Evolutionary Computing
COMP 5660-001/6660-001/6660-D01 – Auburn University
Fall 2022 – Assignment Series 1
Automated Bridge Design

Deacon Seals
Braden Tisdale
Sean Harris
Daniel Tauritz, Ph.D.

August 19, 2022

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 problem you will be solving is the design of bridges that are assessed via a physics simulation. This problem is representative of problems which require the selection or generation of complicated solutions that optimize an objective function which may not be easily solved with a heuristic-based approach, and where an exhaustive search is infeasible. This is a large class of problems which comes up frequently in the real world, and EAs are one of the most commonly-used optimizers for these problems.

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.

Problem statement

In this assignment you will implement bridge generation using a physics simulation of moderate fidelity. If you're familiar with bridge building games such as World of Goo or any of the games in the Bridge Constructor franchise, you may find the problem in this assignment series to be similar. Automated design is a hallmark application of evolutionary computing that can generate art, solve engineering tasks, and even write code. For this assignment series, your task is to generate structurally sound bridges to support a roadbed spanning a gap using parameterized geography, materials, and load points. You will be provided with all the code required for the simulation of your bridges.

Your algorithm will designate a fixed number of (x, y) coordinate pairs as a solution representing a specific bridge design. The coordinate pairs represent nodes (connection points), which are automatically connected by frame elements (edges) if within a set distance from one another. These frame elements use set material and cross-section characteristics specified by a problem instance. During the physics simulation,

frame elements will stretch, compress, and bend slightly under load until the forces in the bridge are at equilibrium, allowing their structural integrity to be determined. In addition to the coordinate pairs specified by your algorithm, there are pre-existing nodes at fixed locations that serve as the foundation of the bridge and connection points along the roadbed.

A solution's fitness is evaluated with a function we have provided for you. This function constructs the bridge represented by your solution. The bridge is then tested using a physics simulation by applying an increasingly-heavy load to points within the roadbed until the bridge fails. Elements of the bridge will fail if their tensile, compressive, or bending stresses exceed the maximum amounts supported by their material and geometry. The bridge is considered to have failed if any element fails. The function then returns the force at which the bridge failed. Your algorithm's goal is to find a solution that maximizes the value returned by this function, in order to design the strongest bridge possible under these conditions.

It is possible that your algorithm chose coordinate pairs that do not represent a valid solution. Namely, if your solution contains nodes at the exact location of provided nodes (those belonging to the roadbed or support locations) or otherwise has characteristics that prevent simulation then it will be declared invalid. If the bridge is invalid, the function will return an arbitrarily-low value to designate that the solution does not satisfy design constraints. Nodes that are disconnected from the fixed points (e.g., a cluster of nodes floating in the air) may, however, be ignored for some assignments to allow for the easier discovery of solutions and for the reduction of points in the solution.

Version control requirements

For each assignment you will be given a new repository on [`https://classroom.github.com`]. **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 base 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

You must implement a random search algorithm which generates bridges by uniform random placement of coordinates within a predefined rectangular area.

In this assignment you are asked to complete the Jupyter notebook `1a_notebook.ipynb`, part of a Python class, and a report. The notebook will guide you through implementation where you will perform the experiments necessary to create the report. While implementing the changes listed 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 a stairstep plot showing evals vs fitness of the run which resulted in the most-fit solution. This report should also include statistical analysis (F-test and t-test) comparing the fitness obtained by each run to the sample data provided in your repository, and a brief discussion interpreting the results of the statistical tests. This sample data can be found in `data/mysteryAlgorithmResults.txt`.

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 2022 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 in a format that can be easily understood by the TAs (if you think you should include instructions on how to interpret your data, then you should!)

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 4, 2022.

Grading
The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green
Algorithmic	30%
Logging and output files	15%
Programming practices, readability, and implementation	15%
Report and plot(s)	20%
Statistical analysis	20%

Assignment 1b: Evolutionary Algorithm Search

Coming soon!

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.