Problem Statement and Goals
Agolearn

Team 1, Agonaught(s)
Yiding Li

Table 1: Revision History

Date	Developer(s)	Change
2024-01-15	Yiding Li	First Draft
2024-01-16	Yiding Li	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

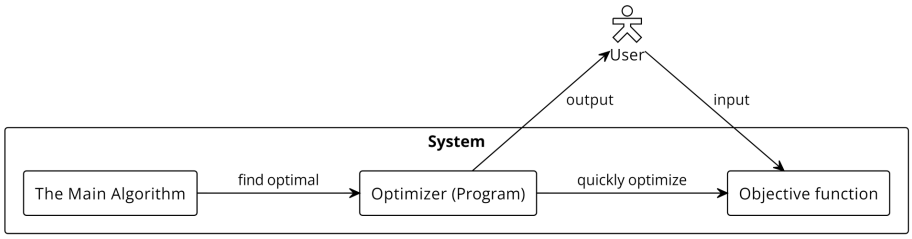
1 Problem Statement

1.1 Motivation

Evolutionary algorithms lend well to solving novel problems that are difficult to model or solve by traditional means. Genetic programming algorithms, in particular, are able to evolve programs that perform well against arbitrary objective functions.

While evolutionary methods lack the efficiency of traditonal optimization methods (such as simplex algorithms and gradient desgent algorithms), they work in more complex environments (such as moving constraints, moving optimization functions, partially observable environments, and objective functions that do not have well-defined derivatives). This project seeks to develop an optimizer that takes advantage of these features.

Figure 1: Input and output of a genetic programming algorithm



1.2 Inputs and Outputs

Input:

1.2.1 Optimizing real-valued functions

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

1.2.2 Optimizing higher-ordered functions

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 evolutoinary operators, (b) number of episodes, and (c) the length of each episode or truncation conditions

Output: A value or genetic program that optimizes the given objective function

1.3 Stakeholders

This project is designed for anyone who seeks to optimize against an objective function. In particular, it is able to optimize against black-box or stochastic functions where traditonal optimization methods fail.

2 Goals

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

3 Stretch Goals

- Optimize against an objective function of functions (genetic programs)
- Implement multi-processing to speed up computation (in reference to frameworks such as Deap)

4 Appendix

4.1 Evolutionary Algorithms

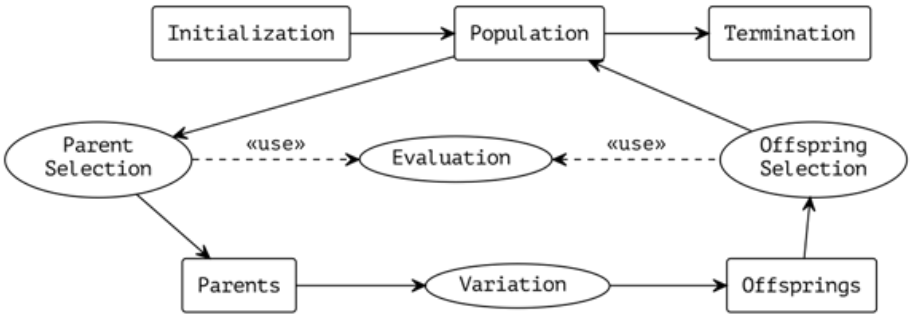
Evolutionary algorithms (**EA**) are optimization algorithms that draw on the evolutionary process. An EA begins with an initial population, then iteratively improves the population through generations by applying various evolutionary operators.

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 offsprings. An offspring may inherit traits from parents (by crossover) or posess 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 Algorithms

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