VnV Report for AgnoLearn

Yiding Li
March 25, 2024

1 Revision History

Date	Version	Notes
2024-04-15	1.0	Final version
2024-04-25	1.1	Export

Contents

2 Symbols, Abbreviations, and Acroyms

This section records information for easy reference.

Table 1: Abbreviations and Acronyms

Symbol	Description
VnV	Verification and Validation
GP	Genetic Program

3 Introduction

VnV

The **verification and validation** (**VnV**) report discusses test cases outlined in the VnV plan. For each test case, the report discusses the outcome of the test, reasons for the outcome, and notable characteristics of the outcome.

At this time, most tests have not been conducted. This is because the user interface is still being iterated. As most test cases require

Table 2: Overview of tests, outcomes, and rationales

Test Case	Outcome	Rationale
T 1	Pass	–
T 2	Pass	–
T 3	Not implemented	Time constraint
T 4	Partial	Time constraint
T 5	Not implemented	Interface incomplete
T 6	Not implemented	T 5 incomplete
T 7	Fail	Time constraint
Code Coverage Metrics	Not implemented	Time constraint

4 Functional Requirement Evaluation

This section discusses tests for functional requirements.

4.1 T 1

Pass. The algorithm converges with the following representations and operator configurations.

Table 3: Converging operator configurations

Repr.	**Variator**	**Selector**	**Evaluator**
Bit string	Bit Mutator	Simple elitist	OneMax
Bit string	Bit Mutator	Tournament	OneMax
GP	Subtree Crossover	Simple elitist	Gymnasium-CartPole
GP	Subtree Crossover	Simple elitist	Symbolic regression
GP	Subtree Crossover & Node Mutator	Simple elitist	Symbolic regression

4.2 T 2

Pass. The following operators have been tested in more than one configurations. Note that a comprehensive test of every operator, with every configuration and with non-trivial training parameters, will become difficult as the number of representations and operators increases.

Table 4: Components that have been tested in multiple algorithm

Component	Operators
Representation	Bit string GP
Variator	Subtree Crossover, Subtree Crossover & Node Mutator
Selector	Tournament Simple elitist selector
Evaluator	OneMax Symbolic regression

4.3 T 3

Not implemented. This test has not been implemented in the interest of time.

4.4 T 4

Partial. Only some representations have been implemented. These representations have been tested in T 1.

Table 5: Implementation status: representations

Representation	Implementation Status
OneMax	Implemented
Floating point	Not implemented
Genetic programming	Implemented
Linear programming	Not implemented
Tangled program graph	Not implemented

4.5 T 5

Not implemented. The interface is still being iterated – while some tests have been implemented as proof of concept, the exact interface is subject to change.

At this point, any external document that references the interface will likely need to be changed in the future. The developer decides to not provide an user guide in the interest of time.

5 Nonfunctional Requirement Evaluation

This section discusses tests for nonfunctional requirements.

5.1 T 6

Not implemented. This test is only possible after T 5, which is incomplete at this point.

5.2 T 7

Fail. Manual inspection shows that not all public fields and methods have been documented.

6 Comparison to Existing Implementations

The usage of the system is distinct.

7 Unit Testing

Unit testing has not been implemented

8 Changes Due to Testing

A number of modules have been developed for use in testing. These modules are concrete implementations of core modules.

Table 6: Concrete components

Type	**Modules**
Representation (Genome)	BitString, Program
Variators	BitMutator SubtreeCrossoverVariator SubtreeVariator
Selectors	Elitist SimpleSelector TournamentSelector
Evaluators	OneMaxEvaluator GymEvaluator SymbolicRegressionEvaluator

9 Automated Testing

Automated testing has not been implemented.

10 Trace to Requirements

Table 7: Derivation of test cases from requirements

T	FR													NFR	
	1	2	3	4	5	6	7	8	9	10	11	12	13	1	2
1	✓														
2		✓	✓	✓	✓										
3						✓	✓								
4								✓	✓	✓	✓	✓			
5													✓		
6														✓	
7															✓

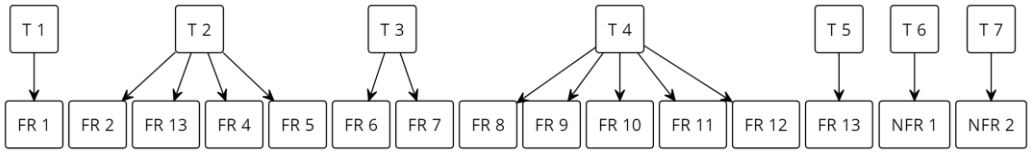

Figure 1: Traces from tests to requirements

11 Trace to Modules

The sheer number of modules, and the fact that high-level tests involve multiple modules, makes these traceability artefacts difficult to read.

The developer is considering decomposing tests into smaller tests, so that each test focuses on one module. This has not been done due to time constraints.

Table 8: Traceability from tests to modules used in tests

Test	**Modules**
T 1	Controller, BitString, BitMutator, Elitist, SimpleSelector, OneMaxEvaluator, TournamentSelector, SubtreeCrossoverVariator, SubtreeVariator, Program, Gy,Evaluator, SymbolicRegressionEvaluator
T 2	Same as modules used in T 1
T 3	–
T 4	Program, OneMax
T 5	–
T 6	
T 7	All modules

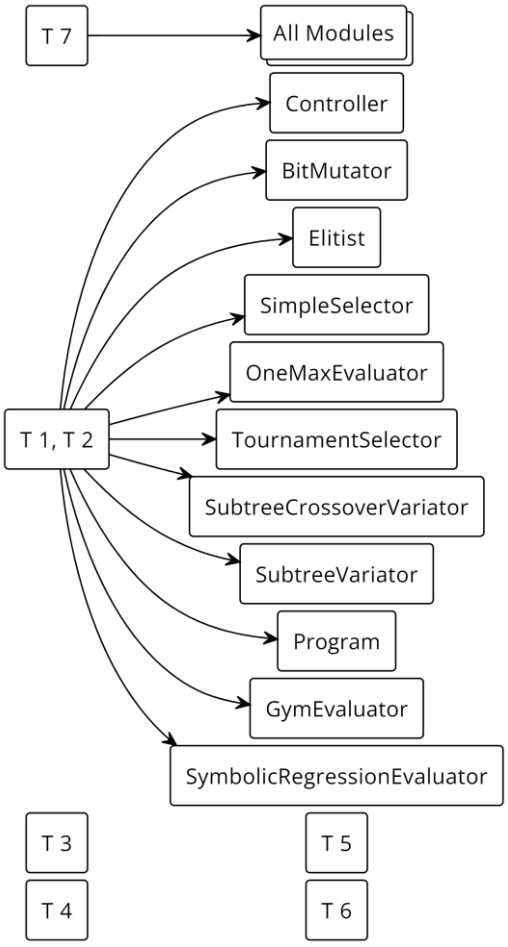

Figure 2: Traces from tests to modules

12 Code Coverage Metrics

Code coverage has not been measured.