Report on Mosses- Evolutionary algorithm

By : Bisrat Berhanu
Date: March 19 2025

What is Evolutionary algorithm

A kind of computer program that is made to solve complicated issues by simulating how **nature evolves**. Natural selection, a biological process in which only the strongest survive and evolve throughout time, serves as an inspiration for them.

In a single sentence Evolutionary algorithm can be explained as "survival of the fittest".

These algorithms start with a population—a collection of potential solutions. Every solution is assessed according to how well it solves the given problem , which is determined by its "fitness." The best solutions are then chosen to produce the next generation, frequently by **crossover**—the mixing of elements from different solutions—and mutation—the introduction of minor, arbitrary changes. Over several generations, this process is repeated, progressively refining the solutions until a good solution is found.

What is MOSES?

Quick background

- Meta-optimizing semantic evolutionary search (MOSES) - abbreviation
- Original paper released by Predrag Janicic, Revised by Linas Vepstas 3 November 2008, revised 30 January 2012

Current status in Icog labs

- **Part of hyperon Team - hyperon is a big AGI framework**
  Team members :
    - Yabsira Derese (Team Lead)
    - Liya
    - Eyobed
    - Kirubel

How Does MOSES Work?

- It starts with a group of programs (called a population) and picks the best ones.
- MOSES improves these programs by adding "knobs" (adjustable parts) and tests small changes, like mutations in evolution.
- It uses tools like hill-climbing or **Bayesian optimization** to find the best knob settings, making programs stronger over time.

Key Features

- **Demes:** Each group of programs comes from one main program with knobs to tune.
- **Fitness Check:** MOSES tests how good each program is and keeps the best ones.
- **Clean-Up:** It simplifies programs to avoid messy, hard to understand "spaghetti code" using a neat format called "**elegant normal form**."

Why is MOSES Special?

- It's faster and more accurate because it finds smart connections between program parts and skips useless changes.
- MOSES learns from examples **(supervised learning)** using scoring rules or training data to guide it.

What Does MOSES Create?

- It outputs a "Combo" program—a simple, Lisp-like code(this is now being implemented in MeTTa) that's easy to understand and use.
- These programs can handle many tasks, like logic, actions, or functions, depending on the problem.
- MOSES works with OpenCog to pick only the most useful variables for a problem, making it even faster and sharper.
- Written in C++, it's a library with examples to help people use it easily.

What Problems Are the MOSES Team Chasing?

- **Complex Boolean Expressions:**
  - The MOSES team at OpenCog is working to solve tricky and complex Boolean logic problems (true/false statements).
  - They are finding clear, efficient solutions for these complex expressions using MOSES' program evolution.

How Is the MOSES Team Approaching the Problem?

- **Focus on Boolean Expressions:**
  - The team is targeting complex Boolean logic problems (true/false statements) for now.
  - They use MOSES, an evolutionary algorithm (EA), to evolve programs that find clear solutions.
- **Their Unique Strategy:**
  - MOSES stands out from other EA methods with special techniques to crack these problems efficiently.

How Do They Think MOSES Solves the Problem?

- **Holman Boolean Reduction:**
  - They use a method called "Holman's elegant normal form" to simplify Boolean expressions.
  - This keeps programs short, readable, and free of messy, redundant parts.
- **Probability Modeling:**
  - They build models to predict which program changes will work best, guiding the evolution process.
  - This makes finding solutions smarter and faster than random guessing.
- **Complex Selection Algorithms:**
  - MOSES combines crossover (mixing program parts), mutation (small random tweaks), and Estimation of Distribution Algorithms (EDA) like Bayesian optimization.
  - These methods help pick the fittest programs and explore useful changes effectively.
- **Extra Optimization Tools:**
  - They also use advanced tricks like SGE (possibly Stochastic Gradient Evolution) and PGE (perhaps Probabilistic Gradient Evolution).
  - These boost MOSES' power to fine-tune solutions for tough Boolean problems.

Current State/Stage of MOSES

- **Porting to MeTTa:**
  - They are moving the C++ version of MOSES to MeTTa, a flexible, multiparadigm language.
  - This aims to make MOSES more adaptable and easier to use.
- **Exploring Optimizations:**
  - The team is researching new optimization methods like SGE (Stochastic Gramatical Evolution) and PGE (Probabilistic Grammatical Evolution).
  - They're testing these to improve MOSES' problem-solving power.

Challenges Faced/Facing by the MOSES Team

- **MeTTa's Young Development:**
  - MeTTa is a new language, only a few years old, so it lacks rich tools and libraries.
  - This makes development harder compared to mature languages like C++.
- **MeTTa's Speed:**
  - MeTTa runs slower than C++, which could slow down MOSES' performance.
  - This creates a challenge for efficient program execution.
- **Porting C++ to MeTTa:**
  - C++ is object-oriented, while MeTTa is multi paradigm, making code conversion hard..
- **Integration with Hyperon Components:**
  - Figuring out how MOSES in MeTTa connects to other Hyperon parts, like Attention Allocation and PLN (Probabilistic Logic Networks), is complex.
  - This requires solving compatibility and interaction issues and is the long term Goal of Moses .

Reference

1. https://github.com/opencog/moses
2. https://wiki.opencog.org/w/Meta-Optimizing_Semantic_Evolutionary_Search