance in phenotype space, as described in the notebook) at each generation, averaged over 10 runs, **2)** statistical analysis and behavioral comparison should be conducted against your GREEN experiment, rather than your random search, and **3)** a second, additional statistical comparison should conducted against your GREEN experiment based on population diversity from the final generation of each run. Include these results in a separate dedicated section of your report.

RED 2 Up to 5% bonus can be earned by investigating having multiple simultaneous Pac-Man agents all employing *identical* controllers, where they all have to die for the game to end, and they share the same score (i.e., there's no competition between the Pac-Man agents). You must add a new primitive, M, returning the distance to the nearest OTHER Pac-Man agent. This experiment should include a similar plot and informal agent analysis as the GREEN experiment (but does not require statistical analysis). Include these results in a separate dedicated section of your report.

RED 3 *If you have completed RED 2*, up to 15% additional bonus can be earned by investigating having multiple simultaneous Pac-Man agents all employing *different* controllers, where they all have to die for the game to end, and they share the same score (i.e., there's no competition between the Pac-Man agents). In addition to the M primitive implemented in RED 2, you must implement an alternative version of the `play_GPac` function called `play_GPac_multicontroller` that accepts multiple Pac-Man controllers (in addition to the typical parameters) and uses each controller to determine moves for a particular Pac-Man agent. Evolution should utilize a single population of controllers, and fitness should be re-assessed each generation with stochastic controller pairing such that all individuals (including adults!) play at least one game per generation, no individual plays more than one game more than any other individual, and the base fitness of an individual is determined by averaging the scores obtained in each game the individual played in during the current generation. Each game played should be counted as one fitness evaluation. This experiment should include a similar plot and informal agent analysis as the GREEN experiment (but does not require statistical analysis). Make sure you analyze all of the parse trees that participated in the highest-scoring game, and compare them to what you observed during the RED 2 experiment. Include these results in a separate dedicated section of your report.

RED 4 Up to 10% bonus points can be earned by investigating the evolution of a controller that controls all ghosts and plays against the default Pac-Man strategy. 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. This investigation should use negative game score as the base fitness metric. This experiment should include a similar plot and informal agent analysis as the GREEN experiment (but does not require statistical analysis). Include these results in a separate dedicated section of your report.

RED 5 *If you have completed RED 4*, up to 15% additional bonus can be earned by investigating having multiple simultaneous ghost agents all employing *different* controllers, against the default Pac-Man strategy. All ghosts share the same fitness for a particular game (i.e., there's no competition between the ghost agents). In addition to the components implemented in RED 4, you must implement an alternative version of the `play_GPac` function called `play_GPac_multicontroller` that accepts multiple ghost controllers (in addition to the typical parameters) and uses each controller to determine moves for a particular ghost agent. Evolution should utilize a single population of controllers, and fitness should be re-assessed each generation with stochastic controller pairing such that all individuals (including adults!) play at least one game per generation, no individual plays more than one game more than any other individual, and the base fitness of an individual is determined by averaging the scores obtained in each game the individual played in during the current generation. Each game played should be counted as one fitness evaluation. This experiment should include a similar plot and informal agent analysis as the GREEN experiment (but does not require statistical analysis). Make sure you analyze all of the parse trees that participated in the highest-scoring game, and compare them to what you observed during the RED 4 experiment. Include these results in a separate dedicated section of your report.

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 17, 2024.

Grading
The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green	Yellow	Red 1	Red 2-5
Algorithmic	50%	55%	50%	60%
Tuning	10%	0%	0%	0%
Programming practices, readability, and implementation	20%	25%	25%	20%
Report and plot(s)	15%	15%	20%	20%
Statistical analysis	5%	5%	5%	0%

Assignment 2c: Competitive Co-evolutionary Search

For this assignment, you need to implement a competitive co-evolutionary algorithm [1, Section 15.3] that can co-evolve GP controllers for Pac-Man and GP controllers for the Ghosts, where one fitness evaluation is counted as the average score of full games played between competing controllers. In this assignment, Ghosts within a given game of GPac will share a singular controller to determine their individual actions. The base fitness of a Ghost controller should be calculated as negative game score. Your competitive co-evolutionary algorithm will consist of two separate populations, one for the Pac-Man controllers and one for the Ghost controllers. The recommended GP approaches are the same for Pac-Man and Ghost controllers as in Assignment 2b, though each species of controller will use different sets of GP primitives. Specifically, Ghosts require the same primitives as Pac-Man except for the following new or modified terminal/sensor primitives:

G Manhattan distance to nearest other ghost (i.e., excluding itself)

M Manhattan distance to nearest Pac-Man

In this assignment, you are asked to complete the Jupyter notebook `2c_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. In your report, include the following:

- A table of every EA parameter used in your experiment.
- Plots showing the distributions of tree size and tree height versus evaluations across both populations in all runs.
- Box plots showing the performance of the five best-base-fitness Pac-Man and Ghost controllers from the end of each run against a common set of 100 randomly-generated Ghost controllers or Pac-Man controllers respectively.
- For the single best-base-fitness Pac-Man and Ghost controller from the end of the first five runs, histograms showing the distribution of scores from the above games against randomly-generated controllers.
- Qualitative analysis of the exhibition game played between the two "best" controllers from the above games. Comment on any notable behaviors and describe the overall effectiveness of the agents' strategies.

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 2024 Assignment 2c", 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) saved to the `data` directory of your repository

YELLOW 1 Up to 10% (bonus for COMP 5660 students) can be earned by additionally analyzing the performance of your best Pac-Man and Ghost controllers against each other using the methodology described in the YELLOW section of the notebook, by splitting half of each group into a test set and producing a matrix plot and histograms. Include these results in a separate dedicated section of your report, and answer the questions in the notebook.

Note on RED deliverables The RED deliverables expand upon those of Assignment 2b by exploring scenarios with multiple Pac-Mans (RED 1 and 3), and investigating the use of multiple separate, cooperating controllers for the Ghost and multiple Pac-Mans (RED 2 and 3).

In scenarios involving multiple Pac-Man players in a game (RED 1 and 3), you must add a new primitive, M, returning the distance to the nearest OTHER Pac-Man agent, and ensure that Ghost terminals behave correctly if multiple Pac-Man characters exist. All Pac-Man characters have to die for the game to end, and they share the same score even if they have separate controllers (i.e., there's no competition between the Pac-Man players). Similarly, all Ghosts share the same fitness for a particular game even if they have separate controllers (i.e., there's no competition between the ghost players).

To enable the use of separate, cooperating controllers, you should implement the *play_GPac_multicontroller* function described in Assignment 2b. Evolution should utilize a single population of controllers per controller type, where multiple cooperating individuals are drawn from the population and assigned fitness for each evaluation. Fitness should be re-assessed each generation with stochastic controller pairing such that all individuals (including adults!) play at least one game per generation, all individuals play in the same number of games each generation (off by at most one), and the base fitness of an individual is determined by averaging the scores obtained in each game the individual played in during the current generation (if they played in more than one). Each game played should be counted as one fitness evaluation.

RED 1 Up to 5% bonus points can be earned by investigating competitive co-evolution using multiple simultaneous Pac-Man characters all employing *identical* controllers and Ghost characters sharing a common controller (as in GREEN). This experiment should include all the same report components as the GREEN experiment, except for the boxplots. Include these results in a separate dedicated section of your report.

RED 2 Up to 20% bonus points can be earned by investigating competitive co-evolution of a controller for a single Pac-Man and multiple simultaneous Ghosts all employing *different* controllers. This experiment should include all the same report components as the GREEN experiment, except for the boxplots. In this analysis, the Ghosts should use exhibition teams formed of the three highest-base-fitness Ghost controllers from the final generation of each run, rather than a single controller. Include these results in a separate dedicated section of your report.

RED 3 Up to 20% bonus points can be earned by investigating competitive co-evolution of multiple simultaneous Pac-Man characters all employing *different* controllers and multiple simultaneous Ghosts all employing *different* controllers. This experiment should include all the same report components as the GREEN experiment, except for the boxplots. In this analysis, each population should use exhibition teams formed of the highest-base-fitness controllers from the final generation of each run, rather than a single controller. Include these results in a separate dedicated section of your report.

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 Friday December 6, 2024.
This assignment also has a unique late policy. The cumulative late penalty for submitting each day before 10:00 PM is:

- Saturday the 7th: -5%
- Sunday the 8th: -15%
- Monday the 9th: -30%
- Tuesday the 10th: -50%
- Wednesday the 11th: -80%

Any submissions after 10:00 PM on Wednesday, December 11th will not be graded and will receive a 0%.

Grading

The point distribution per deliverable category is as follows (note that Assignment 2c is worth twice the points as assignments 2a & 2b):

Assessment Rubric \ Deliverable Category	Green	Yellow	Red
Algorithmic	45%	30%	45%
Tuning	5%	0%	0%
Programming practices, readability, and implementation	30%	30%	35%
Report and plot(s)	20%	30%	20%
Notebook questions	0%	10%	0%

References

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing.* Second Edition, Springer-Verlag, Berlin Heidelberg, 2015, ISBN 978-3-662-44873-1.

[2] Dave Cliff and Geoffrey F. Miller, *Tracking the red Queen: Measurements of Adaptive Progress in Co-Evolutionary Simulations.* In Advances in Artificial Life, Lecture Notes in Computer Science, Volume 929, Pages 200-218, Springer-Verlag, Berlin Heidelberg, 1995, ISBN 978-3-540-59496-3. `http://www.cs.uu.nl/docs/vakken/ias/stuff/cm95.pdf`

[3] Mouret, Jean-Baptiste, and Jeff Clune, *Illuminating search spaces by mapping elites* arXiv preprint arXiv:1504.04909 (2015). `https://arxiv.org/abs/1504.04909`