Problem Statement and Goals
Agolearn

Team 1, Agonaught(s)
Yiding Li

Table 1: Revision History

Date	Developer(s)	Change
2024-01-15	Yiding Li	(0.0) First Draft
2024-01-16	Yiding Li	(0.1) Add and resolve TODO items 1. Clarify inputs 2. Clarify goals 3. Refine stakeholders 4. Fix an erroneous figure and its caption 5. Resolve TODO item on adding references
2024-01-17	(0.2) Yiding Li	Return **Motivation** to **Problem**: describe "what it does", not "what it is".

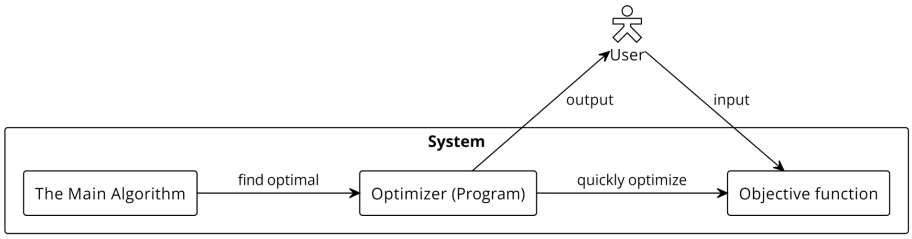
1 Problem Statement

1.1 Problem

This project seeks to implement an evolutionary optimizer for arbitrary real-valued functions and higher-ordered functions.

While evolutionary algorithms lack the efficiency of traditional optimization algorithms, they lend well to solving poorly understood, dynamic, or black-box problems. As such, usage of evolutionary algorithms is key to imeplemting this project. Generic programming algorithms, in particular, make it possible to build functions that optimize higher-order functions.

Figure 1: Input and output of a genetic programming algorithm



1.2 Inputs and Outputs

1.2.1 Optimizing real-valued functions

Input:

1. A real-valued function.
2. Parameters that control the evolutionary process. Examples are: (a) choices of evolutionary operators, (b) number of episodes, and (c) the length of each episode or truncation conditions.

Output: A real vector that optimizes the given real-valued function.

1.2.2 Optimizing higher-ordered functions

Input:

1. A higher-order function.
2. Parameters that control how solutions are generated. Examples are: (a) node functions, (b) tree depth, and (c) node count.
3. Parameters that control the evolutionary process. Examples are: (a) choices of evolutionary operators, (b) number of episodes, and (c) the length of each episode or truncation conditions.

Output: A function that optimizes the given higher-order function.

1.3 Stakeholders

This project can be used by anyone who seeks to optimize an objective function. It should be especially suited for derivative-free functions.

2 Goals

- Optimize against real-valued functions
- Optimize against higher-order functions

3 Stretch Goals

- Implement multi-processing to speed up computation (in reference to frameworks such as Deap)

4 Appendix

4.1 Evolutionary Algorithms

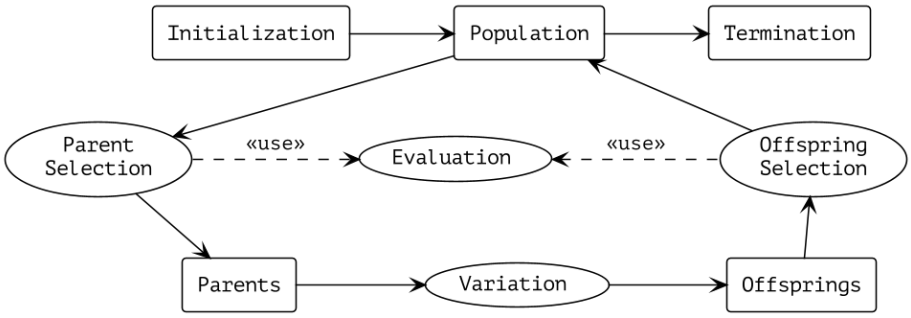
Evolutionary algorithms (**EA**) are optimization algorithms that draw on the evolutionary process. An EA begins with an initial population, then iteratively improves the population through generations by applying various evolutionary operators.

4.2 Evolutionary Operators

Evolutionary operators divide into **parent selectors**, **variators**, and **survivor selectors**. These operators emulate events in an evolutionary process:

- **Parent selectors** select from the population to form the parent pool.
- **Variators** act on the parent pool to produce a pool of offspring. An offspring may inherit traits from parents (by crossover) or posses novel traits (by mutation).
- **Survivor selectors** select from the offspring pool to form the population for the next generation.

Figure 2: Stages and operators in an evolutionary episode



4.3 Genetic Programming

Genetic programming (**GA**) algorithms are evolutionary algorithms that evolve with programs. That is, such algorithms evolve *functions* against an

higher-order objective function that takes functions as input. Genetic algorithms can evolve agents that behave well in a particular environment (e.g. a bipedal walker) or construct mathematical models (e.g. symbolic regression)