# Saturday 15/04/25 Task Submission **Name:** Nardos Amakele

# Technical Report on MOSES and Its Integration with MeTTa in OpenCog

## 1. Introduction

### What is MOSES?

MOSES (Meta-Optimizing Semantic Evolutionary Search) is a high-level evolutionary algorithm used for program optimization and synthesis. It works on representation-building and probabilistic modeling. It has been developed to automatically generate and optimize programs so it is particularly helpful in symbolic reasoning and AI-based decision-making. MOSES diverges from conventional algorithms like genetic programming that has spaghetti code with large and complicated structures,, since Moses tries to produce correct and human-readable programs in a lisp-like mini-language.This algorithm has been applied in fields like sentiment evaluation and agent control.

MOSES is part of the OpenCog AI framework and is utilized for applications with:

* Successful program evolution
* Automatic feature selection
* Probabilistic modeling for optimization

### MOSES is an important component of OpenCog, an open-source AGI (Artificial General Intelligence) software framework. It is in the process of being integrated into MeTTa, a meta-language in OpenCog, Hyperon, for enhancing symbolic reasoning and AI program synthesis.

### Key Features of MOSES

* Structured Expression Trees: Used to represent programs as a tree structure.
* Optimization Algorithms: Employs hill-climbing, simulated annealing and more.
* Normalization & Reduction: Removes redundant or unnecessary code (avoiding "spaghetti code").
* Probabilistic Modeling: Rapidly searches for program variations.

## 2. Problem MOSES is Addressing

Traditional program evolution methods like Genetic Programming (GP) and Evolutionary Programming (EP) are afflicted with:  
 · **Complexity and Redundancy** – The majority of evolved programs become cumbersome and difficult to comprehend, often referred to as "spaghetti code."

· **Computational Expense** – Large search spaces result in inefficiency.

· **Lack of Transparency** – Programs produced are generally highly nested, making them difficult to examine and adjust.

MOSES avoids these by:  
 · Reducing complexity via Holman's Elegant Normal Form, which involves removing useless terms and maintaining structured simplicity.

· Efficiently searching the program space using probabilistic modeling instead of random mutations.

· Generating human-readable output that can be directly understood and applied.

## 3. How MOSES Works: The Algorithm

MOSES employs two nested optimization loops to refine programs effectively:

### Outer Loop (Metapopulation Management)

The outer loop possesses a metapopulation, a pool of candidate programs, and performs the following:

· **Exemplar Selection**: The best-performing program (exemplar) is chosen from the meta-population. This can be a blank program or a simple heuristic rule to start with.

· **Representation Building**: The selected program is extended with adjustable parameters ("knobs"). These knobs provide flexibility in program behavior, allowing fine-tuning.

· **Merging Optimized Programs**: Following optimization, better programs are recombined into the meta-population. Worse candidates are discarded, ensuring that only good solutions remain.

### Inner Loop (Local Optimization)

Once a program is selected, MOSES explores its local neighborhood by adjusting its parameters. This process involves:

**Generating New Variants**  
MOSES modifies knob values to create multiple program instances. Each instance represents a possible improvement.

**Scoring Each Variant**  
A fitness function evaluates how well each instance performs.  
Example: If used in a medical diagnosis system, fitness could be measured by prediction accuracy on patient data.

**Optimization Using Search Algorithms**  
MOSES refines the best candidates using:

* + **Hill-Climbing** – Incremental improvements.
  + **Simulated Annealing** – Accepting temporary performance drops to escape local optima.

**Normalization & Reduction**  
The best-performing programs are simplified by removing redundant elements. This makes them more efficient and interpretable.

**Merging Back to the Outer Loop**  
The best instances become exemplars and are returned to the meta-population for the next iteration.

This cyclical process ensures that MOSES continuously refines its programs, leading to increasingly optimized solutions.

## 4. Current State of MOSES

As per the information provided in the OpenCog website, MOSES is currently used in:

* **Computational Biology** (gene expression analysis)
* **Natural Language Processing (NLP)** (sentiment analysis, text categorization)
* **AI Agent Control** (robotics decision-making)

Currently, MOSES is being integrated into MeTTa to enhance structured knowledge representation, enhance symbolic and sub-symbolic AI model compatibility and provide a more flexible programming environment for AI reasoning tasks.

## The work in iCogLabs now includes porting to MeTTa and creating type definitions for the demes, knobs, and other components of MOSES to eventually fully implement the system in MeTTa and incorporate it into the OpenCog system.

## 5. Challenges in Integrating MOSES with MeTTa

### Representation Compatibility

* **MOSES** generates programs in Combo, which is a tree-based representation.
* **MeTTa** employs a different model of execution, requiring MOSES output adaptation.

### Optimization in Symbolic AI Contexts

* **MOSES** must generate programs that can effectively interact with symbolic reasoning systems.
* New optimization techniques may be required to handle complex relationships between AI concepts.

### Data Structure Challenges in MeTTa

* **MOSES's** C++ implementation uses vectors and structured data containers for storing field sets.
* Finding an equivalent effective data structure in MeTTa is a major technical challenge.

## 6. Conclusion

MOSES is a strong AI utility that assists in developing and refining structured programs on its own. Incorporating MOSES into MeTTa will make AI more capable of grasping concepts, creating knowledge, and making independent decisions more efficiently.