CS4205 - Evolutionary Algorithms

Assignment 2

This document describes the final practical assignment of the TU Delft course Evolutionary Algorithms (CS4205). This assignment will count for 30% of the final grade and can be done on one of the different topics listed below, each with a respective supervisor:

- *Evolutionary Van Gogh* (Single-Objective Discrete Optimization)
  Supervisor: Damy Ha
- *Genetic Programming for Symbolic Regression* (Genetic Programming)
  Supervisor: Johannes Koch
- *Circles in a Square* (Single-Objective Real-Valued Optimization)
  Supervisor: Anton Bouter
- *Evolving a Lunar Lander with Genetic Programming*
  Supervisor: Thalea Schlender

A more detailed description for each of these assignments is provided in the appendix.

The basic principles of the assignment are the same across the topics. This assignment is expected to be done in *groups of 5 students*. You are supposed to self-arrange into groups, there is a discussion forum for group finding on Brightspace to help you with this. Groups of less than 5 students may receive additional members or be split up at the discretion of the teaching staff.

Once you have a group, each of you needs to sign up on the same group in the Brightspace group interface. We will assign the topics to groups using a preference-maximization approach. For this, prior to the start of this assignment, each group of students must denote their relative preference for each of the 4 possible exercises in the Brightspace survey. The deadline for signing up for a group *and* for indicating your topic preference is **Sunday, May 18, 2025, 23:59**.

Requirements

Weekly progress meetings will be held between the group and the designated supervisor. During each of these meetings, you are required to give an update on your progress and you are given the opportunity to ask questions. For the scheduling of these meetings, each group will be individually contacted by their designated supervisor.

The requirements of the final deliverable are as follows. Each group must hand in the following *deliverables* on Brightspace in a single zip file prior to the deadline of **Sunday, June 8, 2025, 23:59**:

- All source code used to produce experimental results in a *zip* file.
- Presentation slides (PDF/PPTX format) to be presented in a **9** minute final presentation on **June 11** or **June 12, 2025**. The schedule for the final presentations will follow at a later date.

Grading

A **complete** submission of the deliverables is required and the source code should run (else, it is a fail). The final grade is based on the following criteria, for which a rubric will be made available at a later stage:

- [**60%**] Content
- [**20%**] Presentation
- [**20%**] Defense

Evolutionary Van Gogh

CS4205: Evolutionary Algorithms - Assignment 2
Supervisor: Damy Ha (d.m.f.ha@lumc.nl)
Topic: Single-objective Discrete Optimization

Figure 1: Vincent van Gogh's "Wheat Field with Cypresses" (1889). *Google Art Project.*

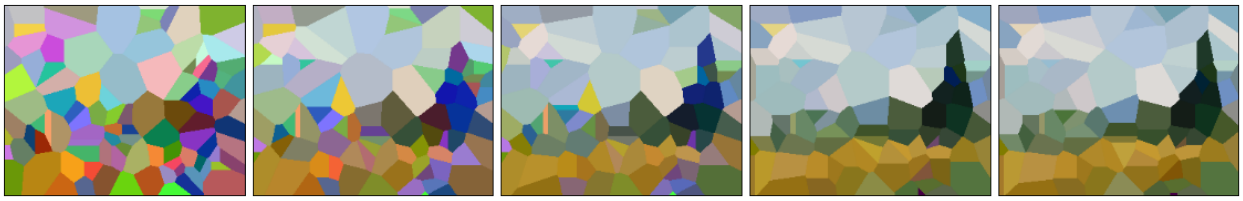

Figure 2: One run of a Genetic Algorithm imitating Figure 1 with Voronoi cells. Shown are the best imitations at various generations.

1 Problem Definition

Genetic Algorithms (GA) excel at solving many optimization problems. In this assignment we will be considering a rather unusual optimization problem: imitating art. The optimization problem in this assignment is to find an approximation of an existing painting, given a (limited) set of primitives. In order to "draw" shapes and therefore mimic an existing painting, we will be using Voronoi diagrams. To make a Voronoi diagram, we start by placing a number of points in an image space, where each point has a color associated with it. We then color the space as follows: For each pixel in the image, we give it the color of the closest point (by Euclidean distance). This partitions the space into regions (also called *cells*) around each point having the associated color of that point.

With this drawing strategy, we let the GA evolve imitations of the painting, using the conventional mechanisms of a GA (mutation, recombination, selection). A particular Voronoi diagram instance is encoded in an individual's genotype as follows: Per point, a string of 5 integers (x, y, r, g, b) is stored describing the point's position ((x, y), ranging from 0 to the image's width or height -1) and color ((r, g, b), ranging from

0 to 255). These point information strings are concatenated to form a long string with all the points of the diagram, forming the genotype.

For the fitness of an individual, we need to establish how well it imitates the reference painting. We use `imgcompare`[1] for this, which subtracts the corresponding pixels in the imitation and the reference image from each other, and then assigns a score to this difference image (lower is better). Given this fitness, we use P+O tournament selection with size 4 to form the next generation.

2 Goal of the Assignment

In this assignment, we want you to experiment with Genetic Algorithms through a hands-on example. You will start out with an existing implementation of a GA imitating Van Gogh's "Wheat Field with Cypresses" oil painting (Figure 1). Note that we are not looking for a different way of drawing. We want to make improvements to the underlying components of the GA that lead to measurable improvement in the GA's performance. A good first step can be to consider here is the current Black-Box Optimization (BBO) approach, and whether elements of Gray-Box Optimization (GBO) can be used.

3 Materials

You are provided with a working version of the algorithm, using *one-point* crossover and a *random* initialization of points across image space. We want you to extend this implementation as described below. For this, you can use the Jupyter Notebook located in the provided ZIP file, which includes code to plot the progress of optimization over time and to display intermediate images.

It's recommended to execute all experiments with a point budget of either **50** or **100** points and a generation budget of **500**. A run can take 3-5 minutes, so take this into account when planning your experiments. However, feel free to explore other experimental configurations if you think it makes sense. You can experiment with available hyperparameters, such as different population sizes and mutation strength.

4 Deliverables

You are expected to deliver a presentation showcasing the mechanisms and experiments you have conducted. In addition, prior to your presentation, you are expected to digitally deliver a single-sided A4 of the experiments you have conducted. Finally, submission of your code is mandatory.

We will meet every week to discuss progress. Below is a recommended list of tasks to guide your work. While progress on each task will not be tracked or graded, staying on track with the weekly goals should make our meetings more useful and help you avoid overload as the deadline approaches. Feel free to work ahead if you would like, these weekly deliverables are just the baseline.

Additional work, such as creative ideas or extra experiments, is encouraged and will be rewarded. However, before you start constructing the most groundbreaking mechanism the world has ever seen, make sure that your experiments are well-designed. Well-structured experimentation is essential for assessing the effectiveness of your approach and is a core learning objective of this assignment.

4.1 Recommended weekly goals

Week 1

- Before our first meeting, get the provided code installed and running on your machines.
- Hypothesize which (simple) mechanism would improve the GA and why.

[1] https://pypi.org/project/imgcompare/

Week 2

- Implement at least one new feature.
- Set-up an experiment to test your feature(s).
- Evaluate why it works or does not work, based on plots that you can show at the meeting. We expect you to not only tell us what you observe but to also *explain* why you think you observe this.
- Prepare a full set of (at least) 3 hypotheses that you want to test, corresponding to different feature ideas and hyperparameter settings you want to try. Make sure at least 2 of these hypotheses consider features you have developed.

Week 3

- Finalize implementation and experiments.
- Prepare your presentation

5 Related Work

1. Darrell Whitley. A genetic algorithm tutorial (up to Chapter 4). *Statistics and computing*, 4(2):65–85, 1994. *(Also part of the mandatory reading material.)*
2. Sebastian Proost. Minimalist Art Using a Genetic Algorithm. 2020. `http://blog.4dcu.be/programming/2020/02/10/Genetic-Art-Algorithm-2.html`

Genetic Programming for Symbolic Regression

CS4205: Evolutionary Algorithms - Assignment 2
Supervisor: Johannes Koch (johannes@cwi.nl)
Topic: Single and Multi-objective Genetic Programming

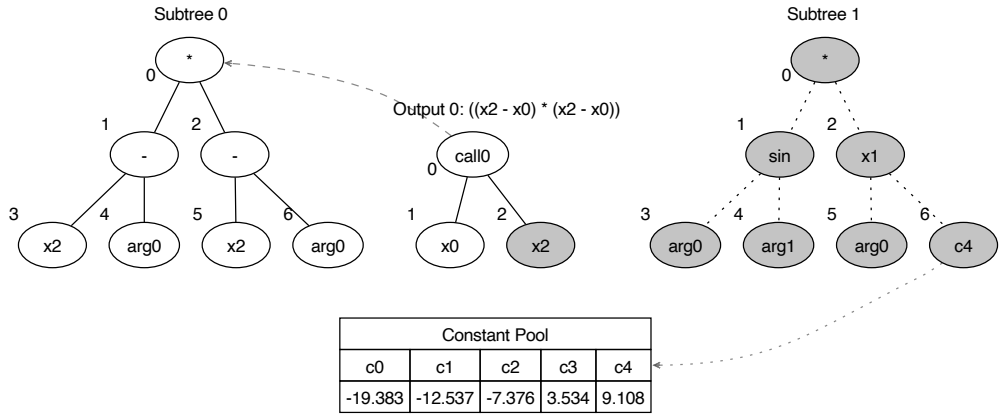

Figure 1: An individual showing the used template representation. Gray nodes are not reachable from any output root node, and thus do not affect fitness.

1 Problem Definition

In recent years, the field of eXplainable AI (XAI) has received increased attention, especially for use cases where AI models can affect lives and livelihoods. Given a dataset and a library of atomic functions such as $\{+, -, \times, \div, \sin\}$, symbolic regression (SR) is the task of finding an interpretable expression that best describes the relation between one (output) variable and other (input) variables. Compact SR models (i.e. expressions) are interesting from the perspective of XAI and interpretable ML (IML), as they are readable and therefore have the potential to be humanly understandable. In this assignment, your goal is to improve a provided SR method based on a mix of genetic programming and differential evolution.

The GP implementation features various variation operators, support for multi-objective optimization, linear scaling and flexible templates. Compared to standard tree-based GP, this implementation uses a template that defines the fixed tree structure and a constant pool that holds all available constants as shown in Figure 1. Each tree node corresponds to a discrete decision variable that can take on all values allowed by the arity of the node and each constant is a continous decision variable. There can be multiple output trees and subtrees/subfunctions the output trees can make use of. Note that in literature these are often referred to as automatically defined trees/automatically defined functions.

In this project, you are free to work on your own ideas as long as it involves modifying the GP algorithm in some way and should definitely include experiments to see if your changes lead to improved performance over the baseline.

2 Goal of the Assignment

The main objective of this assignment is to gain hands-on experience working with evolutionary computation. In this project, you are free to choose the direction you want to explore, as long as the project focus is on the evolutionary algorithm - it is an evolutionary algorithms course after all. This could include any of the following:

- Trying out new variation operators (e.g. a simplification operator)
- Improving constant optimization
- Improving the multi-objective performance by using an archive
- Improving the re-use of subtrees (possibly with multiple outputs)
- Improving search efficiency for large numbers of decision variables (many input features, operators, constants,...)
- ...

3 Materials

You are provided with a working version of the algorithm, including various initialization, selection and variation operators discussed in the lectures. In addition, basic utilities to run multiple experiments and to make first comparisons are provided based on [1]. To support multi-objective optimization, selection using non-dominated sorting as per NSGA-II[2] has been implemented. Other than that, the field guide to GP ([3], Lecture 4 reading material) might be useful.

It is recommnded to execute all (single-objective) experiments with a population size of at least **25**, a computational budget allowing for **500** generations and at least **10** repetitions per problem/method combination. Note that runs can take a few minutes, take this into account when planning your experiments.

4 Recommended Project Timeline

Week 1

- Get the provided code installed and running.
- Come up with potential ideas on what you want to do. Already start thinking about why these ideas are interesting and how you can implement and test it.
  Note that as the course is on evolutionary algorithms, your ideas should include modifying the existing algorithm in some way.
- As a group, decide on what you want to do for the final deliverable and plan accordingly. Your experiment setting should include cross-validation, multiple problems, and several runs per fold. If this is not the case, motivate why this is not needed for your project (i.e. you need to convince us during the final presentation).

Week 2

- Implement at least one new feature.
- You should have results for the baseline you want to compare to.
- Evaluate why the modification works or not based on results (e.g. plots) you can present during the meeting. Note that you should be able to reason about *why* you observe these results.

Week 3

- Work on the final presentation.
- At the end of the week, submit the final deliverable (code and presentation). We expect you to hand in all code made in the course of the assignment.

5 Related Work

[1] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems* (2021).

[2] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 182–197.

[3] Poli, R., Langdon, W. B., and McPhee, N. F. *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

Solving Circles in a Square

CS4205: Evolutionary Algorithms - Assignment 2
Supervisor: Anton Bouter (bouter@cwi.nl)
Topic: Real-Valued Single-Objective Optimization

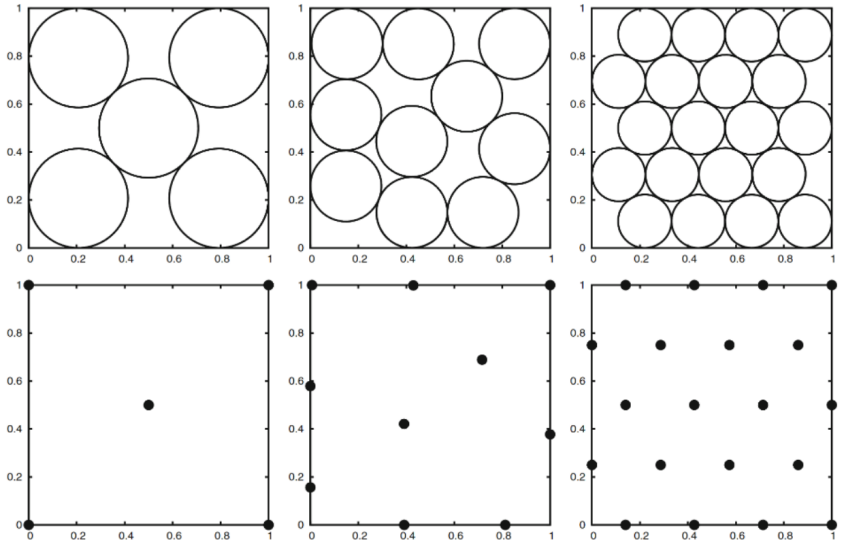

Figure 1: Top row: optimal circle packings for 5, 10, and 20 circles. Bottom row: corresponding point scatterings [1].

1 Problem Definition

This final assignment considers a scalable and real-valued optimization problem called "Packing n Circles in a Square (CiaS, see, e.g., [1])." This problem has many applications in real-world optimization problems like loading containers, communication networks, facility dispersion/layout, etc. Even though CiaS can be compactly formulated and efficiently evaluated, it is not trivial to solve and has interesting characteristics that differ from many other commonly used benchmark problems. For at least 2 circles, the CiaS problem is equivalent to finding an optimal scattering of points in the unit square. To obtain the optimal scattering of points, the smallest distance between any pair of points has to be maximized. Examples of optimal circle packings and their corresponding point scatterings are shown in Figure 1. The fitness function for the point scattering problem is as follows:

$$f_{scatter}(\boldsymbol{x}) = \min_{0 \leq i < j < \ell/2} ||(x_{2i}, x_{2i+1}) - (x_{2j}, x_{2j+1})||$$

2 Goal of assignment

To obtain state-of-the-art performance on CiaS is not trivial, as specific properties of the CiaS problem must be considered and exploited. It would be interesting to see to what extent EAs can be tuned to CiaS to also obtain state-of-the-art results with an EA. You will be doing so by improving one of the provided EAs and perform a performance comparison before and after it has been tuned for CiaS.

An especially useful resource is the Packomania website [2]. On this website a large list of the optimal, or best known, packings is maintained. Currently, the list goes up to 9996 circles, with some intermittent omissions. You can also find other references and resources for more information on the website.

3 Provided algorithms

In order to give you a head start with some readily available code, the implementations for two algorithms are provided. Both have been implemented in a different language in order to give you a choice that suits your preference. The two algorithms are:

- AMaLGaM [3] (C code)
- Evolution Strategies [4] (Python code)

4 Recommended Project Timeline

This section describes an outline of the expected progress over the course of the assignment. Nevertheless, these guidelines are not enforced and only your final results are graded.

Week 1. Your assignment is to improve the baseline performance of an EA and report the findings. To measure the baseline, you are to benchmark the EA as provided and produce graphs or tables with the results. You should also make yourself familiar with the code and think of ideas for the improvement that you are going to make (see week 2).

Week 2. In the second week you should start with implementing the proposed changes in order to improve the performance of the EA on the CiaS problem. To give you an idea of what you should work on, you should change or work on at least the following three points for the EA:

1. Change the constraint handling technique by implementing either boundary repair [1] or the more advanced constraint domination [5].
2. Create a problem specific initialization scheme
3. Optimize the hyperparameters (population size, etc)

A guideline for the second week is to have an implementation for the points mentioned above, and have some preliminary results on whether they are working as expected.

Week 3. At the end of the third week you are required to hand in the final deliverables (code and presentation slides). Thus, you should finish the benchmarks for the algorithm and complete the presentation. The choice of the proposed changes must be motivated and the outcome of the comparison clearly conveyed. Make sure that the comparison is done in a fair and reproducible manner and that statistical significance testing has been performed to determine if the changes resulted in a significant improvement.

Related work (also to get you started)

[1] Peter A. Bosman and Marcus Gallagher. The importance of implementation details and parameter settings in black-box optimization: A case study on gaussian estimation-of-distribution algorithms and circles-in-a-square packing problems. *Soft Comput.*, 22(4):1209–1223, feb 2018.

[2] Eckard Specht. Packomania website, Feb 2022. `www.packomania.com` (accessed: 30.03.2022).

[3] Peter A. N. Bosman, Jörn Grahl, and Dirk Thierens. Benchmarking parameter-free amalgam on functions with and without noise. *Evolutionary Computation*, 21:445–469, 9 2013.

[4] Nikolaus Hansen, Dirk V. Arnold, and Anne Auger. Evolution strategies. In Janusz Kacprzyk and Witold Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 871–898. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[5] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2):311–338, 2000.

Evolving a lunar lander with genetic programming

CS4205: Evolutionary Algorithms - Assignment 2
Supervisor: Thalea Schlender (t.schlender@lumc.nl)
Topic: Genetic Programming

Figure 1: A successful lunar landing

1 Problem Definition

Reinforcement learning (RL) is the task of training an agent to perform a certain task in an environment. The RL problem considered in this assignment is the Lunar Lander environment of Gymnasium. RL problems can be approached using EAs.

2 Goal of the Assignment

The goal of the assignment is to gain a better understanding of how Genetic Programming works, why it works that way, and how it can be used in another application than the one seen in the lecture (symbolic regression), namely, RL. Examples of aspects to consider for improving GP are: using a different representation, crafting new atomic functions, designing new fitness functions, optimising coefficients etc.

3 Materials

Documentation of the Lunar Lander environment is available at the Gymnasium website (discrete version). You are provided with a working GP library written 1 in Python at `https://github.com/matigekunstintelligentie/genepromulti`. The action space consists of 4 discrete actions, 0: do nothing 1: fire left orientation engine, 2: fire main engine, 3: fire right orientation engine. A multi-tree is implemented in the library to produce 4 action values, but you are welcome to use other solutions that can generate multiple outputs. The multi-tree is differentiable and programmed using PyTorch. An RL loop is set up for you, but it is up to you where and how to use this loop. You are further provided with a `requirements.txt` file that lists the libraries needed to implement your solution, as well as a Python notebook `solution.ipynb` that contains some utility functions. **All** the code you produce to tackle this assignment should be written in this notebook (unless a strong motivation for doing otherwise is brought forward and the supervisor agrees).

4 Suggested Weekly Timeline

At the end of Week 1 you have

1. Got the baseline agent running
2. Come up with potential ideas on what you want to do. Already start thinking about why these ideas are interesting and how you can implement and test it. Note that as the course is on evolutionary algorithms, your ideas should include modifying the existing algorithm in some way.

At the end of Week 2 you have

1. Implemented at least one new feature.
2. Set up experiments to test the performance of the baseline and the improved agent(s)
3. Have the results of those experiments for the baseline agent.
4. Started working on the presentation.

At the end of Week 3 you have

1. All experimental results.
2. Evaluated your implemented improvements - Do they work? Why? What evidence do you have?
3. Finalized the presentation.

5 Deliverables

The completed assignment should be uploaded to Brightspace (see the general guidelines). Important: For this assignment, the following applies to the final deliverable:

- As mentioned before, all code should be contained in the notebook `solution.ipynb`.
- The notebook should also contain two (or more) GIFs: one showing the behaviour of the baseline agent (i.e., the agent obtained by the baseline GP) and one (or more) showing the behaviour of the improved agent(s) (depending on how many improvements you realise). See, e.g., https://stackoverflow.%com/questions/51527868/how-do-i-embed-a-gif-in-jupyter-notebook for how to include GIFs in a notebook.

6 Related Work

1. Poli R., Langdon W.B., McPhee N.F., 2009: A field guide to genetic programming — Part II: Advanced genetic programming. (Available as reading material for lecture 3.)
2. urlhttps://en.wikipedia.org/wiki/Reinforcement learning
3. Hein, D., Udluft, S. and Runkler, T.A., 2018. Interpretable policies for reinforcement learning by genetic programming. Engineering Applications of Artificial Intelligence, 76, pp.158-169.
4. Zhang, H., Zhou, A. and Lin, X., 2020. Interpretable policy derivation for reinforcement learning based on evolutionary feature synthesis. Complex & Intelligent Systems, 6(3), pp.741-753.