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.

January 9, 2023

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

You must implement an EA which generates bridges. This assignment will utilize the same framework that was utilized for Assignment 1a, and builds on some of the code you produced in that assignment. Treating the problem of bridge design as a black-box problem, your EA must generate bridges 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 an evals-vs-fitness plot showing the progress of evolution averaged over 30 runs. This report should also include statistical analysis (F-test and t-test) comparing the best fitness obtained by each run to data generated by the algorithm you implemented during 1a, and a brief discussion interpreting the results of the statistical tests. The report must also include every configured parameter used in your experiment.

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 x66y Fall 2022 Assignment 1b", where x and y need 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), 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!)

YELLOW 1 Up to 10% (bonus points for COMP 5660 students) for an implementation of Stochastic Universal Sampling (see Figure 5.2, page 84 in the textbook) with accompanying report and statistical analysis comparing the impact on performance against another parent selection technique. You should create a new config file (i.e., create config file(s) in addition to `green1b_config.txt` such that each file contains the configurations used in your comparison).

RED 1 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. Describe your design in your report and provide statistical analysis comparing performance against another recombination operator. Students can earn up to 15% for this investigation. You should create a new config file (i.e., create config file(s) in addition to `green1b_config.txt` such that each file contains the configurations used in your comparison).

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 18, 2022.

Grading
The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green	Yellow	Red
Algorithmic and attempt at parameter tuning	40%	60%	60%
Passing unit tests	15%	0%	0%
Logging and output files	5%	5%	5%
Programming practices, readability, and implementation	10%	10%	10%
Report and plot(s)	15%	10%	10%
Statistical analysis	15%	15%	15%

Assignment 1c: Constraint Satisfaction and Multi-Objective Evolution

Using the EA you created in Assignment 1b, you must implement two new EAs. The first new EA is a constraint satisfaction EA that considers a new design constraint for allowing boats and other traffic to pass through the bridge. The second new EA is a multi-objective EA (MOEA) that considers both the weight your bridge can support and the total length of material needed to construct the bridge.

In this assignment you are asked to complete both of the Jupyter notebooks `1c0_notebook.ipynb` and `1c1_notebook.ipynb`, several functions outside the notebooks, and a report. The notebooks will guide you through implementation of your EAs, and will explain how to perform the experiments necessary to create the report. While completing the notebooks, think about what data you will need to record in order to write the report described below.

Once you've finished the notebooks, you need to write a report. The reporting requirements for each algorithm are different:

Constraint Satisfaction EA For each experiment, an evals-vs-raw-fitness plot showing the progress of evolution averaged over 30 runs, statistical analysis comparing the (F-test and t-test) comparing the best fitness obtained by each run to data generated by your EA employing both constraint satisfaction approaches described in `1c0_notebook.ipynb`, and a brief discussion interpreting the results of the statistical tests. Note that for statistical analysis with the constraint satisfaction EA, you should use the best raw fitness.

Multi-Objective EA For each experiment, an evals-vs-fitness plot showing the progress of evolution for each objective averaged over 30 runs. This report should also include a plot of the best Pareto front found in any run, where we count Pareto front P1 as better than Pareto front P2 if the proportion of solutions in P1 which dominate at least one solution in P2 is larger than the proportion of solutions in P2 which dominate at least one solution in P1. Visualize the bridges of the individuals within the best Pareto front and comment on how they differ. **Note**: the green deliverables only require a single MOEA experiment without statistical analysis.

The report must also include every configured parameter used in your experiment or at least a reference to which file contains the parameters for each experiment discussed.

The same unit tests that were provided in Assignment 1b are included in your 1c repository. While failing unit tests will no longer count directly against your grade (they are not a deliverable), you will still be penalized if any of the algorithms implemented during 1b have errors. The unit tests can supplement your 1b feedback report in helping you resolve any remaining errors.

The deliverables of this assignment are:

GREEN 1 your source code and completed notebooks

GREEN 2 a PDF document headed by your name, AU E-mail address, and the string "COMP x66y Fall 2022 Assignment 1c", where x and y need 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), in a format that can be easily understood by the TAs (if you think you might should include instructions on how to interpret your data, then you should!)

YELLOW 1 Up to 10% (bonus points for COMP 5660 students) for implementing crowding as a diversity preservation mechanism. In order to maintain compatibility with the MOEA implementation prescribed within the notebook, crowding should be implemented such that a value [0,1) is added to all individuals in a level of non-domination proportional to that individual's crowding distance relative to others in their non-domination level. Using a diversity metric of your choice, record the diversity of the Pareto front from the last generation of each run and perform statistical analysis (including F-tests

and t-tests) to compare against diversity observed in your GREEN deliverables. Plot the best Pareto front with crowding and compare with the front plotted from the GREEN deliverables to informally identify which algorithm performed better.

RED 1 Up to 10% bonus for implementing constraint satisfaction by casting it as a multi-objective problem that considers unpenalized fitness and negative count of constraint violations as the two maximization objectives. Perform statistical analysis (using F-tests and t-tests) to compare performance with your best Assignment 1c constraint satisfaction implementation using raw fitness. Report on your findings as appropriate.

RED 2 Up to 10% bonus for implementing a repair function that modifies an individual's genotype such that the constraints are enforced (feel free to observe how constraint violations are checked in the `constraint_satisfaction_simulation` function). Perform statistical analysis (using F-tests and t-tests) to compare performance with your best Assignment 1c constraint satisfaction implementation using raw fitness. Report on your findings as appropriate.

RED 3 The mayor of Auburn took a recent trip to the Golden Gate Bridge and has been inspired to request landmark-worthy bridge designs that make trade-offs between strength, required bridge material, and height. Up to 10% bonus for implementing and experimenting with 3-objective search using weight, negative bridge material, and bridge height. Find the best 3-objective Pareto front generated by a 30-run experiment using the criteria described earlier. Include in your report a 3D plot of this Pareto front. Using the individuals in this Pareto front and non-domination sort, form 2-objective Pareto fronts considering only the objectives shared with GREEN/YELLOW, respectively, and compare against the best Pareto fronts from those deliverables through informal analysis of Pareto front plots. Include these plots and your comparison in your report.

RED 4 We alluded in the notebook that rigorous analysis of multi-objective results is complicated! For up to 15% bonus, perform a rigorous comparison of the two MOEA configurations of your choice (that use the same objectives) using the hypervolume indicator metric as calculated using normalized values for each objectives. Include in your report a detailed description of the analysis technique as well as your results.

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 October 2, 2022.

Grading
The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green	Yellow	Red
Algorithmic	45%	60%	60%
Logging and output files	5%	5%	5%
Programming practices, readability, and implementation	20%	10%	10%
Report and plot(s)	15%	10%	10%
Statistical analysis	15%	15%	15%

Assignment 1d: Island-Model EA

Using the EA you created in assignments 1b and 1c, you must implement an island-model EA.

In this assignment you are asked to complete the Jupyter notebook `1d_notebook.ipynb`, several functions outside the notebook, 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 an evals-vs-fitness plot showing the progress of evolution across all islands (e.g., calculate average and best fitness values across all islands) averaged over 30 runs. This report should also include statistical analysis (F-test and t-test) comparing the island topologies you implemented during this assignment, and a brief discussion interpreting the results of the statistical tests. The report must also include every configured parameter used in your experiment.

The deliverables of this assignment are:

GREEN 1 your source code and completed notebooks

GREEN 2 a PDF document headed by your name, AU E-mail address, and the string "COMP x66y Fall 2022 Assignment 1d", where x and y need 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), in a format that can be easily understood by the TAs (if you think you might should include instructions on how to interpret your data, then you should!)

YELLOW 1 Up to 10% (bonus points for COMP 5660 students) for implementing an island-model MOEA using the MOEA components you implemented in Assignment 1c. Fitness should be calculated by performing non-domination sort on each island such that non-domination sort only compares individuals in the same population/island. This should provide interesting island-specific fitness definitions! Note that you should be able to reuse your island model implementation and the only modifications required should be to your main search function defined in the notebook. Your report should include an evals-vs-fitness plot showing the progress of evolution for each objective across all islands averaged over 30 runs. No formal statistical analysis is required.

RED 1 Up to 10% bonus for implementing an island topology meaningfully different from the prescribed topologies in the notebook. Experiment with your new topology using the single-objective task in the GREEN deliverables and perform statistical analysis (using F-tests and t-tests) to compare performance with your best configuration observed in this assignment. Report on your findings and methodology as appropriate.

RED 2 Up to 15% bonus for implementing and experimenting with heterogeneous island configurations (i.e., where the islands in your topology use different parameters). Experiment using the single-objective task in the GREEN deliverables and perform statistical analysis (using F-tests and t-tests) to compare performance with your best configuration observed in this assignment. Report on your findings and methodology as appropriate.

RED 3 We prescribe in the YELLOW 1 deliverable that multi-objective fitness should be calculated via non-domination sort on each island to generate island-specific definitions of fitness. In addition to completing YELLOW 1, for up to 10% bonus, perform an experiment where fitness is assigned by performing non-domination sort with all individuals across all islands. Your report should include an evals-vs-fitness plot showing the progress of evolution for each objective across all islands averaged over 30 runs. Provide an informal comparison with the per-objective measurements obtained by the best MOEA configuration found for YELLOW 1. No formal statistical analysis is required.

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 October 16, 2022.

Grading
The point distribution is as follows:

Assessment Rubric \ Deliverable Category	Green	Yellow	Red 1	Red 2	Red 3
Algorithmic	45%	50%	50%	50%	60%
Logging and output files	5%	5%	5%	5%	5%
Programming practices, readability, and implementation	20%	20%	10%	10%	10%
Report and plot(s)	15%	25%	15%	15%	25%
Statistical analysis	15%	0%	20%	20%	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.