2021

Increasing Software Reliability using Mutation Testing and Machine Learning

Michael Allen Stewart

Increasing Software Reliability using Mutation Testing and Machine Learning

by

Michael Allen Stewart

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Science

College of Computing and Engineering
Nova Southeastern University

2021

We hereby certify that this dissertation, submitted by Michael Allen Stewart conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

[illegible] **10/6/21**

Francisco J. Mitropoulos, Ph.D. **Date**
Chairperson of Dissertation Committee

[illegible] **10/6/21**

Michael J. Laszlo, Ph.D. **Date**
Dissertation Committee Member

Sumitra Mukherjee **10/6/21**

Sumitra Mukherjee, Ph.D. **Date**
Dissertation Committee Member

Approved:

[illegible] **10/6/21**

Meline Kevorkian, Ed.D. **Date**
Dean, College of Computing and Engineering

College of Computing and Engineering
Nova Southeastern University

2021

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Increasing Software Reliability using Mutation Testing and Machine Learning

by
Michael Allen Stewart
October 2021

Mutation testing is a type of software testing proposed in the 1970s where program statements are deliberately changed to introduce simple errors so that test cases can be validated to determine if they can detect the errors. The goal of mutation testing was to reduce complex program errors by preventing the related simple errors. Test cases are executed against the mutant code to determine if one fails, detects the error and ensures the program is correct. One major issue with this type of testing was it became intensive computationally to generate and test all possible mutations for complex programs.

This dissertation used machine learning for the selection of mutation operators that reduced the computational cost of testing and improved test suite effectiveness. The goals were to produce mutations that were more resistant to test cases, improve test case evaluation, validate then improve the test suite's effectiveness, realize cost reductions by generating fewer mutations for testing and improving software reliability by detecting more errors. To accomplish these goals, experiments were conducted using sample programs to determine how well the reinforcement learning based algorithm performed with one live mutation, multiple live mutations and no live mutations. The experiments, measured by mutation score, were used to update the algorithm and improved accuracy for predictions. The performance was then evaluated on multiple processor computers.

One key result from this research was the development of a reinforcement algorithm to identify mutation operator combinations that resulted in live mutants. During experimentation, the reinforcement learning algorithm identified the optimal mutation operator selections for various programs and test suite scenarios, as well as determined that by using parallel processing and multiple cores the reinforcement learning process for mutation operator selection was practical. With reinforcement learning the mutation operators utilized were reduced by 50 – 100%.

In conclusion, these improvements created a 'live' mutation testing process that evaluated various mutation operators and generated mutants to perform real-time mutation testing while dynamically prioritizing mutation operator recommendations. This has enhanced the software developer's ability to improve testing processes. The contributions of this paper's research supported the shift-left testing approach, where testing is performed earlier in the software development cycle when error resolution is less costly.

Acknowledgements

To all my professors and the staff at Nova Southeastern University, I would like to thank you for the concepts you have taught me, including how to conduct research and guiding me through the doctoral program. Particularly helpful to me during my time at NSU were Dr. Francisco Mitropoulos, Dr. Sumitra Mukherjee, and Dr. Michael Laszlo, who I would like to express my deepest appreciation in helping to focus on a dissertation topic and make contributions to the fields of machine learning and software testing.

Several friends and colleagues have been supportive of my efforts to achieve this Ph.D. In particular, I would like to recognize Ryan Yocum for the many discussions that have furthered the application of this research. My employer Assurant Inc. has provided me the opportunity to work full-time while pursuing multiple degrees.

Finally, I would like to thank my family and generations of relatives that have given me the faith and ability to succeed. Mom, you have taught me how to be creative and enjoy life. Dad, you taught me to always work hard and help others. To my wife Nadia and children Julissa and Victoria, you have provided me love, support and the inspiration to finish my academic journey. May this work be an inspiration to others as well.

Table of Contents

List of Tables

List of Figures

Chapter 1 - Introduction

Mutation testing is a type of software testing proposed by (Lipton, 1971) where program statements are deliberately changed to introduce simple errors so that test cases can be validated to determine if they can detect the errors. The goal of mutation testing is to reduce complex program errors by preventing the related simple errors. For example, given a program that states if (a>=b) then c=1 else c=0 can be mutated by an operation replacing >= with < producing if (a<b) then c=1 else c=0. When using test data of a=1, b=0 the result c should be 1 but the mutant produces c=0. The test cases are executed against the mutant code to determine if one fails, detects the mutant and helps ensure the program is correct. A mutation score is calculated as the percent of mutants caught. One major issue with this type of testing from (Jia and Harman, 2011) is that it becomes computationally intensive to test all possible mutations for complex programs.

This dissertation will present a practical approach for the application of parallel machine learning within the context for mutation testing, including the selection of mutation operators to reduce the computational cost of testing and improve test suite effectiveness. With this, the need to increase the usage of mutation testing for complex programs can be fulfilled. The proposal is to use reinforcement learning for mutation testing that improves mutation scores achieved previously (Strug & Strug, 2018, June) by predicting which mutation operators best identify deficient test coverage.

These improvements will assist with the creation of a ‘live’ mutation testing process within the .NET development environment that dynamically evaluates various mutation operators, generates mutants and prioritizes test cases to perform real-time mutation testing as code is modified. This will enhance a software developer’s ability to improve testing processes and extend the work by (Derezińska & Trzpil, 2015). The contribution of this paper's research will support the shift-left testing approach, where testing is performed earlier in the software development cycle when error resolution is less costly.

Problem Statement and Goal

The problem is that continuous software testing can be a daunting process, even when testing is engrained into the development process. Although attempts have been made to address this problem (Demeyer et al., 2018), the approach to limit testing without test case validation can discard pertinent tests. Testing can also become dispensable to meet development deadlines. As discussed by (Martin et al., 2007) companies sometimes deploy limited testing resources to find software defects. When testing becomes incomplete it inevitably leads to faulty software. These defects are becoming more of an issue as the reliance increases on software for essential services such as financial, transportation, and healthcare.

Many challenges lead to a lack of testing and faulty software. First, software testing requires proper communication and documentation to define what is needed. The potential for misinterpretation exists, which can lead to missing or invalid test scenarios. Even valid test scenarios can become a challenge to execute and evaluate, since applying all test scenarios can be labor-intensive and error-prone. These challenges result in less than sufficient testing and increase the time developers spend on debugging. According to recent reviews by (Campos & de Almeida Maia, 2017), the annual cost of debugging software has reached $312 billion globally.

To address this concern, testing must become more agile when integrated within the software development process. With the adoption of Continuous Integration and

Continuous Delivery, the goal as described by (Shahin et al., 2017) is to reduce the time to deliver software changes but lack of proper testing the goal cannot be fully realized. Continuous Testing, which as described by (Demeyer et al., 2018) improves testing feedback and must be incorporated with software delivery. To complement Continuous Delivery with Continuous Testing, the Test Suite which is composed of Test Cases must cover the software (i.e. test completeness) and identify defects that exist (i.e. test quality). Another factor that needs to be addressed for the software testing process is the amount of time and effort it can take to develop and execute a comprehensive test suite.

The goal is to assist software developers with an approach for comprehensive testing and improving testing effectiveness of their software implementation. It will evaluate the factors that impact software quality then use parallel Reinforcement Learning (RL) for mutation operator selection to identify deficient testing more effectively than a classification-based approach (Strug & Strug, 2018, June). This dissertation proposes a quantitative approach by measuring the faults detected by test suites built with RL-assisted operator selection as compared to those developed without. Through the implementation of these integrated mutation testing approaches, the expectation is an increase in the percentage of defects detected (Qu et al., 2007).

Relevance and Significance

The research proposed in this paper will provide benefits to current software development trends, by improving upon recent work by (Derezińska & Trzpil, 2015) that helped facilitate mutation testing. This dissertation will address this through the use of machine learning for mutation testing and test case selection. The general goal with machine learning as states by (Lu et al., 1996) is to obtain knowledge from patterns within data, using various approaches to accomplish this goal. Mutation testing (DeMillo et al., 1979) is a process that replicates program faults to validate the program test suite. The mutation operators are functions that replicate common programming errors, such as using an incorrect operator. During mutation testing the mutants are either caught by a test case and considered killed or not caught and are considered live. The mutation score (Namin et al., 2008) is the number of mutants killed divided by the total number of mutants and indicates the test suite's effectiveness. Test case selection using machine learning was presented by (Ghiduk et al., 2018) to improve the test case prioritization process. Machine Learning has already started to have an impact on software testing techniques in many ways, as discussed by (Briand, 2008). The software testing process consumes and generates an enormous amount of data. If the evaluation of this data is not performed in an automated or efficient manner, such as parallel machine learning, the test results may not be accurate or complete.

To establish the importance of mutation testing for determining test effectiveness, (Chekam et al., 2017) performed a comparison with other widely adopted test effectiveness metrics, including statement coverage and branch coverage that avoids the unreliable clean program assumption. Statement coverage is a minimal requirement that measures the percentage of program statements that are exercised by the tests but since this measure does not consider the program state and various conditions that can cause the statements to execute differently. A stronger requirement called branch coverage is also utilized. With branch coverage, it measures the percentage of program control flow that is exercised by the tests. However, with both approaches, the measurement assumes that the program is correct, but if the program contains defects these measurements may be inadequate. By introducing program defects the mutation testing approach exercises the tests more completely, thus providing a better measurement of the test effectiveness.

The most significant aspect of this dissertation is the introduction of machine learning for test case selection and mutation testing during the early stages of the development process, as opposed to later after the development process has been completed. This supports the ability to develop software in an agile manner, using the Test-Driven Development (TDD) process proposed by (Beck, 2003) and the Continuous Integration (CI) process proposed by (Booch, 1994). With TDD, software requirements are incrementally encoded as tests that developers must satisfy by coding application logic. The TDD approach was incorporated with mutation testing by (Derezińska & Trzpil, 2015) to provide an interactive process for more agile mutation testing. CI is a development practice where software developers frequently integrate code changes to a shared source repository. Test case selection using reinforcement learning was utilized by Netflix (Kirdey et al., 2019) to develop a system called Lerner that integrates with their CI framework for test execution scheduling. Using TDD and CI helps to reduce program defects by establishing and executing a test suite that ensures program logic is working as expected.

Barriers and Issues

Much research has been conducted related to the issues with software testing (Whittaker, 2000) which includes selecting, running and evaluating test scenarios. Some additional issues are selecting the variable data to be used, execution paths to cover, which test cases to automate and how to evaluate the test case results. For example, if a method is supposed to find all occurrences of some string within an arbitrary text, how can we determine that each instance will always be detected? Although there are many approaches to address some of these issues, such as using category-partition for generating test cases (Ostrand and Balcer, 1988) and using data flow and control flow for evaluating test cases (Hutchins et al., 1994), the issues are not completely resolved since software is still released with defects. To address the barrier and limitation with the variable data, the test scenario evaluation needs to explore the possible combinations. If the algorithmic approach is static, such partitioning there will be inherent limitation based on the data provided. But if the algorithm is able to explore by taking various actions and receive rewards for success, using the proposed reinforcement learning approach will result in a more dynamic approach.

The category-partition method (CPM) for creating test suites uses a generator to produce test specifications from functional specifications. The advantages of this method are that the tester can easily modify the test specification when necessary and can control the complexity and number of the tests by annotating the test specification with constraints. One major barrier with the implementation of CPM is the size of the test suite generated, which can be huge for complex programs. Given a method having five parameter variables and two global variables with a minimum of two possible values per variable the product of all choices which would result in $2^7 = 128$ test cases. With non-trivial programs, the number of variables, range of possible values and number of methods is much higher, so the potential number of tests will be much higher as well.

With control flow, the test cases are selected with the goal to ensure that every source statement is executed at least once. With data flow, the goal is to evaluate test cases to ensure that they exercise the code such that execution proceeds from the definition of a memory location to the use of that memory location for each DEF-USE pair. The limitations with both approaches are that it is difficult to understand complex code logic, which is necessary to achieve various coverage levels, then distinguish the feasible vs. non-feasible paths and the process can be very time consuming for non-trivial programs. The approach proposed in this research to utilize parallel processing will help reduce the issue of time consumption by partitioning the problem, then allowing each component to evaluate a subset of test cases simultaneously.

Lastly, there are barriers to measuring the testing progress that needs to be overcome to realize an integrated testing approach. For the approach to be effective, the measure should give an updated indication of the testing progress. One question posed by (Whittaker, 2000) is if large numbers of defects are found is this good or bad? It could be an indication of comprehensive testing or there may still be many undetected defects. With the proposed approach of using mutation testing, the test suite effectiveness becomes measurable using the mutation score. The mutations are defects and will be generated with the intention of detection. If not detected, the test suite can be enhanced to ensure testing is comprehensive.

Summary

This chapter introduced mutation testing, mutation operators and the importance of software testing. The goal of the proposed research is to develop an approach to assist software developers with improving testing effectiveness and the correctness of their implementation based on given requirements. To complete this goal, the algorithm will utilize parallelized reinforcement learning for mutation operator selection and should result in a more efficient testing solution.

Chapter 2 - Review of the Literature

The usage of machine learning, software testing and parallel processing are key elements to achieve this dissertation's goal of a more effective testing process. This goal will be implemented by mutation testing and reinforcement learning. By using the mutation score the testing effectiveness will be able to be measured. The following sections review the relevant literature:

- Machine Learning
- Software Testing
- Parallel Processing

Machine Learning

The process of engineering test suites can be a formidable effort. Complex applications can require many test cases within the test suite. These tests must consider the inputs and outputs of the code they are testing. By using machine learning (Briand et al., 2008) developed a process to learn relationships between the inputs and outputs as the test suites are executed. With this information, the testers can understand the capabilities of the test suite. Their process uses the C4.5 decision tree algorithm (Quinlan, 1993) within the WEKA (Waikato Environment for Knowledge Analysis) machine learning library (Frank et al., 2016) since it produces machine learning models that are easier to interpret. The paper reported promising results by eliminating redundant test cases and a significant reduction in the test suite size but also found a reduction in the number of faults detected, leaving room for the test suite improvements that this paper's research hopes to obtain using machine learning to assist with identifying missing test cases.

Another use of machine learning for mutation testing was presented by (Guillaume, 2015) and (Kurtz Jr, 2018). Their basic approaches were to reduce the number of mutants generated by randomly selecting a percentage of mutants or by reducing all mutants for a given operator. Those approaches were compared with a machine learning approach for mutation operator selection. The papers conclude that a machine learning approach is significantly superior but anticipate future improvements by more advanced machine learning approaches, such as multi-layer perceptron. This dissertation proposes to explore these improvements among others.

Recently progress has been made using machine learning in the context of mutation testing. With their earlier work (Strug & Strug, 2012) presented an approach that represented mutants using a graph kernel to compare mutant similarities and then used k-Nearest Neighbor (k-NN) machine learning algorithm to predict if a test would detect a mutant, reducing the number of mutants executed. Additional research by (Strug & Strug, 2017) proposed an updated kernel called a hierarchical control flow graph (HCFG), which is a combination of control flow diagram and hierarchical graphs. This limited mutant execution in a more dynamic way by utilizing the structure of the program for which the mutants were generated. In their next research, (Strug & Strug, 2018, June) proposed to simplify the mutant evaluation process by using bytecode comparison instead of source code control flow, which was more complicated. The latest research by (Strug & Strug, 2018, September) takes an even more extreme approach by predicting the mutation testing results (killed vs. live) based on machine learning models, without having to execute any mutation testing after the initial training process. A similar approach was proposed by (Zhang et al., 2018) except they used a Random Forrest machine learning algorithm (Liaw & Wiener, 2002), which is a generalization of tree-based classification that uses multiple decision trees to correct overfitting, to create their predictive mutation testing.

While those papers reduced mutation execution using machine learning, this dissertation proposes a novel approach using machine learning to limit mutation operators and generate mutants during program development, thus reducing the number of mutations generated during an agile development process. The proposed research of applying test case selection and mutation testing in real-time will help keep the test suite more updated and predictable by measuring mutation score of the test suite over time. To utilize a more effective machine learning algorithm, instead of using a supervised learning approach, this paper proposes using a Reinforcement Learning (RL) approach as presented by (Sutton & Barto, 1998). As shown in Figure 1, the agent learns to choose actions in an environment by performing actions then observing the subsequent states and rewards. It continues until the reward is consistent and acceptable.

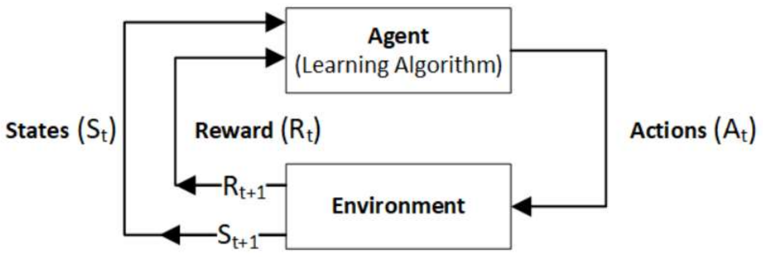


Figure 1. The general Reinforcement Learning approach.

This is another key difference when compared with the supervised learning approach presented by (Strug & Strug, 2018, June) and provides the advantage of agility.

This approach is model-free, which means it has no initial concept of the environment's dynamics and utilizes online learning, where the agent is constantly learning while running. This is appropriate for test case selection since there is no strict model to identify faults and according to (Campos & de Almeida Maia, 2017), the existence of faults is prevalent within software systems. For test selection, given previous test results in each state the agent performs an action that prioritizes the test cases based on the reward of failed tests from the environment during test cycle execution. This process was proposed by (Spieker et al., 2018) and is shown in Figure 2.

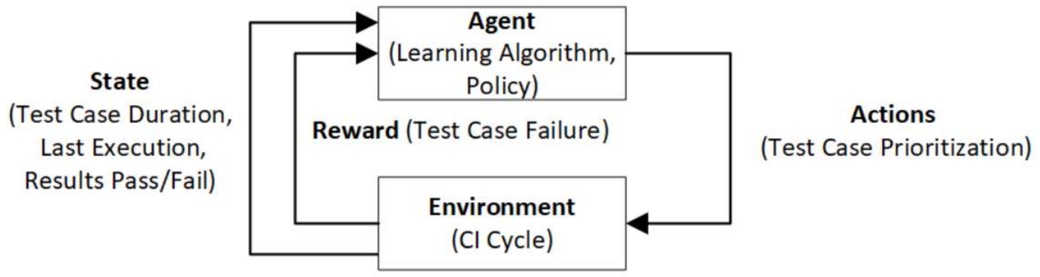

Agent
(Learning Algorithm, Policy)
State
(Test Case Duration, Last Execution, Results Pass/Fail)
Reward (Test Case Failure)
Actions
(Test Case Prioritization)
Environment
(CI Cycle)

Figure 2. Reinforcement Learning for Test Case Selection.

One of the challenges with machine learning is determining the data elements, called features, to use during training that will produce accurate predictions during testing. The paper by (Jalbert & Bradbury, 2012) utilized the Support Vector Machine (SVM) machine learning algorithm to categorize mutation scores (i.e. low, medium, high) which reduces the mutation score prediction to a three-group classification problem. The machine learning features include various class-level metrics (e.g. # of methods, # of attributes, inheritance depth) and method-level metrics (e.g. # lines of code, # of parameters, nested depth, cyclomatic complexity) as well as accumulated test case metrics (e.g. average # test method lines of code, average # test parameters, average test cyclomatic complexity). To collect these metrics required using several Java tools, which included an Eclipse IDE plugin for code metrics, EMMA for test metrics and Javalanche for method-level mutations. This technique for predicting mutation score (# mutants killed / total # mutants) achieved an accuracy of >50% using source code and test suite metrics which outperformed the random accuracy of 33.33%.

In the work by (Zhang, et al., 2018) additional metrics were evaluated to investigate the contribution of the 14 individual features, including propagation features (method lines of code, method complexity), infection features (mutation operator, mutated statement type), and execution features (number executed, number tests covering mutated statement). The features were used by various classification algorithms, including Random Forrest, Naïve Bayes, SVM and C4.5 Decision Tree. It was determined that the coverage features, including the number of times that the mutation was executed by tests and the number of tests that covered the mutation, were the most important features.

Various source code and test metrics are evaluated as features by (Spieker et al., 2018) using Reinforcement Learning (RL) to prioritize test case selection. In Figure 3, the reward function utilized various features, including a count of test failures, each test failure and test failure time. The states (i.e. test case metrics) are provided as inputs X_i to the network. Feedforward estimates the policy π based on current weights and activation functions. The actions (i.e. test case priority) are output O_a from the network. A random factor is used for exploration and experience for replay training. During backpropagation, weights W_i are updated using error estimate or loss from loss function O_a - O_e using gradient descent. Neural networks are shown effective for data mining (Lu et al., 1996).

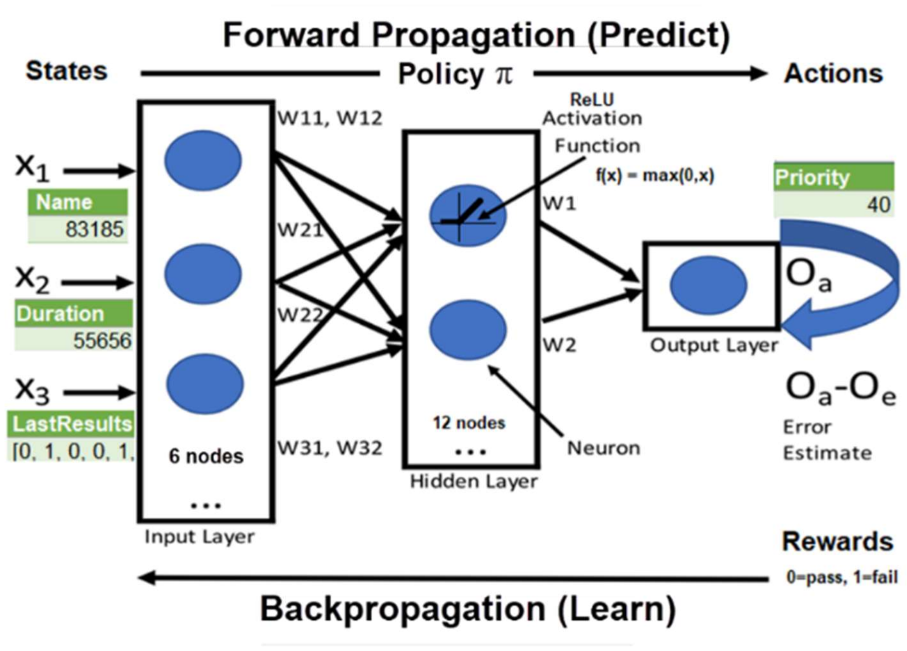
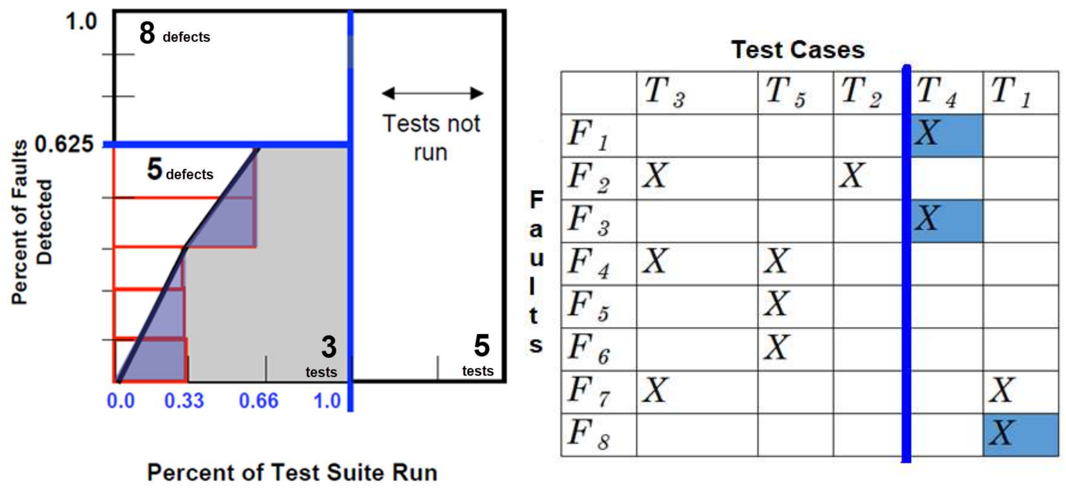


Figure 3. The Neural Network (NN) used by reinforcement learning.

To evaluate the performance of the network, instead of only using percent of faults detected (PFD) the results were compared using the normalized average percentage of faults detected (NAPFD) from (Qu et al., 2007) as an evaluation metric. The goal of using this metric is to detect as many faults *m* with the least test cases run *n* where *p* is the faults detected by executed test cases divided by the faults detected by all test cases and TF_i is the number of test cases that detect fault F_i. In the following example: m=8, n=3, p=5/8. The NAPFD of 44% considers how fast faults are detected, as opposed to the PFD of 62.5% as shown in Figure 4 illustrates a sample calculation of the NAPFD, which is used as a more accurate metric to assess the test suite's effectiveness.

$$NAPFD = p - \frac{TF_1 + TF_2 + \ldots + TF_m}{m \times n} + \frac{p}{2n} = \frac{5}{8} - \frac{0+2+0+2+1+1+1+0}{8 \times 3} + \frac{\frac{5}{8}}{2 \times 3} = 0.44$$



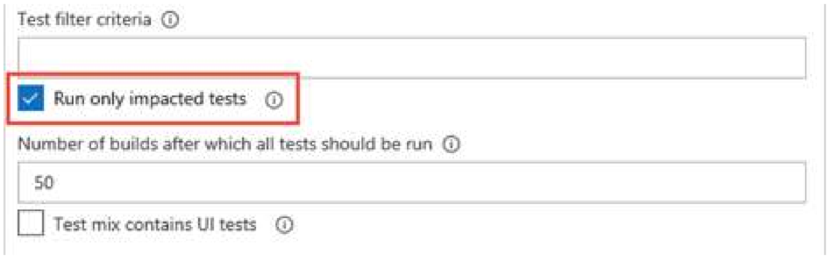
	T_3	T_5	T_2	T_4	T_1
F_1				X	
F_2	X		X		
F_3				X	
F_4	X	X			
F_5		X			
F_6		X			
F_7	X				X
F_8					X

Figure 4. The Normalized Average Percentage of Faults Detected (NAPFD).

Software Testing

One challenge with software testing is the large number of tests required to evaluate complex applications. When there are many test cases within the test suite, the tests can be classified, ordered or prioritized to improve the overall effectiveness or reduce the number of test executions required (Lenz et al., 2013). Some techniques for prioritizing test cases were presented by (Rothermel et al., 2001) in the context of regression testing. They define the prioritization problem, given test suite T, permutations PT of T and function F from PT to real numbers award values so that the best ordering can be determined. Although there are many factors to consider for the award value, some are increased test coverage or faster fault detection. For an approximation of the fault detection potential, the well-established method of mutation score from mutation analysis (Jia and Harman, 2011) is utilized. In the work by (Vincenzi et al., 2006) an incremental approach is taken to limit the time and resource constraints with mutation testing. The mutation testing improvements proposed by this dissertation could improve past research by (Rothermel, et al., 2001) that present non-machine learning techniques for test case prioritization, as well as provide guidance for future research.

The effort and time required to perform testing can also be mitigated by risk-driven testing, as discussed by (Briand, 2008 and Spinellis et al., 2009) where fault prediction models are used to identify potential fault locations and reduced testing effort by prioritizing test cases based on potential risk. Another approach is using Test Impact Analysis (TIA), which is a technique that helps determine which subset of tests need to execute for a given set of code changes. Microsoft has spent significant effort to develop the Test Impact Analysis approach. They have patented the process (Huene et al., 2011) which generates dependency maps between source code changes and tests in automated builds by using test coverage within a data store. It is incorporated within the Visual Studio IDE and Azure DevOps Services. As illustrated in Figure 5, to reduce testing effort during automated builds Test Impact Analysis[1] limits execution to only the test cases that are necessary for code that has been added or updated. This figure illustrates the ability to limit test case execution by selecting 'Run only impacted tests' that have been impacted by related code changes.


Test filter criteria

[✓] Run only impacted tests

Number of builds after which all tests should be run: 50

[ ] Test mix contains UI tests

Figure 5. Test Impact Analysis within Microsoft Azure DevOps Services.

[1] https://blogs.msdn.microsoft.com/devops/2017/03/02/accelerated-continuous-testing-with-test-impact-analysis-part-1/

For additional savings in mutation testing execution time, this dissertation considers a related machine learning approach similar to that of (Menzies et al., 2007 and Huang et al., 2017) using static code attributes (e.g. lines of code, lines of comments) and effort aware attributes (e.g. lines added, line updated, lines deleted), as well as test case metrics to assist with defect predictions. With the idea that the approaches could be combined to improve test case and mutation operator selection.

Parallel Processing

The last significant aspect of this dissertation is the introduction of parallel processing, to reduce the learning time which allows the process to become more practical in real-world software development. The benefits of using parallel methods for reinforcement learning were established by (Nair et al., 2015) but utilized a massively distributed approach, which would not be practical in many software development situations where developers work locally, possibly disconnected or with limited network resources. To address this concern the work by (Mnih, Badia et al., 2016) evaluated various asynchronous methods for deep reinforcement learning, including parallelization using multiple threads locally on computers with multicore CPUs. As stated by (Etiemble, 2018) since the CPU frequency limit was reached there has been a shift towards multicore processors and according to (Patterson, 2010) successful parallel software improves processing efficiency by using the multiple cores. When developing a multi-threaded approach, (Boehm, 2005) expressed the importance to consider concurrency issues as well as the performance benefits and using a language that was originally designed with thread support, such as C#.

Summary

By applying machine learning techniques, the task of mapping input parameters to outputs actions can be accomplished, but care must be taken on using the correct machine learning approaches. The process of software testing can require significant effort in terms of test execution, so choosing to execute fewer tests that still validate the application correctness is beneficial. Reductions in the learning time can be achieved with parallel processing techniques. In the next chapter, a description of the methodology will be presented on how these techniques will be combined for the proposed research to be completed.

Chapter 3 - Methodology

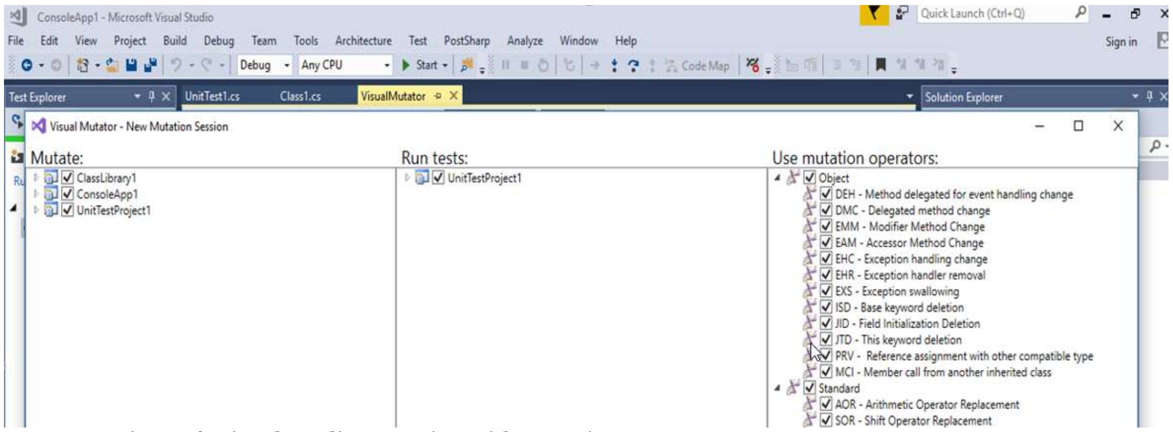
Introduction

The proposed research looks to build a 'real-time' process capable of selecting mutation operators during mutation testing that increases the test suite effectiveness. To achieve this, a parallel reinforcement learning algorithm must be implemented. The algorithm will be measured by the loss and reward values defined earlier.

Approach

Since the idea is to integrate testing within the software development process, the approach must be easily accessible to the software developer. The proposal is to enhance with parallelized ML the approach by (Derezinska, 2006), (Derezińska & Szustek, 2007, 2008) and (Derezińska & Trzpil, 2015) where mutation testing is performed in .NET by Visual Mutator[2], a Visual Studio Integrated Development Environment (IDE) extension.

Mutation testing starts with a selection of code, tests, mutation operators in Figure 6, then mutant generation and finally test suite evaluation in Figure 7. Figure 6 illustrates the ability to manually configure mutation testing within the IDE using all selected mutation operators. Figure 7 illustrates the ability to automatically generate and execute first order mutants (live vs. killed) to validate the test suite. The enhanced extension will utilize reinforcement learning for mutation operator selection.


ConsoleApp1 - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test PostSharp Analyze Window Help
Sign in
Debug
Any CPU
Start
Code Map
Test Explorer
UnitTest1.cs
Class1.cs
VisualMutator
Solution Explorer

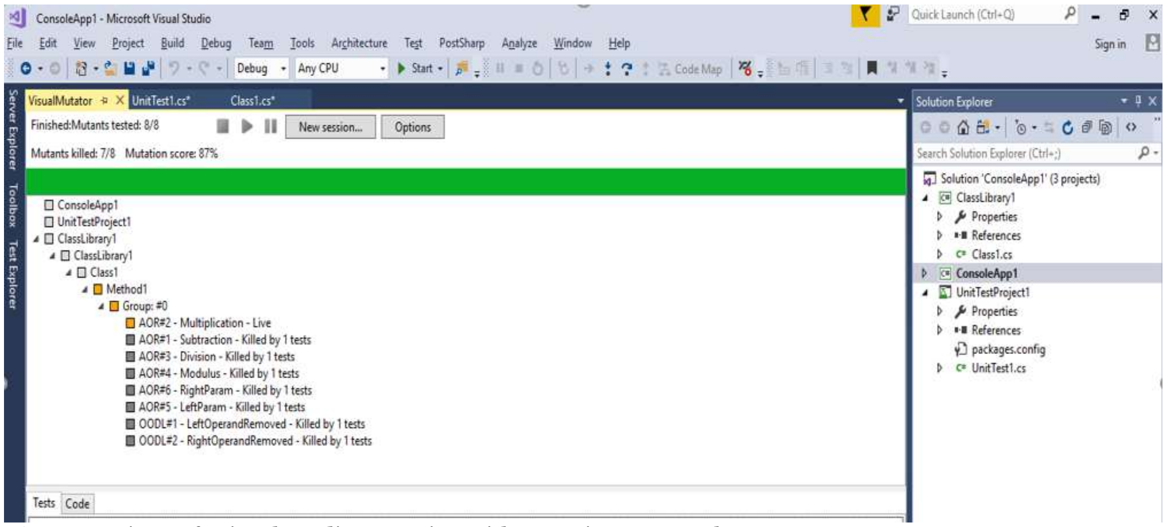
Visual Mutator - New Mutation Session

Mutate:
- ☑ ClassLibrary1
- ☑ ConsoleApp1
- ☑ UnitTestProject1

Run tests:
- ☑ UnitTestProject1

Use mutation operators:
- ☑ Object
  - ☑ DEH - Method delegated for event handling change
  - ☑ DMC - Delegated method change
  - ☑ EMM - Modifier Method Change
  - ☑ EAM - Accessor Method Change
  - ☑ EHC - Exception handling change
  - ☑ EHR - Exception handler removal
  - ☑ EXS - Exception swallowing
  - ☑ ISD - Base keyword deletion
  - ☑ JID - Field Initialization Deletion
  - ☑ JTD - This keyword deletion
  - ☑ PRV - Reference assignment with other compatible type
  - ☑ MCI - Member call from another inherited class
- ☑ Standard
  - ☑ AOR - Arithmetic Operator Replacement
  - ☑ SOR - Shift Operator Replacement

Figure 6. Microsoft Visual Studio extension with mutation operators.



Finished:Mutants tested: 8/8

Mutants killed: 7/8 Mutation score: 87%

- ConsoleApp1
- UnitTestProject1
- ClassLibrary1
  - ClassLibrary1
    - Class1
      - Method1
        - Group: #0
          - AOR#2 - Multiplication - Live
          - AOR#1 - Subtraction - Killed by 1 tests
          - AOR#3 - Division - Killed by 1 tests
          - AOR#4 - Modulus - Killed by 1 tests
          - AOR#6 - RightParam - Killed by 1 tests
          - AOR#5 - LeftParam - Killed by 1 tests
          - OODL#1 - LeftOperandRemoved - Killed by 1 tests
          - OODL#2 - RightOperandRemoved - Killed by 1 tests

Solution 'ConsoleApp1' (3 projects)
- ClassLibrary1
  - Properties
  - References
  - Class1.cs
- ConsoleApp1
- UnitTestProject1
  - Properties
  - References
  - packages.config
  - UnitTest1.cs

Figure 7. Microsoft Visual Studio extension with mutation test results.

[2] https://visualmutator.github.io/web/

To incorporate a more efficient mutation generation process, a machine learning driven suggestion for mutation operators would be incorporated. The suggestions would be based on mutation operator performance during reinforcement learning using code repositories then made available to developers in the context of current program code, similar to Microsoft's IntelliCode feature[3] in Figure 8 that provides Artificial Intelligence (AI) code completion suggestions as stars but requires offline supervised training.

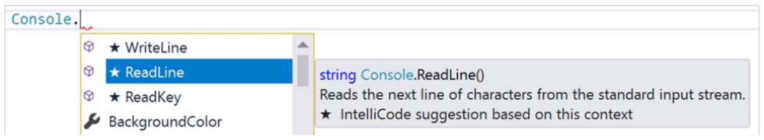

```
Console.
  ★ WriteLine
  ★ ReadLine      string Console.ReadLine()
  ★ ReadKey       Reads the next line of characters from the standard input stream.
  BackgroundColor ★ IntelliCode suggestion based on this context
```

Figure 8. IntelliCode within Microsoft Visual Studio.

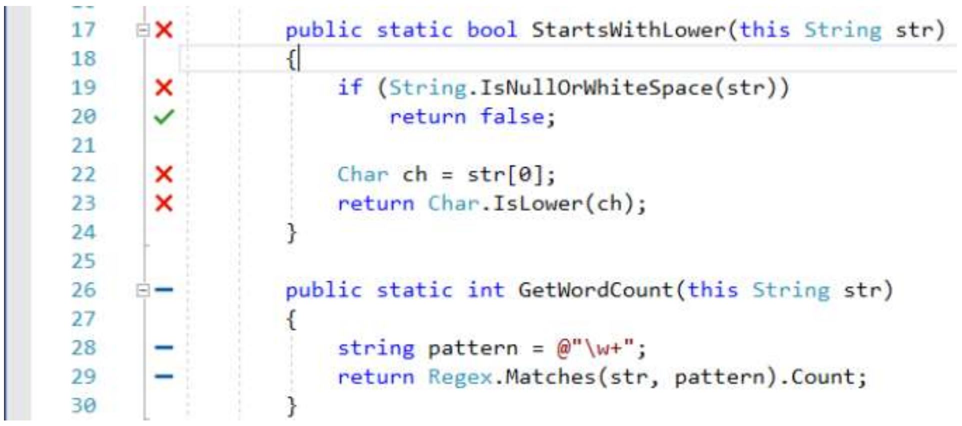
To accomplish the research goals a quantitative approach will be utilized. During the mutation operator selection process, data will be gathered on the number of mutations generated, mutation score and testing execution time. This data can be used to measure and compare the performance of mutants generated with and without the use of machine learning mutation operator selection. The non-machine learning approaches to mutation operator selection will be to 1. Select all operators, 2. Select operators randomly, 3. Select a specific subset of operators. This will help to determine how effective machine learning is at reducing the total number of mutants generated and reducing execution time while continuing to provide an accurate analysis of the test suite.

To reduce test execution, an incremental process to perform mutation testing during program coding would be developed, called 'live' mutation testing. Reinforcement learning is appropriate for mutation operator selection since there is no strict model for the impact of mutations on software system test suites.

[3] https://docs.microsoft.com/en-us/visualstudio/intellicode/faq

With ‘live’ mutation testing, the mutation operators will be selected, mutations will be generated then tests will be selected and executed as the software is developed so missing tests can be identified earlier. This will help to promote shift left (Demeyer et al., 2018) where testing is brought closer to the beginning of the Software Development Lifecycle (SDLC), as opposed to testing towards the end of the SDLC.

The ‘live’ unit testing feature[4] is already available within Microsoft’s Visual Studio IDE and illustrated in Figure 9 where both test coverage evaluation and unit test execution are performed in real-time for test results from the test suite. The test coverage identifies the amount of code tested but ‘live’ unit testing does not guarantee test quality, which is how well does the test suite perform at identifying potential defects?



```
public static bool StartsWithLower(this String str)
{
    if (String.IsNullOrWhiteSpace(str))
        return false;

    Char ch = str[0];
    return Char.IsLower(ch);
}

public static int GetWordCount(this String str)
{
    string pattern = @"\w+";
    return Regex.Matches(str, pattern).Count;
}
```

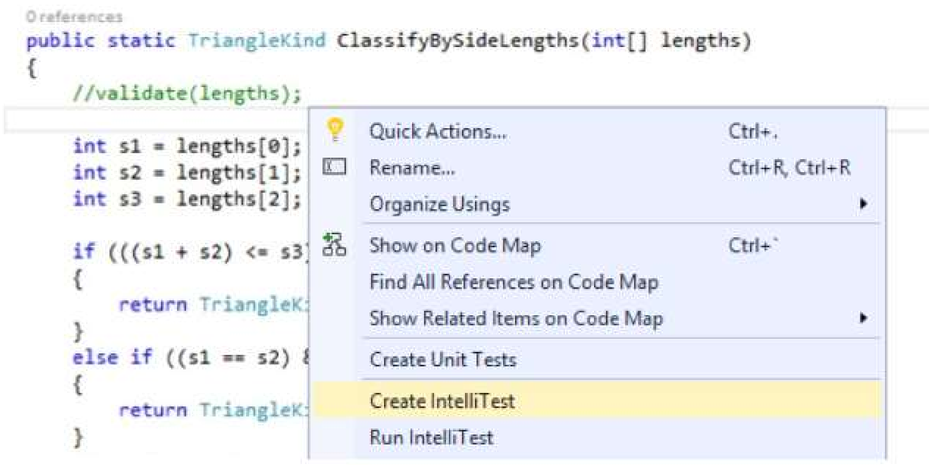
Figure 9. Live Unit Testing within Microsoft Visual Studio.

With ‘live’ mutation testing the goal would be to identify a single syntactic error, placing a higher emphasis on first order mutants (FOM), where mutants are generated by applying a mutation operator once against the source code.

[4] https://docs.microsoft.com/en-us/visualstudio/test/live-unit-testing

This is opposed to testing later when there is more of a chance that multiple errors have been introduced, reducing need for second order mutants (SOM) and higher order mutants (HOM) that simulate multiple syntactic errors. HOMs are often constructed by first formulating the FOMs, then joining them together, which takes longer to compute (Ghiduk et al., 2018).

To execute test case selection and mutation testing the code libraries will need to have associated test suites. With the introduction of Test-Driven Development (TDD) by (Beck, 2003), more test cases are being created by the business and quality analysts that play a role in test development. There are many tools available, including some evaluated by (Honfi & Micskei, 2019) that allow for unit test generation. Microsoft's IntelliTest feature[5] in Figure 10 generates test suites based on program analysis. This figure illustrates how it can automatically generate test suites with high code coverage using automated white box analysis. Since the reachability of program statements is not decidable, the goal (Tillmann & De Halleux, 2008) is to provide a good approximation and high coverage of the program statements.



```
0 references
public static TriangleKind ClassifyBySideLengths(int[] lengths)
{
    //validate(lengths);

    int s1 = lengths[0];
    int s2 = lengths[1];
    int s3 = lengths[2];

    if (((s1 + s2) <= s3)
    {
        return TriangleK
    }
    else if ((s1 == s2) &
    {
        return TriangleK
    }
```

Figure 10. IntelliTest within Microsoft Visual Studio.

[5] https://docs.microsoft.com/en-us/visualstudio/test/intellitest-manual/introduction

Once the tests have been developed, programmers can focus on the task of implementing more complex logic to satisfy the tests. TDD can also lead to a more accurate representation of the requirements since the unit tests are more formalized using structured syntax as opposed to using manual testing processes that rely on requirements documentation with abstract natural language.

For machine learning to be successful, an evaluation of features will be performed, including code metrics (e.g. total number of methods, total lines of code, operator occurrence counts), effort metrics (e.g. new vs. updated classes, new vs. updated methods, modified lines of code) and test metrics (e.g. total number of test cases, test results, test duration, total number of mutants, live vs. killed mutants, mutation score). Given the features, the algorithm would attempt a binary classification and predict usage (i.e. select vs. deselect) for each mutation operation with the objective to limit mutants necessary to evaluate the test suite's effectiveness. For mutation testing, Figure 11 proposes agent prioritizing mutation operators for methods and classes within code repo.

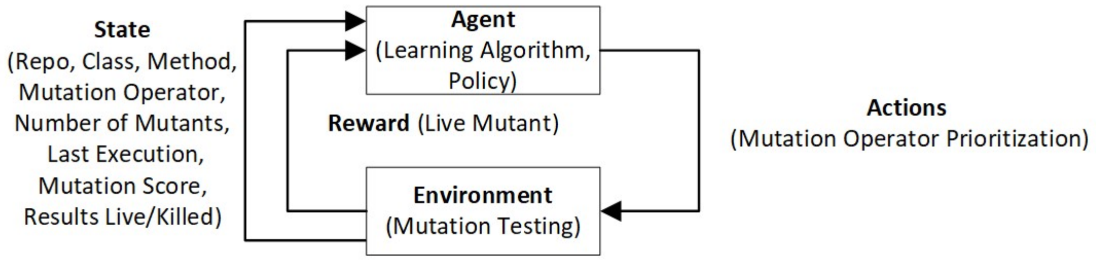

State
(Repo, Class, Method, Mutation Operator, Number of Mutants, Last Execution, Mutation Score, Results Live/Killed)
Agent
(Learning Algorithm, Policy)
Reward (Live Mutant)
Environment
(Mutation Testing)
Actions
(Mutation Operator Prioritization)

Figure 11. Reinforcement Learning for Mutation Operator Selection.

To constantly evaluate the results of the machine learning mutation operator advice, there must be an efficient process to execute reinforcement learning. To meet this demand the core concept of machine learning in Figure 12 the approach will utilize a parallel process having *n* multiple agents, each with a deep Q-network to predict mutation operators based on rewards, as well as randomly sampled shared experience replay to allow the agents to learn from each other. This improves on the approach of (Nair et al., 2015) by using both multi-threaded agents and shared experience replay memory, which was suggested as future work. The results can be evaluated with different network, agent, environment configurations and without synchronization of network gradients (Grounds & Kudenko, 2005) or parallelized stochastic gradient descent addressed by (Recht et al., 2001).

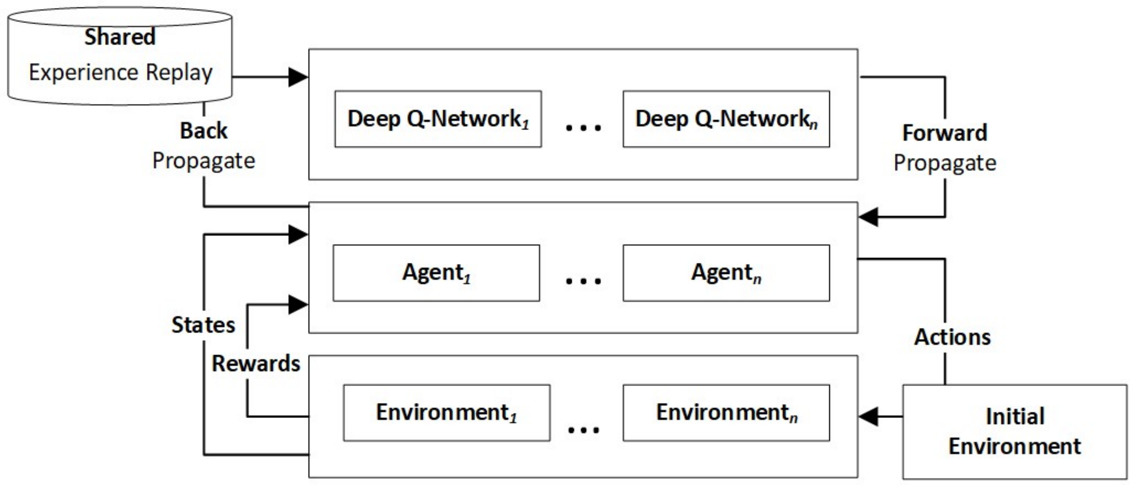


Figure 12. Reinforcement Learning with Parallel Processing.

Experiment Design

To evaluate the approach, as well as issues and barriers previously mentioned, several experiments will be conducted and measured. The proposed experiments are as follows:

- Experiment 1: Learning Mutation Testing with One Live Mutation
- Experiment 2: Learning Mutation Testing with Multiple Live Mutations
- Experiment 3: Learning Mutation Testing with No Live Mutations
- Experiment 4: Comparing Mutation Testing Approaches with Two Cores
- Experiment 5: Comparing Mutation Testing Approaches with Four Cores

Implementation

The algorithm defined in Chapter 3 Methodology; Figure 12 was implemented as a Windows application called Mutation Testing with Parallel Deep Reinforcement Learning (MTPDRL)[6]. The experiments were conducted using Windows Form (MutantTesterDRL.exe) for reinforcement learning and Windows Console (MutantTesting.exe) for mutation testing applications with object-oriented programming in C# using the custom classes in Figure 13. In addition, existing open-source libraries were used, such as Deep-QLearning[7], Mutty[8] and Cecil[9].

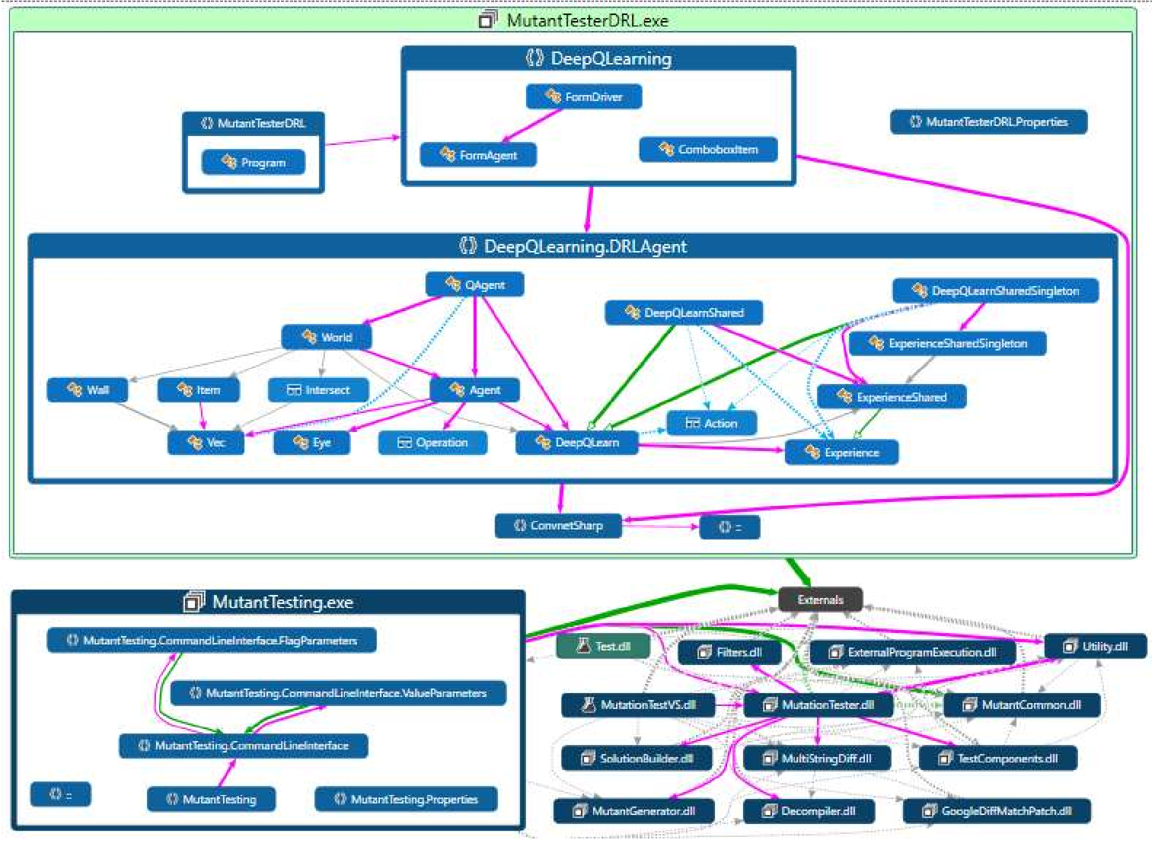


Figure 13 Mutation Testing with Parallel Deep Reinforcement Learning code map.

[6] https://github.com/mstewart1972/MutationTestingWithDeepParallelReinforcementLearning
[7] https://github.com/dubezOniner/Deep-QLearning-Demo-csharp
[8] https://github.com/angusmcintosh/Mutty
[9] https://github.com/jbevain/cecil

MutantTesterDRL.exe

The DeepQLearning.FormDriver class is used to specify parameters and instantiate instances of the DeepQLearning.FormAgent class as thread or process. The FormAgent instantiates the DeepQLearning.DRLAgent.QAgent class which uses the DeepQLearn, DeepQLearnShared or DeepQLearnSharedSingleton classes for reinforcement learning.

DeepQLearn

This class was part of the original Deep-QLearning library and utilizes the Trainer class within the ConvNetSharp library to define and utilize neural networks as part of the reinforcement learning process. There are multiple algorithms supported to update network weights, including the classic Stochastic Gradient Descent but this research utilized ADADELTA by (Zeiler, 2012). The idea with this method of updating the network weights during backpropagation is to prevent the need for manual tuning of the hyperparameters, such as learning rate or momentum and handle adverse conditions with respect to the input data types and network layer units.

DeepQLearnShared

This class was added as an extension for reinforcement learning with shared experience and inherits functionality from the DeepQLearn class. The shared experience replay was implemented using a static ConcurrentDictionary, which is part of the .NET framework System.Collection.Concurrent namespace and is thread-safe. During backpropagation agents will contribute round-robin towards the shared experience, replacing randomly when maximum experience limit is reached and randomly choose a specified batch size number of elements for network training.

DeepQLearnSharedSingleton

This class was added as an extension for reinforcement learning with shared experience but was implemented using the singleton pattern that ensures instantiation is limited to a single instance. The class also allows serialization to save experience.

Experience

This class maintains the state0, action0, reward0, state1 fields where an agent is in state0 and does action0. The environment then assigns reward0 and provides new state, state1. Experience stores this information, which is used during the Q-learning update.

World

This class implements the environment, which is comprised of agents and codebase. The agents utilize actions (i.e., mutation operators) as a means to evaluate the codebase (i.e., code pieces) for rewards (i.e., mutation score). For the experiments, mutation operator selection was evaluated using different methods, including random or machine learning. To maintain the reinforcement learning cycle, the world utilizes a clock that ticks for each forward/backward propagation and can be set with a duration limit. To ensure that the machine learning process converges, DeepQLearning.FormAgent implements criteria (if average Q-learning loss is >=0.50, checking every 100 intervals), that evaluates and resets the experience if the criteria is not met, as shown in Figure 14.

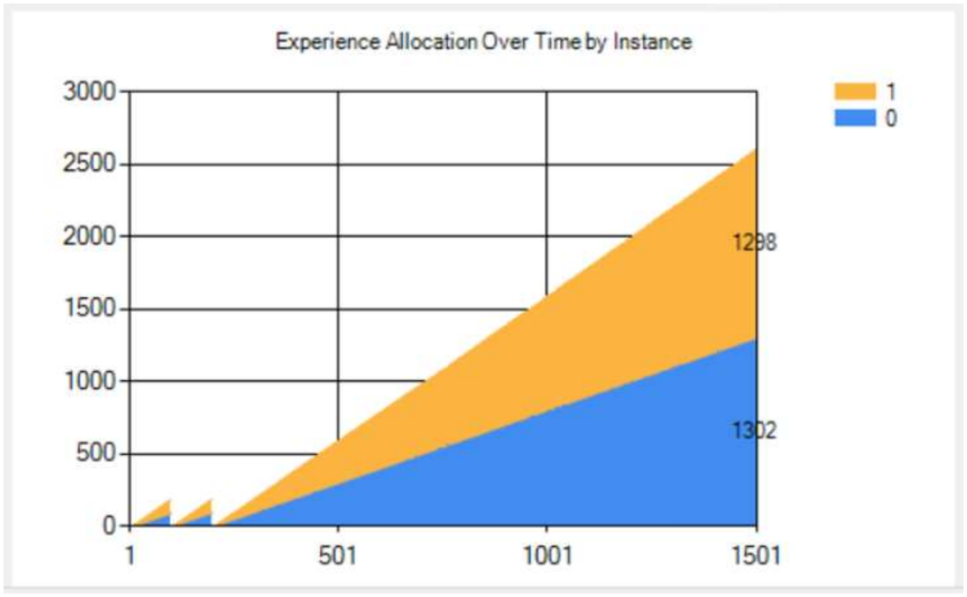


Figure 14. Reinforcement learning with experience reset when criteria not met.

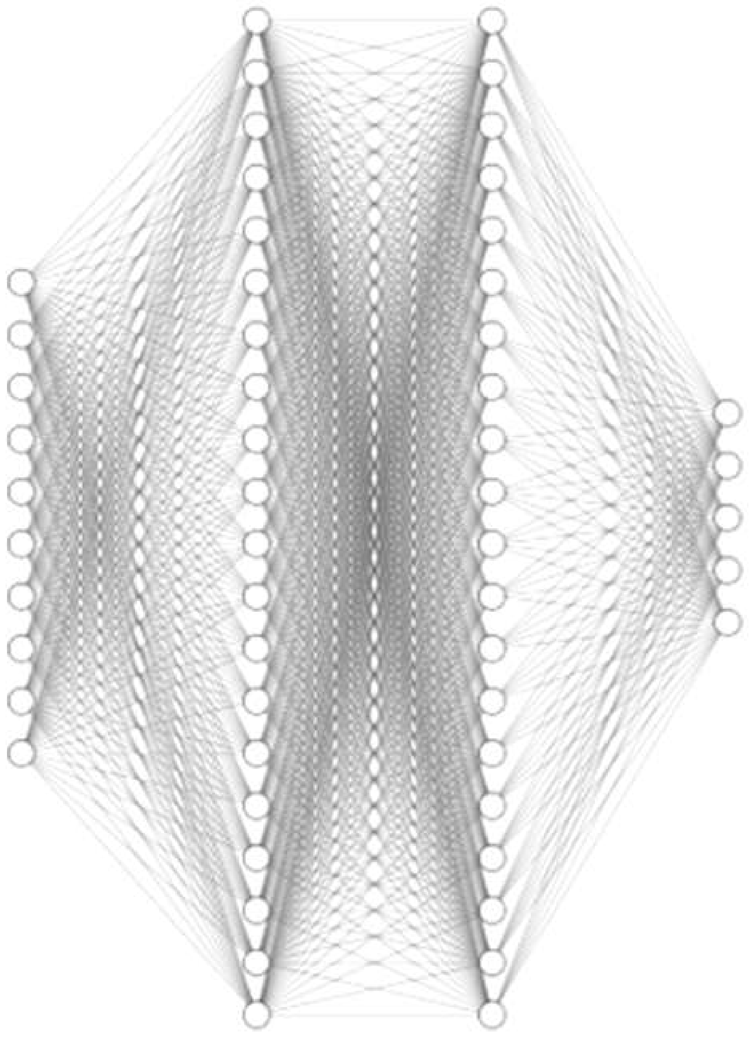
Item

This class implements the rewards, red is positive, and green is negative, that the agent can detect. As shown in Figure 14, items are placed at locations within the environment.

Agent

This class implements the agent and has partial observability within the environment, limited to the module that it is processing. The agent has one eye that can detect item properties using the Eye class, which for these experiments use static values since a single module and class were utilized. The Cecil[8] library provides metadata on modules, types and methods which would allow detecting properties, such as type.name, type.methods.count, type.fields.count to learn within a larger codebase containing multiple modules and types.

The agent has 1 eye, can detect 3 item properties, can take 2 ^ number of mutation operators possible actions and has temporal window of 4, so the number of inputs is current state(1x3) + previous states(1x3x4) + actions(2^4x4) = 79. The item text and integer values are word2vec[9] or one hot encoded as real numbers, which become inputs to the network for forward propagation through the neural network, as shown in Figure 15.



Input Layer $\in \mathbb{R}^{79}$ - Hidden Layer $\in \mathbb{R}^{96}$ - Hidden Layer $\in \mathbb{R}^{96}$ - Output Layer $\in \mathbb{Z}^{16}$

Figure 15. Neural network configuration utilized for reinforcement learning.

[9] https://github.com/tmteam/Word2vec.Tools

The number of actions is 2^number of mutation operators. During research it was determined that the machine learning performed best with limited actions, so the algorithm utilized a mutation category to limit the number of operators. Even though the number of actions can vary between categories, it is fixed to 2^4=16 for the basic arithmetic replacement categories (e.g., basic addition where + is replaced with -, *, /, %). The output is an integer representing one of the possible combinations of the category mutation operators, where each operator is either enabled or disabled, that the agent chooses as action to take for mutation testing. The operation occurrence count of each mutation operator combination utilized is maintained to analyze the agent results.

The reward function computed for backward propagation is favorable to mutation operators that result in live mutations and unfavorable to operators that result in killed mutations. This is accomplished using multiple conditions, as well as factors. First, the reward = min_reward where min_reward = (1 / number mutation operators) * minFactor when there are no live mutations, to promote disabling the most possible operators. Second, reward = score_reward + max_reward where score_reward is 1 - mutation score and max_reward is number mutation operators * max_factor when score_reward != 0, which promotes enabling the most possible operators.

MutantTesting.exe

The MutantTester.MutationTester class and MutationTest() method performs mutation testing based on parameters specified by the DeepQLearning.DRLAgent.Agent class during the Backward() propagation method. The results from the MutantTesting.exe are parsed and the Reinforcement Learning reward is calculated for the DeepQLearning.DRLAgent.DeepQLearn class to retain experience and adjust the network weights using the Trainer class by the Train() method. The reward function looks to select mutation operators that maximize the result of live mutations. A detailed diagram of the mutation testing program is shown in Figure 16.

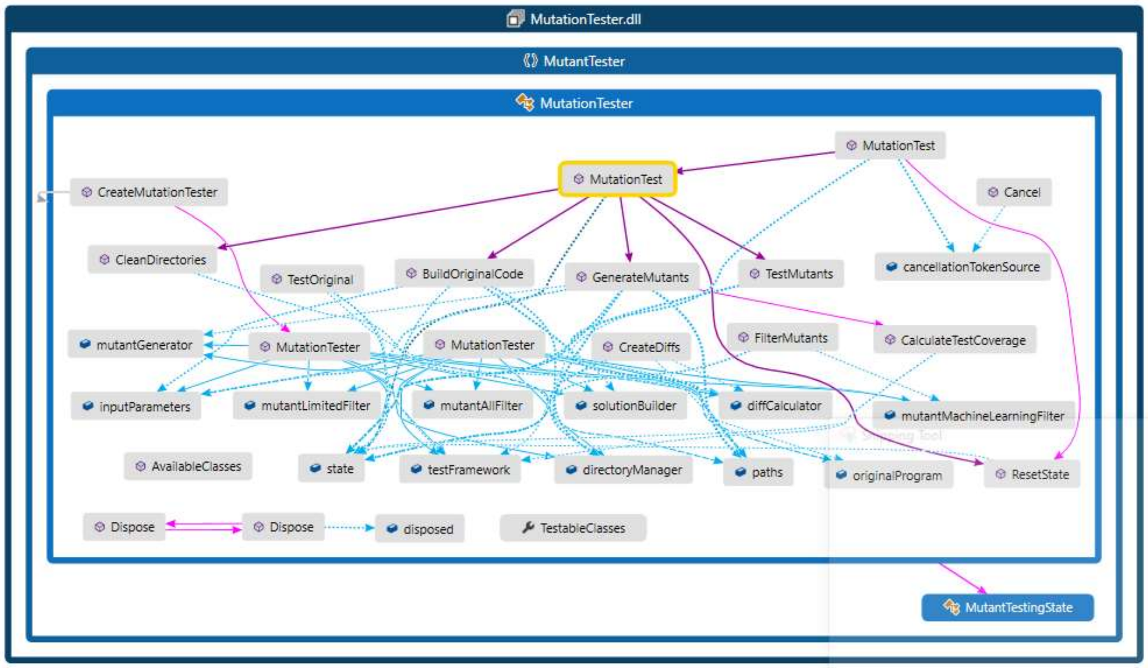


Figure 16. Code map for the Mutation Testing application.

The BuildOriginalCode() method is called by MutationTest() method to compile the .NET solution that contains the program source code for both the application logic and the unit test suite. It utilizes the .NET command-line interface (CLI) and build command to build the project and its dependencies into a set of binaries. The binaries include the project's code in Intermediate Language (IL) files with a .dll extension.

The GenerateMutants() method is called by MutationTest() method, which uses the mutation operators passed to generate mutated copies of the original IL that was built. The MutantGeneration.ReinforcementMutationCreation.ReinforcementMutationFinder class and GetAllReinforcementInstructionMutations() method takes both the mutation category (e.g., `BA=basic addition replacements`) and operators (e.g., `1111-all, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem, 0000-none`), which allows for the reinforcement learning algorithm to choose various mutation operator combinations.

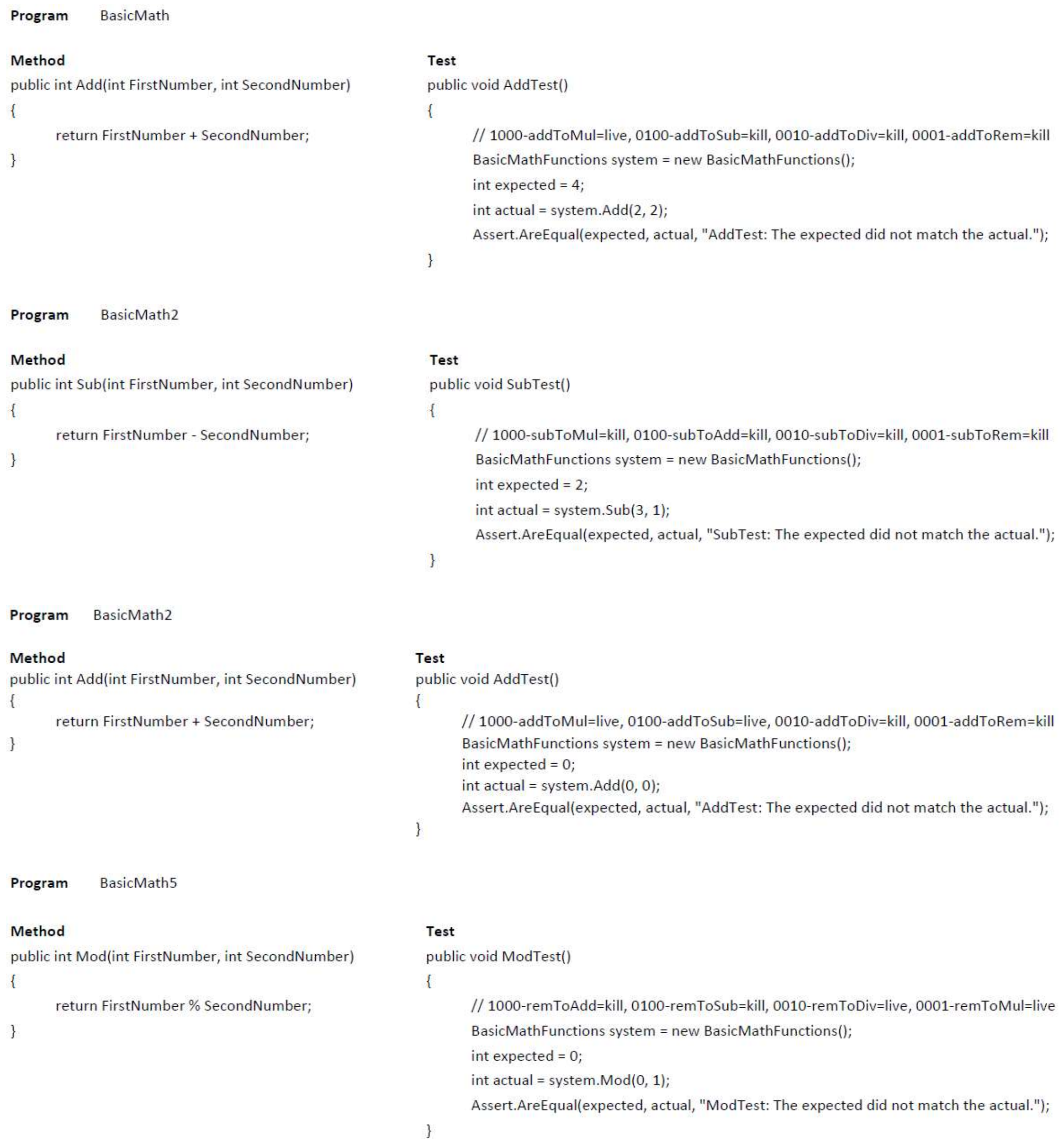
For IL manipulation, the MutantGeneration.MutationGenerators namespace contains classes for the various mutation categories (e.g., InstructionMutationGenerators) that implement the GenerateMutations() method to generate Mutation objects, for each of the classes, methods, or instructions in each of the applications modules. In order to decompile and alter the IL code, the Decompiler.DllDecompiler class uses the Mono.Cecil[8] library.

Finally, the TestMutants() method is called by the MutationTest() method to execute the unit test suite against all of the mutated assemblies. The DotnetTestFramework class and the TestAsync() method supports the MSTest[9], NUnit[10] and xUnit[11] testing frameworks. It utilizes the .NET command-line interface (CLI) and test command to execute the unit tests within the given solution and reports the success or failure of each test. For each test suite execution, results from unit tests are returned as either test fail (i.e., killed mutation) or test pass (i.e., live mutation).

[9] https://github.com/Microsoft/testfx-docs
[10] https://nunit.org/
[11] https://xunit.net/

Datasets

To perform mutation testing, sample programs with test suites were created as shown in Figure 17. These programs perform basic arithmetic operations and corresponding test methods that utilize the NUnit[10] test framework. This allows the experiments to focus on the backpropagation process for mutation operator selection results.



Program BasicMath

Method
```
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber + SecondNumber;
}
```

Test
```
public void AddTest()
{
        // 1000-addToMul=live, 0100-addToSub=kill, 0010-addToDiv=kill, 0001-addToRem=kill
        BasicMathFunctions system = new BasicMathFunctions();
        int expected = 4;
        int actual = system.Add(2, 2);
        Assert.AreEqual(expected, actual, "AddTest: The expected did not match the actual.");
}
```

Program BasicMath2

Method
```
public int Sub(int FirstNumber, int SecondNumber)
{
        return FirstNumber - SecondNumber;
}
```

Test
```
public void SubTest()
{
        // 1000-subToMul=kill, 0100-subToAdd=kill, 0010-subToDiv=kill, 0001-subToRem=kill
        BasicMathFunctions system = new BasicMathFunctions();
        int expected = 2;
        int actual = system.Sub(3, 1);
        Assert.AreEqual(expected, actual, "SubTest: The expected did not match the actual.");
}
```

Program BasicMath2

Method
```
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber + SecondNumber;
}
```

Test
```
public void AddTest()
{
        // 1000-addToMul=live, 0100-addToSub=live, 0010-addToDiv=kill, 0001-addToRem=kill
        BasicMathFunctions system = new BasicMathFunctions();
        int expected = 0;
        int actual = system.Add(0, 0);
        Assert.AreEqual(expected, actual, "AddTest: The expected did not match the actual.");
}
```

Program BasicMath5

Method
```
public int Mod(int FirstNumber, int SecondNumber)
{
        return FirstNumber % SecondNumber;
}
```

Test
```
public void ModTest()
{
        // 1000-remToAdd=kill, 0100-remToSub=kill, 0010-remToDiv=live, 0001-remToMul=live
        BasicMathFunctions system = new BasicMathFunctions();
        int expected = 0;
        int actual = system.Mod(0, 1);
        Assert.AreEqual(expected, actual, "ModTest: The expected did not match the actual.");
}
```

Figure 17. Sample programs with test suites for mutation experiments.

By using a reinforcement learning algorithm, some of the data required for learning is generated by the agent itself by trial-and-error actions within the environments. This is unlike supervised learning, where large amounts of labeled data with the correct input-output pairs are explicitly presented. Most of the reinforcement learning happens online, as the agent interacts with the environment over several iterations and eventually begins to learn the policy that describes which actions to maximize the reward. This was one of the driving factors for choosing RL as opposed to other ML approaches.

To perform additional mutation testing, additional code libraries can be identified. Now that a number of high-profile C# software development organizations, including Microsoft have transitioned to an open-source approach, including test suites available for analysis. In the research from (Derezinska, 2006) the author evaluates mutation testing operators using an array of subject C# programs, including NUnit[10], NHibernate, NAnt and Microsoft's Mono which in 2001 was an early attempt at open-sourcing the .NET Common Language Infrastructure (CLI) for cross-platform portability. In subsequent research on mutation testing tools from the same author (Derezinska, & Szustek, 2008), only two years later there were more C# programs available for analysis. These included Spring.NET, Castle.Core, NCover and CruiseControl.NET. Since then, even more open-source C# libraries have been made available on GitHub with Microsoft's open-source re-development of the .NET Standard called .NET Core, which includes runtime, framework, compiler and tool components. Using open-source projects prevents the extra effort and potential legal issues with commercial data, as well as allows future researchers to validate and contribute to the goals set forth by this dissertation.

To evaluate forward propagation of machine learning features, more complicated programs with multiple classes and assemblies will be required. Additional data that is required could be obtained using code, build and test metrics from the continuous integration of open-source libraries on public GitHub repositories as shown in Figure 18.

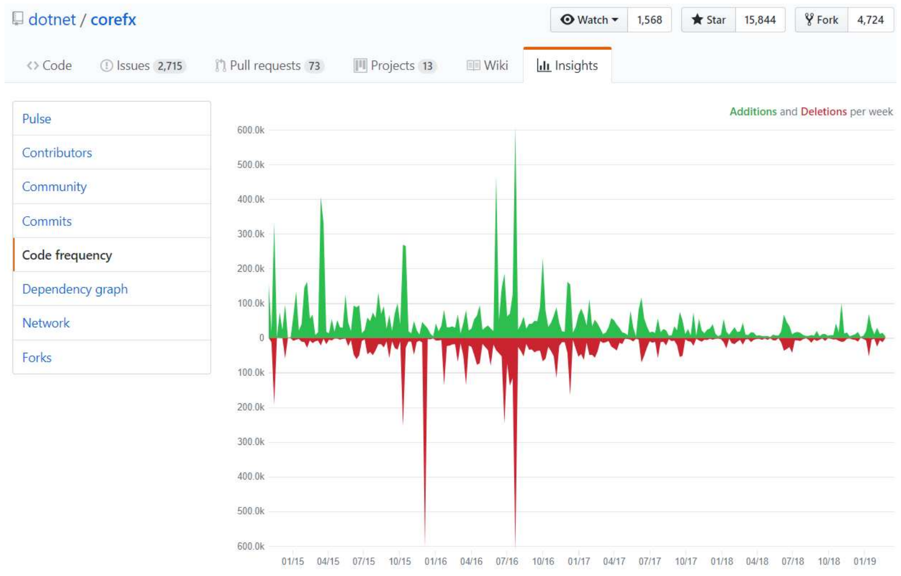


Figure 18. Code churn metrics within GitHub.

To perform test case selection evaluation, datasets are available that provide test case results and have been used by previous research. This idea for 'live' mutation testing uses an approach similar to that of (Madeyski, & Kawalerowicz, 2017) when capturing data for their continuous defect prediction process. There are other public datasets available, including Kaggle.com and governmental organizations, such as NASA that have been used by previous research on software fault analysis (Menzies et al., 2007).

Measures

For an evaluation of reinforcement learning for mutation testing, the experiments will use measurements: 1. Loss, 2. Reward, 3. Elapsed time, 4. Mutation score and 5. CPU %.

Experiment 1: Learning Mutation Testing with One Live Mutation

The purpose of this experiment is to determine if reinforcement learning can identify the optimal mutation operator selection for a program and test suite that has one possible live mutation. The BasicMath program, unit test and basic addition mutation in Figure 23 will be used. In this scenario, the algorithm should identify that the combination of 1000 is the correct combination to turn off all but the one mutation operator (i.e., + to *) that will produce live mutant and identify faulty test case. The environment will allow the agent to run until the reward converges or 24 hours. This first experiment's success criteria are the ability for the reward function to converge and train the agent to successfully navigate the environment, maximizing rewards and correct operator selection. The failure criteria are the inability of reinforcement learning to train the agent successfully or cause loss function to reside in local minima. These results will be documented and utilized as justification for subsequent experiments. The result from this experiment will be formatted as Table 1.

Experiment Results Format	success	fail	
test run	1	2	...
average elapsed time (hh:mm:ss)	**hh:mm:ss**	**hh:mm:ss**	
average loss	#	#	
average reward	#	#	
average mutants total	#	#	
average mutants kill	#	#	
average mutants live	#	#	
average mutation score (kill/total)	#	#	
mutation operator combination	**1000**	0000,0001,0010,0011,0100, 0101,0110,0111,1001,1010, 1011,1100,1101,1110,1111	

Table 1. Experiment 1 results format.

Experiment 2: Learning Mutation Testing with Multiple Live Mutations

The purpose of this experiment is to determine if reinforcement learning can identify the optimal mutation operator selection for a program and test suite that has multiple live mutations. The BasicMath5 program, unit test and basic modulo mutation in Figure 23 will be used. In this scenario, the algorithm should identify that the combination with 0011 is the correct combination to turn off all but two mutation operators (i.e., % to / and % to *) that will produce live mutants and identify faulty test cases.

Experiment Results Format	success	fail	
test run	1	2	...
average elapsed time (hh:mm:ss)	hh:mm:ss	hh:mm:ss	
average loss	#	#	
average reward	#	#	
average mutants total	#	#	
average mutants kill	#	#	
average mutants live	#	#	
average mutation score (kill/total)	#	#	
mutation operator combination	0011	0000,0001,0010,0100,0101, 0110,0111,1000,1001,1010, 1011,1100,1101,1110,1111	

Table 2. Experiment 2 results format.

The success criteria will be similar to the first experiment in that the agent must successfully navigate the environment, maximizing rewards and correct operator selection. The result from this experiment will be formatted as Table 2.

Experiment 3: Learning Mutation Testing with No Live Mutations

The purpose of this experiment is to determine if reinforcement learning can identify the optimal mutation operator selection for a program and test suite that has no possible live mutations. In this scenario, since all mutations are killed, the algorithm should identify 0000 is the correct combination to turn off all mutation operators since none will produce live mutants that identify faulty test cases. The BasicMath2 program, unit test and basic subtraction mutation in Figure 23 will be used.

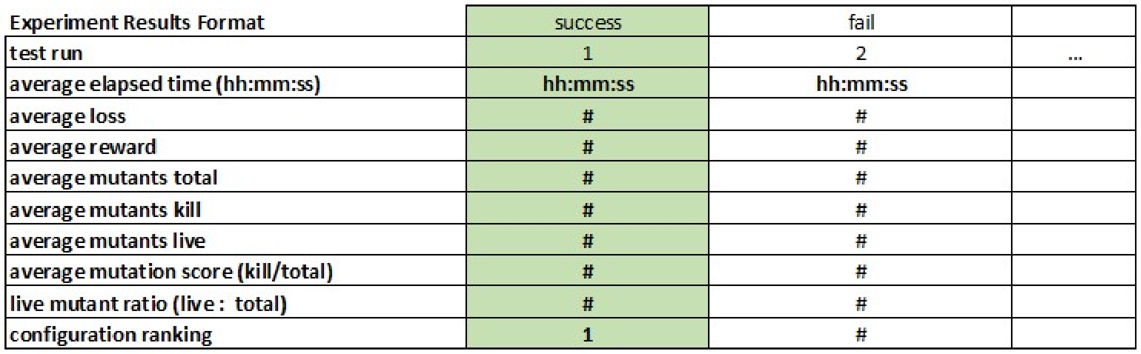
Experiment Results Format	success	fail	
test run	1	2	...
average elapsed time (hh:mm:ss)	hh:mm:ss	hh:mm:ss	
average loss	#	#	
average reward	#	#	
average mutants total	#	#	
average mutants kill	#	#	
average mutants live	#	#	
average mutation score (kill/total)	#	#	
mutation operator combination	0000	0001,0010,0011,0100, 0101, 0110,0111,1000,1001,1010, 1011,1100,1101,1110,1111	

Table 3. Experiment 3 results format.

The success criteria will be similar to the first two experiments in that the agent must successfully navigate the environment, maximizing rewards and correct operator selection. The result from this experiment will be formatted as Table 3.

Experiment 4: Comparing Mutation Testing Approaches with Two Cores

The purpose of this experiment is to evaluate the impact of parallel deep reinforcement learning selection of mutation operators vs. selection of all or random operators using agents as multiple threads on the mutation testing and operating system performance. During reinforcement learning, 2 agents with duration of 1500 intervals for 5 runs will be executed on a laptop with 2 physical cores, for total of 2*1500*5 = 15k tests. Each run will execute until reward convergence is determinate, based on the baseline experiment results. The average should mitigate the risk of anomalies. For this experiment operating system performance metrics will be collected using Windows process explorer, as proposed by (Huffman, 2014). This experiment will guide the development of Visual Studio extension for mutation testing operator selection. The BasicMath2 program, unit test and basic addition mutation in Figure 23 will be used.



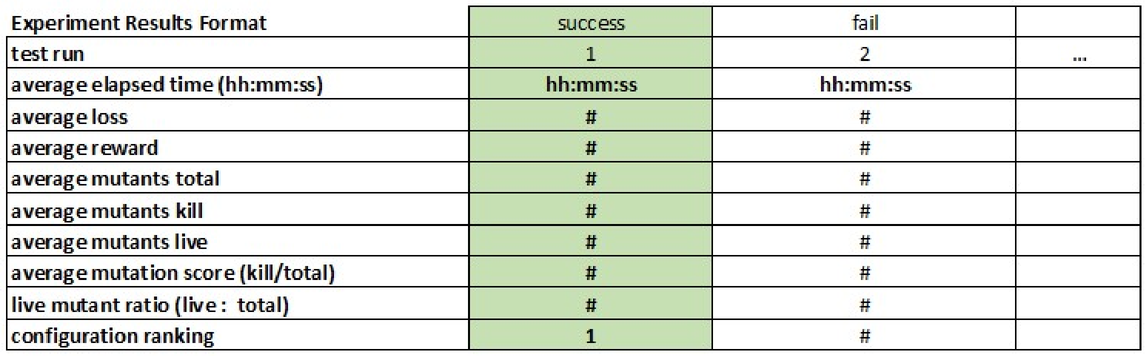
Experiment Results Format	success	fail	
test run	1	2	...
average elapsed time (hh:mm:ss)	hh:mm:ss	hh:mm:ss	
average loss	#	#	
average reward	#	#	
average mutants total	#	#	
average mutants kill	#	#	
average mutants live	#	#	
average mutation score (kill/total)	#	#	
live mutant ratio (live : total)	#	#	
configuration ranking	1	#	

Table 4. Experiment 4 results format.

For this experiment a ranking will be assigned to the different configurations based on the metric of live mutant ratio, which is calculated as average mutants live / average mutants total. The success criteria will be similar to the previous experiment in that the agent must successfully navigate the environment but in addition will include top configuration ranking metric. The result from this experiment will be formatted as Table 4.

Experiment 5: Comparing Mutation Testing Approaches with Four Cores

The purpose of this experiment is to evaluate the impact of parallel deep reinforcement learning selection of mutation operators vs. selection of all or random operators using agents as multiple threads on the mutation testing and operating system performance. During reinforcement learning, 2 agents with duration of 1500 intervals for 5 runs will be executed on a laptop with 4 physical cores, for total of 2*1500*5 = 15k tests. Each run will execute until reward convergence is determinate, based on the baseline experiment results. The average should mitigate the risk of anomalies. For this experiment operating system performance metrics will be collected using Windows process explorer, as proposed by (Huffman, 2014). This experiment will also guide the development of Visual Studio extension for mutation testing operator selection. The BasicMath2 program, unit test and basic addition mutation in Figure 23 will be used.



Experiment Results Format	success	fail	
test run	1	2	...
average elapsed time (hh:mm:ss)	hh:mm:ss	hh:mm:ss	
average loss	#	#	
average reward	#	#	
average mutants total	#	#	
average mutants kill	#	#	
average mutants live	#	#	
average mutation score (kill/total)	#	#	
live mutant ratio (live : total)	#	#	
configuration ranking	1	#	

Table 5. Experiment 5 results format.

For this experiment a ranking will be assigned to the different configurations based on the metric of live mutant ratio, which is calculated as average mutants live / average mutants total. The success criteria will be similar to the previous experiment in that the agent must successfully navigate the environment but in addition will include top configuration ranking metric. The result from this experiment will be formatted as Table 5.

Resources

For this research, the following basic and available resources were required:

- Laptop – Developer machine with 2 physical Intel ® Core® CPU @2.50GHz processors (4 logical processors), 16GB memory (L1 cache:256KB, L2 cache:1MB, L3 cache:8MB) and Windows 10 64-bit operating system.

- Laptop – Developer machine with 4 physical Intel ® Xeon® CPU @3.00GHz processors (8 logical processors), 16GB memory (L1 cache:256KB, L2 cache:1MB, L3 cache:8MB) and Windows 10 64-bit operating system.

- Programming software – The C# programming language (Microsoft Corporation, 2013) and Visual Studio integrated development environment (IDE).

- Analysis software – Windows process explorer (Microsoft Corporation, 2019).

- Documentation software – Microsoft Office (2019).

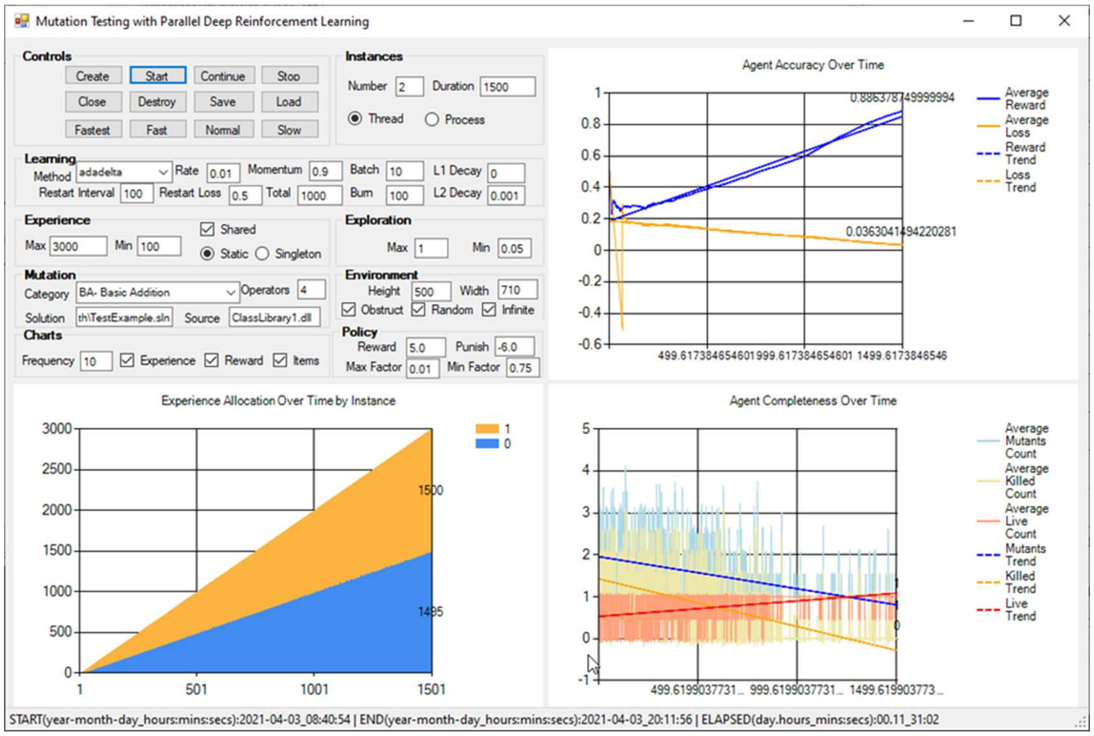
Summary

The experiments will be performed while also running other developer applications, including Visual Studio, Microsoft Outlook, Microsoft Word, Microsoft Excel, Microsoft Teams, Chrome Internet Browser. This will help to determine the feasibility of running the reinforcement learning process in real-world situations and provide a better estimate of the metrics captured in the experiment results.

Chapter 4 – Results

Introduction

The experiments previously designed were conducted. To execute the experiments a sophisticated multi-thread, multi-process test-harness application described in the implementation section was utilized, Mutation Testing with Parallel Deep Reinforcement Learning (MTPDRL)[6] is shown in Figure 19. It was based on the Q-learning research by (Mnih, Kavukcuoglu, et al., 2013) and the aforementioned Deep-QLearning[6] library that implemented reinforcement learning using a single-threaded process. The MTPDRL application was built to specify parameters, execute experiments and visualize data. The output data was collected, aggregated and prepared for the following results.


Mutation Testing with Parallel Deep Reinforcement Learning

Controls

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ✓ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

Mutation: Category BA- Basic Addition, Operators 4, Solution th\TestExample.sln, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ✓ Obstruct, ✓ Random, ✓ Infinite

Charts: Frequency 10, ✓ Experience, ✓ Reward, ✓ Items

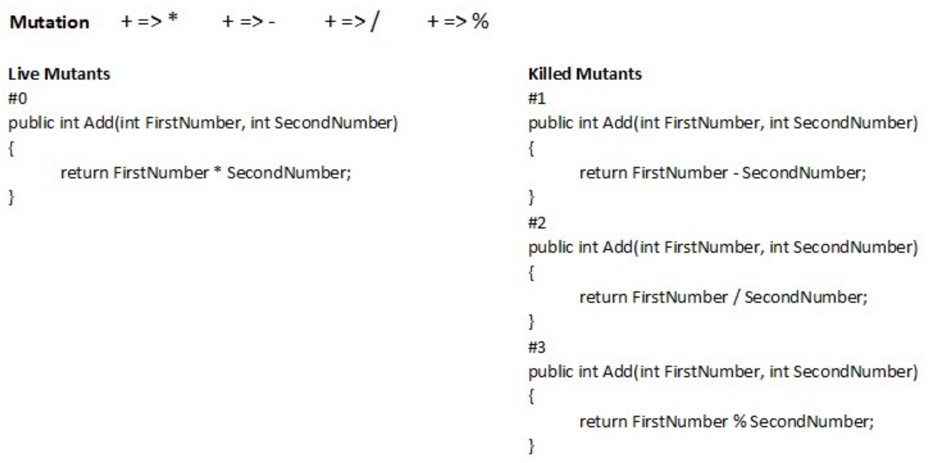
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-04-03_08:40:54 | END(year-month-day_hours:mins:secs):2021-04-03_20:11:56 | ELAPSED(day.hours_mins:secs):00.11_31:02

Figure 19. Mutation Testing with Parallel Deep Reinforcement Learning.

Experiment 1: Learning Mutation Testing with One Live Mutation

The purpose of this experiment is to determine if reinforcement learning can identify the optimal mutation operator selection for a program and test suite that has one possible live mutation. The addition mutants possibly generated are shown in Figure 20.



Mutation + => * + => - + => / + => %

Live Mutants

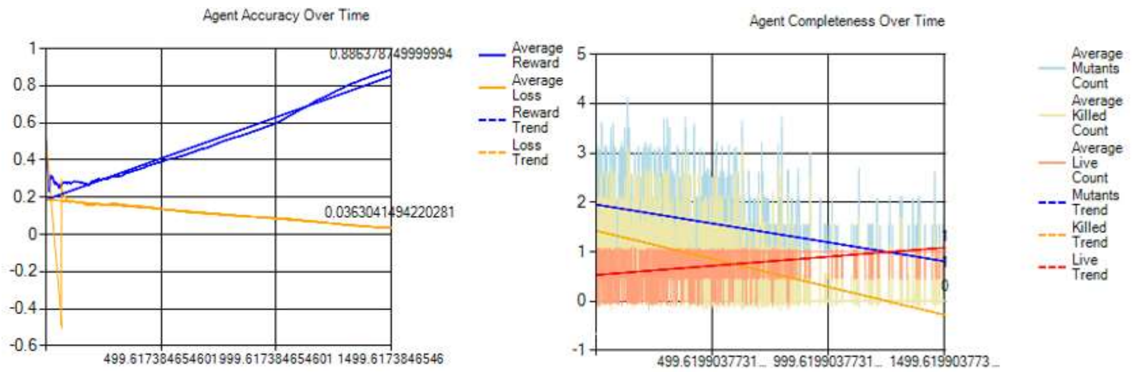
```
#0
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber * SecondNumber;
}
```

Killed Mutants

```
#1
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber - SecondNumber;
}
#2
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber / SecondNumber;
}
#3
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber % SecondNumber;
}
```

Figure 20. Possible mutants with one live mutant for experiment 1.

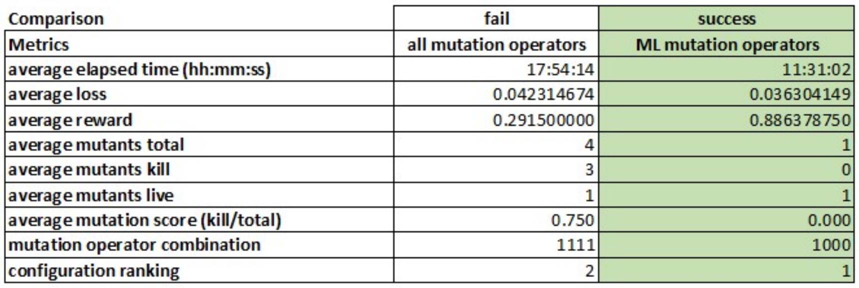
The testing indicated the learning algorithm convergence was definitive at 1500 cycles. At that point, the machine learning actions shown in Figure 21 were evaluated and the 1000 combination had the highest occurrence and identified as recommended mutation.



```
Actions:
(combination,count)
(0000, 38), (0001, 32), (0010, 34), (0011, 36), (0100, 42), (0101, 39), (0110, 35), (0111, 32), (1000, 804), (1001, 42), (1010, 45), (1011, 33),
(1100, 28), (1101, 36), (1110, 36), (1111, 32)
```

Figure 21. ML agent reward, loss and mutation performance for one live mutant.

The results indicated that reinforcement learning using agent for mutation operator selection was successful, obtaining high reward with low loss, generating and testing fewer mutations after training for approximately ~11.5 hours vs. all operators executing for ~18 hours as shown in Table 6. Additional details on the individual agent performance from this and all experiments are available within the appendix.

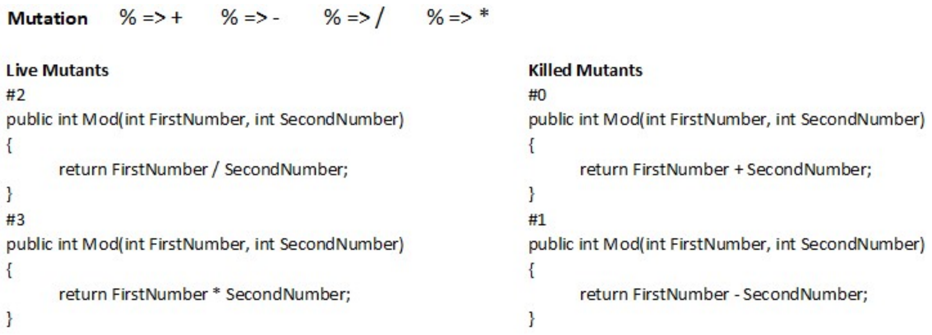


Comparison	fail	success
Metrics	all mutation operators	ML mutation operators
average elapsed time (hh:mm:ss)	17:54:14	11:31:02
average loss	0.042314674	0.036304149
average reward	0.291500000	0.886378750
average mutants total	4	1
average mutants kill	3	0
average mutants live	1	1
average mutation score (kill/total)	0.750	0.000
mutation operator combination	1111	1000
configuration ranking	2	1

Table 6. Learning Mutation Testing with One Live Mutation.

Experiment 2: Learning Mutation Testing with Multiple Live Mutations

The purpose of this experiment is to determine if reinforcement learning can identify the optimal mutation operator selection for a program and test suite that has multiple live mutations. The modulo mutants possibly generated are shown in Figure 22.



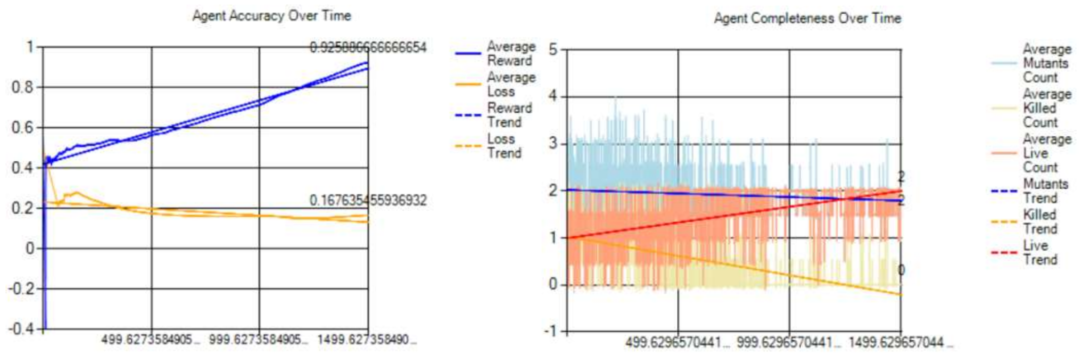
Mutation % => + % => - % => / % => *

Live Mutants
```
#2
public int Mod(int FirstNumber, int SecondNumber)
{
        return FirstNumber / SecondNumber;
}
#3
public int Mod(int FirstNumber, int SecondNumber)
{
        return FirstNumber * SecondNumber;
}
```

Killed Mutants
```
#0
public int Mod(int FirstNumber, int SecondNumber)
{
        return FirstNumber + SecondNumber;
}
#1
public int Mod(int FirstNumber, int SecondNumber)
{
        return FirstNumber - SecondNumber;
}
```

Figure 22. Possible mutants with two live mutants for experiment 2.

An observation was the learning algorithm, including shared agent experience continued to converge after attempting various actions with multiple live mutants around 1500 cycles as shown in Figure 23 and the 0011 combination had the highest action occurrence and thus was identified as recommended mutation.



```
Actions:
(combination,count)
(0000, 32), (0001, 308), (0010, 59), (0011, 665), (0100, 43), (0101, 27), (0110, 37), (0111, 43), (1000, 27), (1001, 37), (1010, 29), (1011, 34),
(1100, 43), (1101, 41), (1110, 47), (1111, 29)
```

Figure 23. ML agent reward, loss and mutation performance for multiple live mutants.

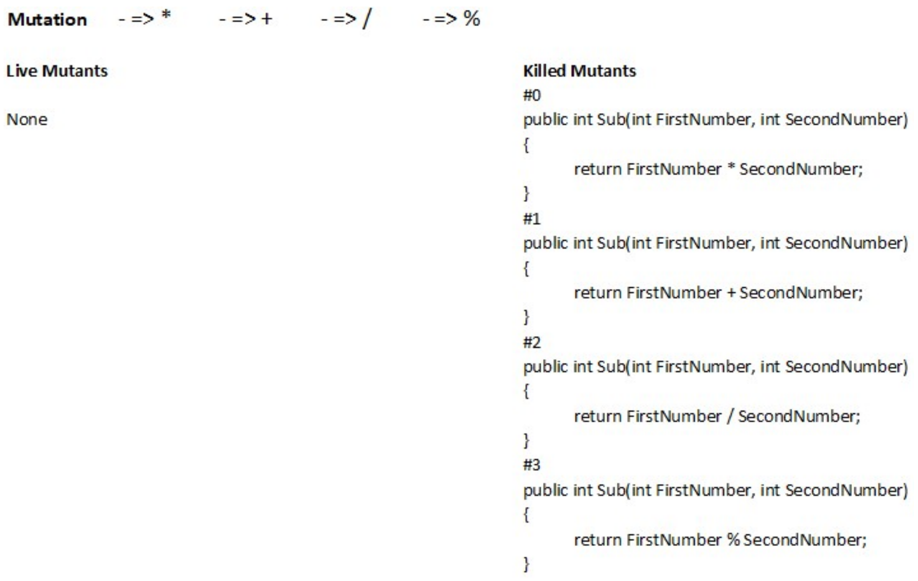
The results indicated that reinforcement learning using an agent for mutation operator selection was successful, obtaining high reward with low loss, generating and testing fewer mutations after training for approximately ~11 hours as shown in Table 7.

Comparison	fail	success
Metrics	all mutation operators	ML mutation operators
average elapsed time (hh:mm:ss)	13:29:11	11:15:18
average loss	0.102743483	0.167635456
average reward	0.540250000	0.925886667
average mutants total	4	2
average mutants kill	2	0
average mutants live	2	2
average mutation score (kill/total)	0.500	0.000
mutation operator combination	1111	0011

Table 7. Learning Mutation Testing with Multiple Live Mutations.

Experiment 3: Learning Mutation Testing with No Live Mutations

The purpose of this experiment is to determine if reinforcement learning can identify the optimal mutation operator selection for a program and test suite that has no possible live mutations. The subtraction mutants possibly generated are shown in Figure 24.



Mutation - => * - => + - => / - => %

Live Mutants

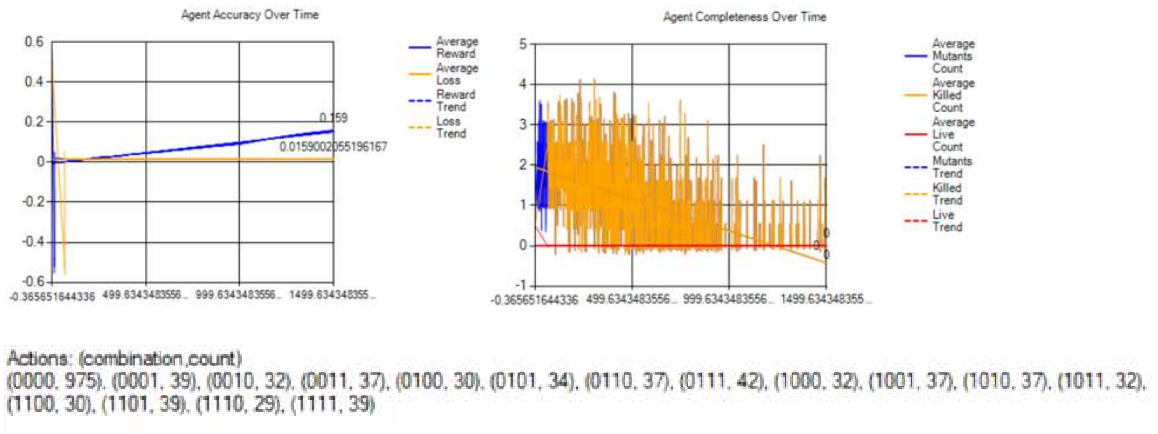
None

Killed Mutants

```
#0
public int Sub(int FirstNumber, int SecondNumber)
{
        return FirstNumber * SecondNumber;
}
#1
public int Sub(int FirstNumber, int SecondNumber)
{
        return FirstNumber + SecondNumber;
}
#2
public int Sub(int FirstNumber, int SecondNumber)
{
        return FirstNumber / SecondNumber;
}
#3
public int Sub(int FirstNumber, int SecondNumber)
{
        return FirstNumber % SecondNumber;
}
```

Figure 24. Possible mutants with no live mutants for experiment 3.

An observation was the learning algorithm, including shared agent experience continued to converge with multiple live mutants around 1500 cycles as shown in Figure 25 and the 0000 combination had the highest occurrence and identified as recommended mutation.



```
Actions: (combination,count)
(0000, 975), (0001, 39), (0010, 32), (0011, 37), (0100, 30), (0101, 34), (0110, 37), (0111, 42), (1000, 32), (1001, 37), (1010, 37), (1011, 32),
(1100, 30), (1101, 39), (1110, 29), (1111, 39)
```

Figure 25. ML agent reward, loss and mutation performance for no live mutants.

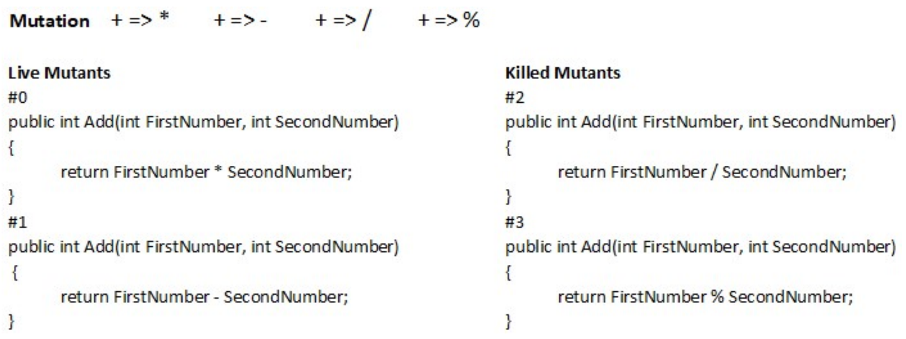
The results indicated that reinforcement learning using an agent for mutation operator selection was successful, obtaining high reward with low loss, generating and testing significantly fewer mutations after training for approximately ~11.5 hours as shown in Table 8.

Comparison	fail	success
Metrics	all mutation operators	ML mutation operators
average elapsed time (hh:mm:ss)	12:26:58	11:28:59
average loss	0.015536947	0.015900206
average reward	0.000000000	0.159000000
average mutants total	4	0
average mutants kill	4	0
average mutants live	0	0
average mutation score (kill/total)	1.000	NaN
mutation operator combination	1111	0000

Table 8. Learning Mutation Testing with No Live Mutations.

Experiment 4: Comparing Mutation Testing Approaches with Two Cores

The purpose of this experiment is to evaluate the impact of parallel deep reinforcement learning selection of mutation operators vs. selection of all or random operators using agents as multiple threads on the mutation testing and operating system performance. The addition mutants possibly generated are shown in Figure 26.



Mutation + => * + => - + => / + => %

Live Mutants
```
#0
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber * SecondNumber;
}
#1
public int Add(int FirstNumber, int SecondNumber)
 {
        return FirstNumber - SecondNumber;
}
```

Killed Mutants
```
#2
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber / SecondNumber;
}
#3
public int Add(int FirstNumber, int SecondNumber)
{
        return FirstNumber % SecondNumber;
}
```

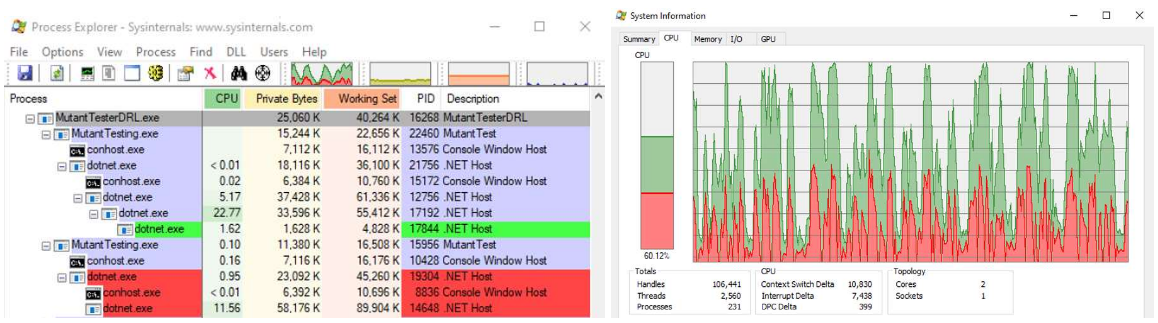
Figure 26. Possible mutants with two live mutants for experiment 4.

The results indicated that the machine learning mutation operator selection process was able to outperform both the traditional approach of selecting all operators, as well as random selection as shown in Table 9.

Comparison	fail	fail	success
Metrics	all mutation operators	random mutation operators	ML mutation operators
average elapsed time (hh:mm:ss)	33:53:02	16:25:42	13:52:11
average loss	0.130015461	0.126778509	0.152961444
average reward	0.549300000	0.504345667	0.798272917
average mutants total	4	2.5	1.4
average mutants kill	2	1.7	0.3
average mutants live	2	0.8	1.1
average mutation score (kill/total)	0.500	0.680	0.214
live mutant ratio (live : total)	0.500	0.320	0.786
configuration ranking	2	3	1

Table 9. Comparing Mutation Testing Approaches with Two Cores.

An observation was that the reinforcement learning selection was able to generate the highest live to total mutant ratio, which resulted in a significant reduction in the mutation testing elapsed time. The driver thread (MutantTesterDRL.exe) maintained references to agent thread instances (MutantTesting.exe) but even while also running other developer applications, had ~40% of CPU capacity still available as shown in Figure 27, which indicates that the 'live' mutation testing process can execute background while developers are coding and performing other tasks. This experiment provided guidance for development of the Visual Studio extension for mutation testing operator selection.


Process Explorer - Sysinternals: www.sysinternals.com
File Options View Process Find DLL Users Help
System Information
Summary CPU Memory I/O GPU
CPU
60.12%

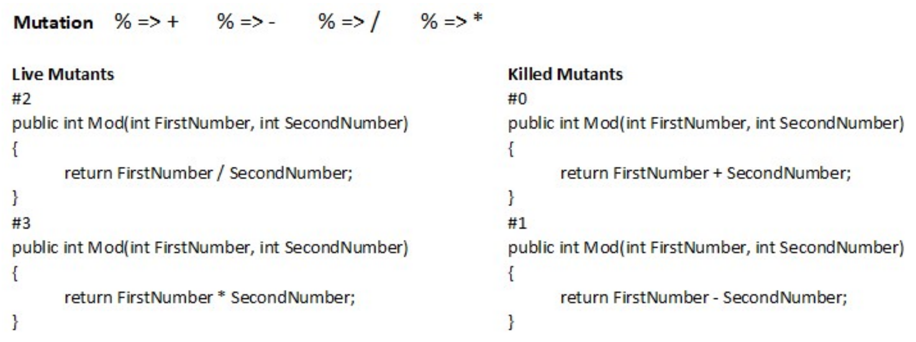
Process	CPU	Private Bytes	Working Set	PID	Description
MutantTesterDRL.exe		25,060 K	40,264 K	16268	MutantTesterDRL
MutantTesting.exe		15,244 K	22,656 K	22460	MutantTest
conhost.exe		7,112 K	16,112 K	13576	Console Window Host
dotnet.exe	< 0.01	18,116 K	36,100 K	21756	.NET Host
conhost.exe	0.02	6,384 K	10,760 K	15172	Console Window Host
dotnet.exe	5.17	37,428 K	61,336 K	12756	.NET Host
dotnet.exe	22.77	33,596 K	55,412 K	17192	.NET Host
dotnet.exe	1.62	1,628 K	4,828 K	17844	.NET Host
MutantTesting.exe	0.10	11,380 K	16,508 K	15956	MutantTest
conhost.exe	0.16	7,116 K	16,176 K	10428	Console Window Host
dotnet.exe	0.95	23,092 K	45,260 K	19304	.NET Host
conhost.exe	< 0.01	6,392 K	10,696 K	8836	Console Window Host
dotnet.exe	11.56	58,176 K	89,904 K	14648	.NET Host

Totals		CPU		Topology	
Handles	106,441	Context Switch Delta	10,830	Cores	2
Threads	2,560	Interrupt Delta	7,438	Sockets	1
Processes	231	DPC Delta	399		

Figure 27. Multiple threads with shared memory and two CPU cores.

Experiment 5: Comparing Mutation Testing Approaches with Four Cores

The purpose of this experiment is to evaluate the impact of parallel deep reinforcement learning selection of mutation operators vs. selection of all or random operators using agents as multiple threads on the mutation testing and operating system performance. The modulo mutants possibly generated are shown in Figure 28.



```
Mutation    % => +      % => -      % => /      % => *

Live Mutants                                              Killed Mutants
#2                                                        #0
public int Mod(int FirstNumber, int SecondNumber)         public int Mod(int FirstNumber, int SecondNumber)
{                                                         {
        return FirstNumber / SecondNumber;                        return FirstNumber + SecondNumber;
}                                                         }
#3                                                        #1
public int Mod(int FirstNumber, int SecondNumber)         public int Mod(int FirstNumber, int SecondNumber)
{                                                         {
        return FirstNumber * SecondNumber;                        return FirstNumber - SecondNumber;
}                                                         }
```

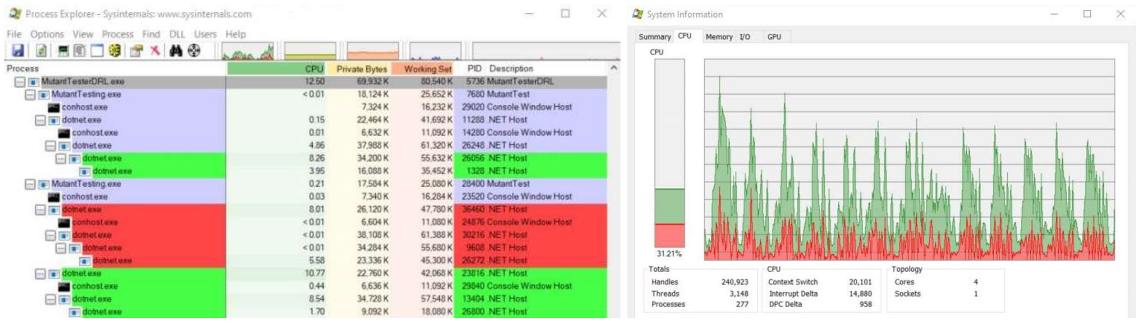
Figure 28. Possible mutants with two live mutants for experiment 5.

The results in Table 10 indicated that the machine learning mutation operator selection process was able to outperform both the traditional approach of selecting all operators, as well as random selection based on the live to total mutant ratio.

Comparison	fail	fail	success
Metrics	all mutation operators	random mutation operators	ML mutation operators
average elapsed time (hh:mm:ss)	12:47:25	10:15:41	10:13:36
average loss	0.069422835	0.126778509	0.143239069
average reward	0.540125000	0.504345667	0.863680540
average mutants total	4	1.7	1.6
average mutants kill	2	1.3	0.3
average mutants live	2	0.4	1.3
average mutation score (kill/total)	0.500	0.765	0.188
live mutant ratio (live : total)	0.500	0.235	0.813
configuration ranking	2	3	1

Table 10. Comparing Mutation Testing Approaches with Four Cores.

An observation depicted in Figure 29, was that the driver thread completed mutation testing in a shorter elapsed time using 4 CPU cores and had ~70% of CPU capacity available for other tasks. This indicates that additional agent threads might be utilized to perform reinforcement learning against more complicated programs.


Process Explorer - Sysinternals: www.sysinternals.com
File Options View Process Find DLL Users Help
System Information
Summary CPU Memory I/O GPU
CPU
31.21%

Process	CPU	Private Bytes	Working Set	PID	Description
MutantTesterDRL.exe	12.50	69,932 K	80,540 K	5736	MutantTesterDRL
MutantTesting.exe	< 0.01	18,124 K	25,652 K	7680	MutantTest
conhost.exe		7,324 K	16,232 K	29020	Console Window Host
dotnet.exe	0.15	22,464 K	41,692 K	11288	.NET Host
conhost.exe	0.01	6,632 K	11,092 K	14280	Console Window Host
dotnet.exe	4.86	37,988 K	61,320 K	26248	.NET Host
dotnet.exe	8.26	34,200 K	55,632 K	26056	.NET Host
dotnet.exe	3.95	16,088 K	35,452 K	1328	.NET Host
MutantTesting.exe	0.21	17,584 K	25,080 K	28400	MutantTest
conhost.exe	0.03	7,340 K	16,284 K	23520	Console Window Host
dotnet.exe	8.01	26,120 K	47,780 K	36460	.NET Host
conhost.exe	< 0.01	6,604 K	11,080 K	24876	Console Window Host
dotnet.exe	< 0.01	38,108 K	61,388 K	30216	.NET Host
dotnet.exe	< 0.01	34,284 K	55,680 K	9608	.NET Host
dotnet.exe	5.58	23,336 K	45,300 K	26272	.NET Host
dotnet.exe	10.77	22,760 K	42,068 K	23816	.NET Host
conhost.exe	0.44	6,636 K	11,092 K	29840	Console Window Host
dotnet.exe	8.54	34,728 K	57,548 K	13404	.NET Host
dotnet.exe	1.70	9,092 K	18,080 K	26800	.NET Host

Totals		CPU		Topology	
Handles	240,923	Context Switch	20,101	Cores	4
Threads	3,148	Interrupt Delta	14,880	Sockets	1
Processes	277	DPC Delta	958		

Figure 29. Comparing Mutation Testing Approaches with Four Cores.

Summary

In summary, all required data was synthesized and the experiments were completed. The results have provided valuable insight towards this dissertation and future research.

Chapter 5 - Conclusion

In conclusion, research regarding mutation testing, mutation selection and machine learning has been conducted but much of it separately and not considering a practical application by software developers using an Integrated Development Environment. Less is available that combines mutation testing, mutation operator selection and reinforcement learning using parallel processing in the Visual Studio IDE for C# development. This dissertation contributes valuable insight and functionality in that area. The results of the experiments demonstrated that the usage of reinforcement learning for mutation operator selection was both effective and practical.

One key contribution from this research was the development of the reinforcement algorithm to identify mutation operator combinations that result in live mutations. This included a criterion to reset the shared experience and restart learning such that the process was able to avoid local minima and always converge on a mutation operator combination recommendation. The policy was consistently successful in minimizing mutation score, with increasing reward and decreasing loss.

With experiments 1 – 3, it was found that the reinforcement learning algorithm was able to identify the correct mutation operator selections for various programs and test suite scenarios, without regard to the number of live mutations. This did not represent every mutation scenario possible with complex programs but does provide evidence for the scenarios evaluated that reinforcement learning was effective by identifying the proper mutation operator combination to detect live mutations and generated 50 – 100% fewer mutations as compared to using all mutation operators.

With experiments 4 and 5, it was determined that by using parallel processing and multiple cores the reinforcement learning process for mutation operator selection was practical. The number of tests (2*1500*5 = 15k) was increased to substantiate the initial experiments results. Additionally, by increasing the number of cores from 2 to 4, there was ~75% more CPU available for other processes to be performed. This combined with tuning the number of concurrent agent threads learning and sharing experience allows for a more complex, realistic codebase to be evaluated for mutation operator selection.

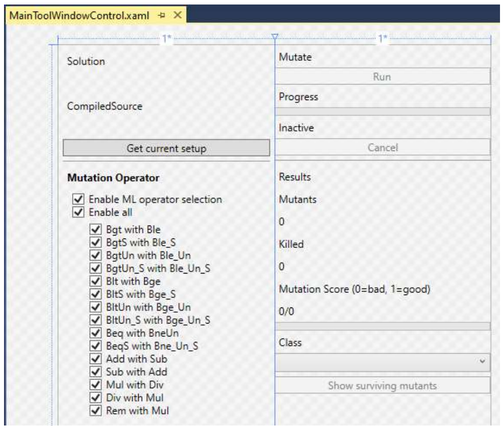
Finally, the required resources for additional research are currently available and growing with the expansion of open-source usage and test-driven development. As shown earlier, there is a need to eliminate software defects from both the software reliance and software development cost perspectives. Given this, the goal of increasing test suite effectiveness using mutation testing and reinforcement learning is possible.

Implications

The implication from the dissertation experiments is that reinforcement learning can be used in the manner required to facilitate mutation operator selection both during software development and deployment. It provides an approach of making mutation testing more viable, which is already considered the most accurate and dependable approach for assessing test suite effectiveness (Strug & Strug, 2012).

Recommendations

Based on experimentation results, the recommendation is to pursue research on improving the machine learning hyper-parameters, incorporating additional machine learning features for training against more complicated programs and development required to implement this paper's reinforcement learning approaches for mutation operator selection as a Visual Studio extension. Transitioning from agents navigating a simple program environment to a more complex, multi-module codebase. To further this recommendation the following design extends the implementation to integrate reinforcement learning within the development and testing environment (IDE).


MainToolWindowControl.xaml

Solution

CompiledSource

Get current setup

Mutation Operator

- ✓ Enable ML operator selection
- ✓ Enable all
  - ✓ Bgt with Ble
  - ✓ BgtS with Ble_S
  - ✓ BgtUn with Ble_Un
  - ✓ BgtUn_S with Ble_Un_S
  - ✓ Blt with Bge
  - ✓ BltS with Bge_S
  - ✓ BltUn with Bge_Un
  - ✓ BltUn_S with Bge_Un_S
  - ✓ Beq with BneUn
  - ✓ BeqS with Bne_Un_S
  - ✓ Add with Sub
  - ✓ Sub with Add
  - ✓ Mul with Div
  - ✓ Div with Mul
  - ✓ Rem with Mul

Mutate

Run

Progress

Inactive

Cancel

Results

Mutants

0

Killed

0

Mutation Score (0=bad, 1=good)

0/0

Class

Show surviving mutants

Figure 30. Mutation Testing with Reinforcement Learning in Visual Studio extension.

MainToolWindow

The interface would allow machine learning feedback to developers on mutation operator selection based on agent traversal through the codebase. Forward propagation using input based on proximity to the agent's current code piece CIL instruction location to adjacent CIL instructions in the library. Based on (Microsoft Corporation, 2020), the CIL instruction set contains 235 possible instructions, so each could have corresponding mutations. Once encoded, the input values fed through the network determine an action, which would correspond to instruction replacements, thus generating a mutant library. The mutant software library would be tested, the mutation score calculated and used as a reward for mutation operator suggestions against the entire codebase.

Appendices

Appendix A – Detailed Experiment Results

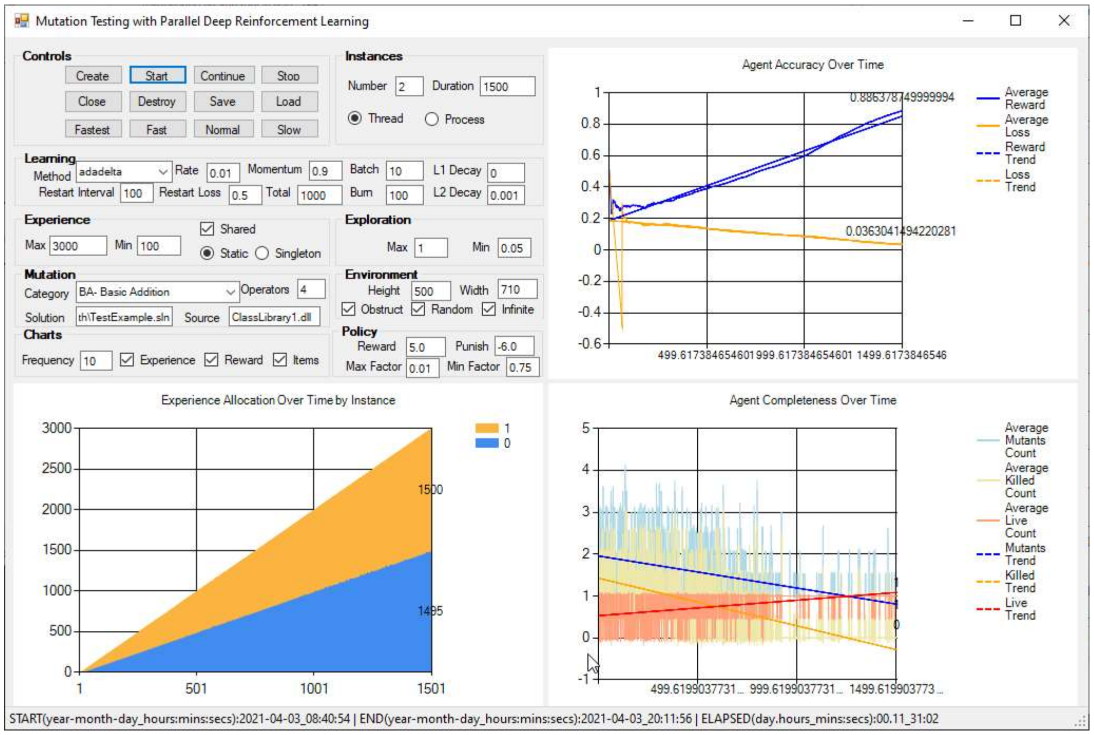
This appendix provides detailed results of experiments 1 through 5. As previously mentioned, each experiment carried out in this study were conducted two developer machines. The first with 2 physical Intel ® Core® CPU @2.50GHz processors (4 logical processors), second with 4 logical Intel ® Xeon® CPU @3.00GHz processors (8 logical processors), both with 16GB memory (L1 cache:256KB, L2 cache:1MB, L3 cache:8MB) and Windows 10 64-bit operating system. The experiments were performed while also running other developer applications, including Visual Studio, Microsoft Outlook, Microsoft Word, Microsoft Excel, Microsoft Teams and Chrome Internet Browser. As part of the experiment, the reinforcement learning agent configurations were tested and evaluated, using the following metrics:

1. Loss, 2. Reward, 3. Elapsed time, 4. Mutation score, 5. CPU percentage.

Below are screenshots with a summary of each experiment's agent hyperparameters, architecture and detailed accuracy results, corresponding to the above evaluation method. The code, program usage, agent files and screenshots are also included in the Git repo available at https://github.com/mstewart1972/ParallelDeepReinforcementLearning.

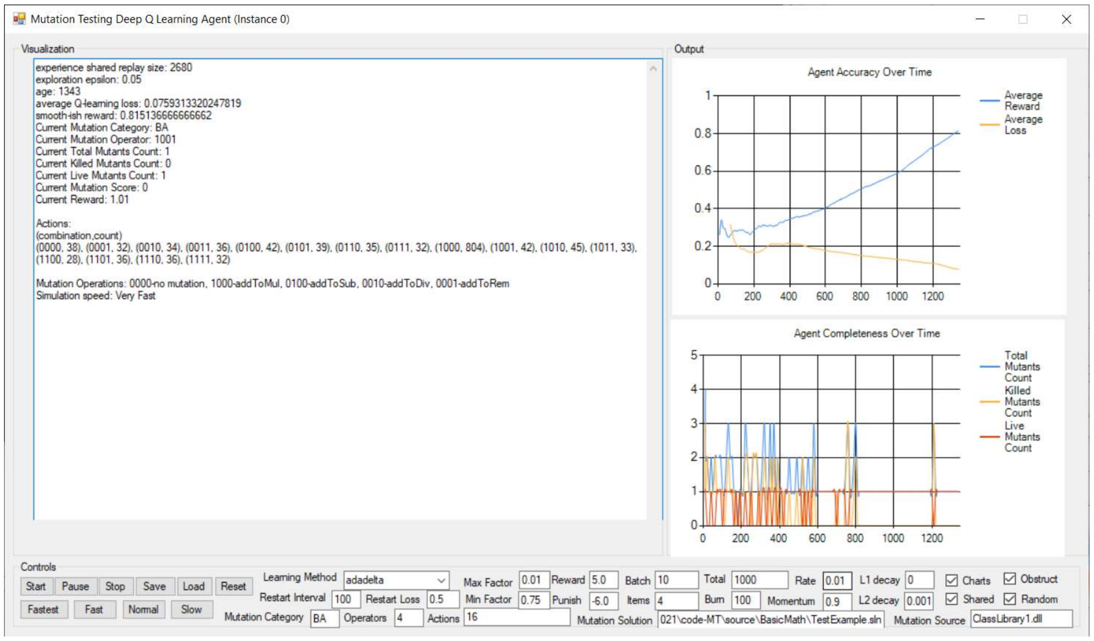
Experiment 1: Learning Mutation Testing with One Live Mutation

Machine Learning selection of mutation operators:



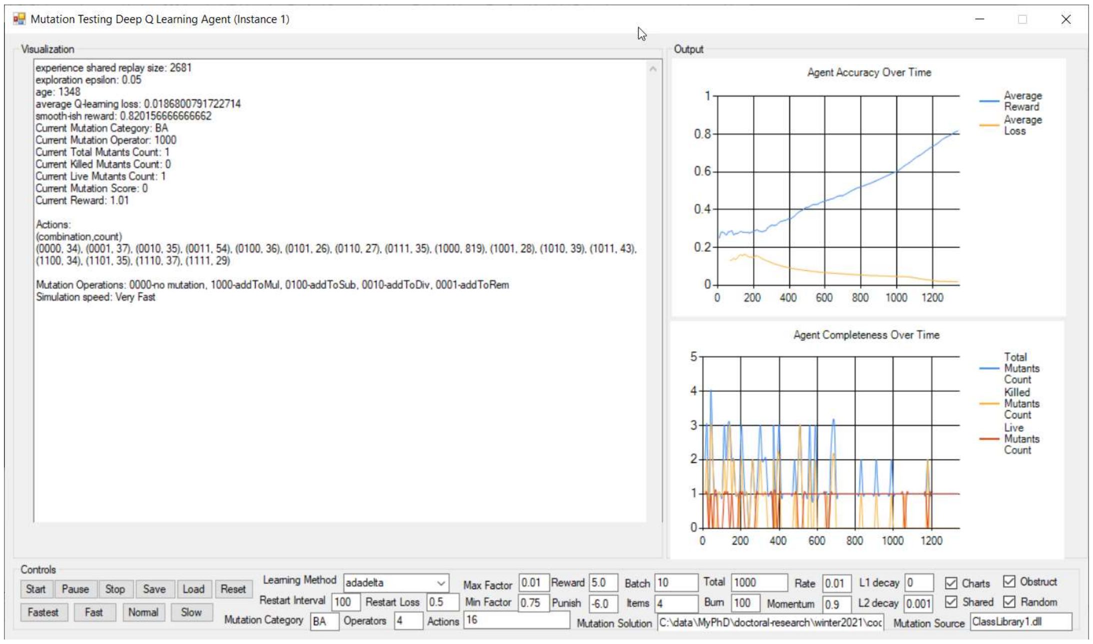
```
Instances
Number 2   Duration 1500
(●) Thread  ( ) Process
Learning
Method adadelta   Rate 0.01   Momentum 0.9   Batch 10    L1 Decay 0
Restart Interval 100   Restart Loss 0.5   Total 1000   Burn 100   L2 Decay 0.001
Experience
Max 3000   Min 100   [✓] Shared   (●) Static  ( ) Singleton
Exploration
Max 1   Min 0.05
Mutation
Category BA- Basic Addition   Operators 4
Solution th\TestExample.sln   Source ClassLibrary1.dll
Environment
Height 500   Width 710
[✓] Obstruct  [✓] Random  [✓] Infinite
Charts
Frequency 10   [✓] Experience  [✓] Reward  [✓] Items
Policy
Reward 5.0   Punish -6.0
Max Factor 0.01   Min Factor 0.75
```

```
START(year-month-day_hours:mins:secs):2021-04-03_08:40:54 | END(year-month-day_hours:mins:secs):2021-04-03_20:11:56 | ELAPSED(day.hours_mins:secs):00.11_31:02
```



```
experience shared replay size: 2680
exploration epsilon: 0.05
age: 1343
average Q-learning loss: 0.0759313320247819
smooth-ish reward: 0.815136666666662
Current Mutation Category: BA
Current Mutation Operator: 1001
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 38), (0001, 32), (0010, 34), (0011, 36), (0100, 42), (0101, 39), (0110, 35), (0111, 32), (1000, 804), (1001, 42), (1010, 45), (1011, 33),
(1100, 28), (1101, 36), (1110, 36), (1111, 32)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

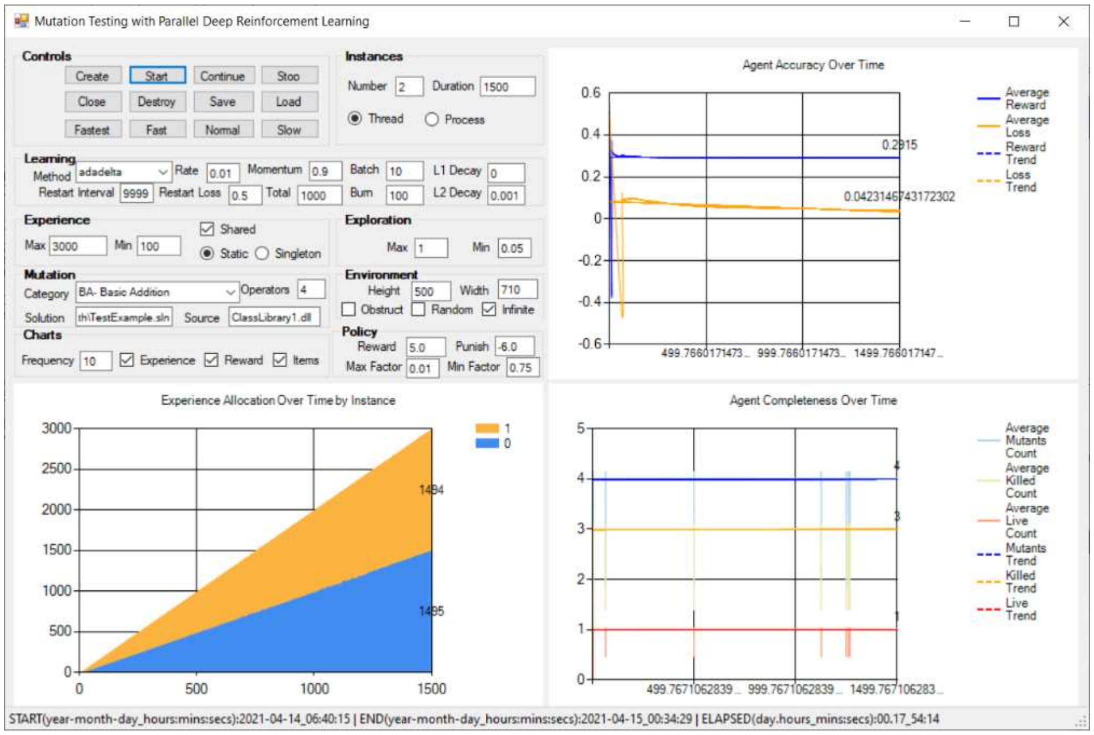
```
Learning Method adadelta   Max Factor 0.01   Reward 5.0   Batch 10   Total 1000   Rate 0.01   L1 decay 0   [✓] Charts   [✓] Obstruct
Restart Interval 100   Restart Loss 0.5   Min Factor 0.75   Punish -6.0   Items 4   Burn 100   Momentum 0.9   L2 decay 0.001   [✓] Shared   [✓] Random
Mutation Category BA   Operators 4   Actions 16   Mutation Solution 021\code-MT\source\BasicMath\TestExample.sln   Mutation Source ClassLibrary1.dll
```



```
experience shared replay size: 2681
exploration epsilon: 0.05
age: 1348
average Q-learning loss: 0.0186800791722714
smooth-ish reward: 0.820156666666662
Current Mutation Category: BA
Current Mutation Operator: 1000
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 34), (0001, 37), (0010, 35), (0011, 54), (0100, 36), (0101, 26), (0110, 27), (0111, 35), (1000, 819), (1001, 28), (1010, 39), (1011, 43),
(1100, 34), (1101, 35), (1110, 37), (1111, 29)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Mutation Category: BA; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☑ Charts; ☑ Shared; ☑ Obstruct; ☑ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

Selection of all mutation operators:



Instances: Number 2; Duration 1500; (● Thread) ○ Process

Learning: Method adadelta; Rate 0.01; Momentum 0.9; Batch 10; L1 Decay 0; Restart Interval 9999; Restart Loss 0.5; Total 1000; Burn 100; L2 Decay 0.001

Experience: Max 3000; Min 100; ☑ Shared; (● Static) ○ Singleton

Exploration: Max 1; Min 0.05

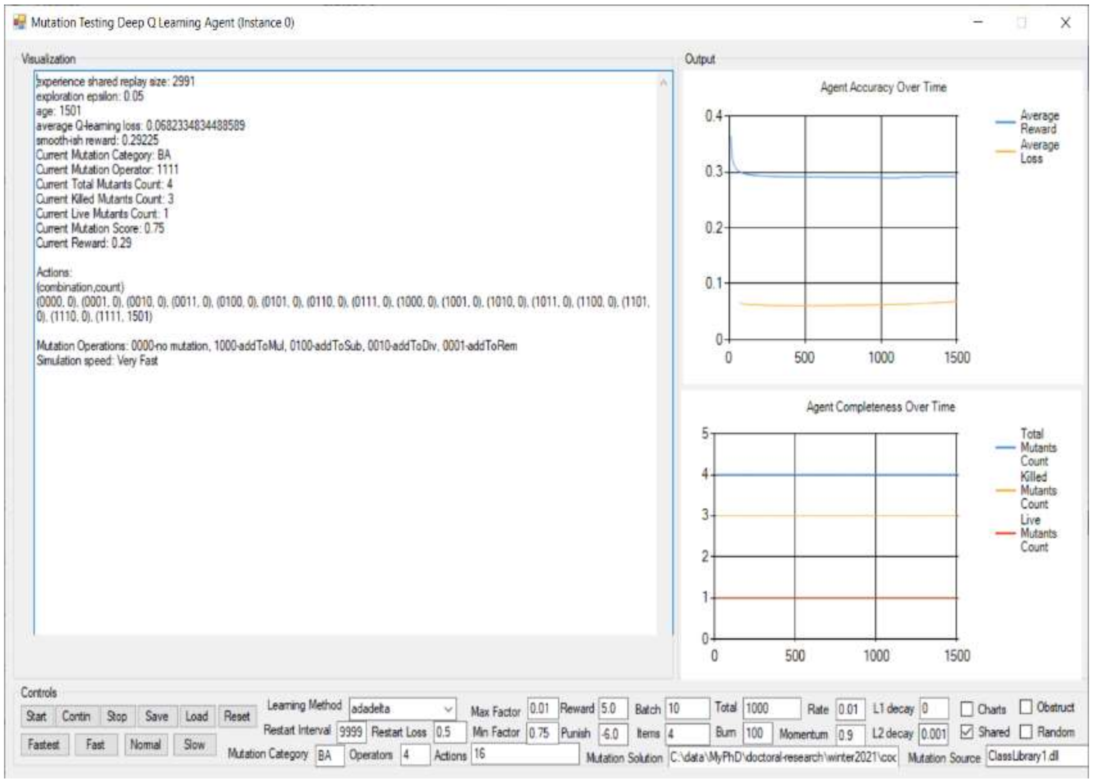
Mutation: Category BA- Basic Addition; Operators 4; Solution th\TestExample.sln; Source ClassLibrary1.dll

Environment: Height 500; Width 710; ☐ Obstruct; ☐ Random; ☑ Infinite

Charts: Frequency 10; ☑ Experience; ☑ Reward; ☑ Items

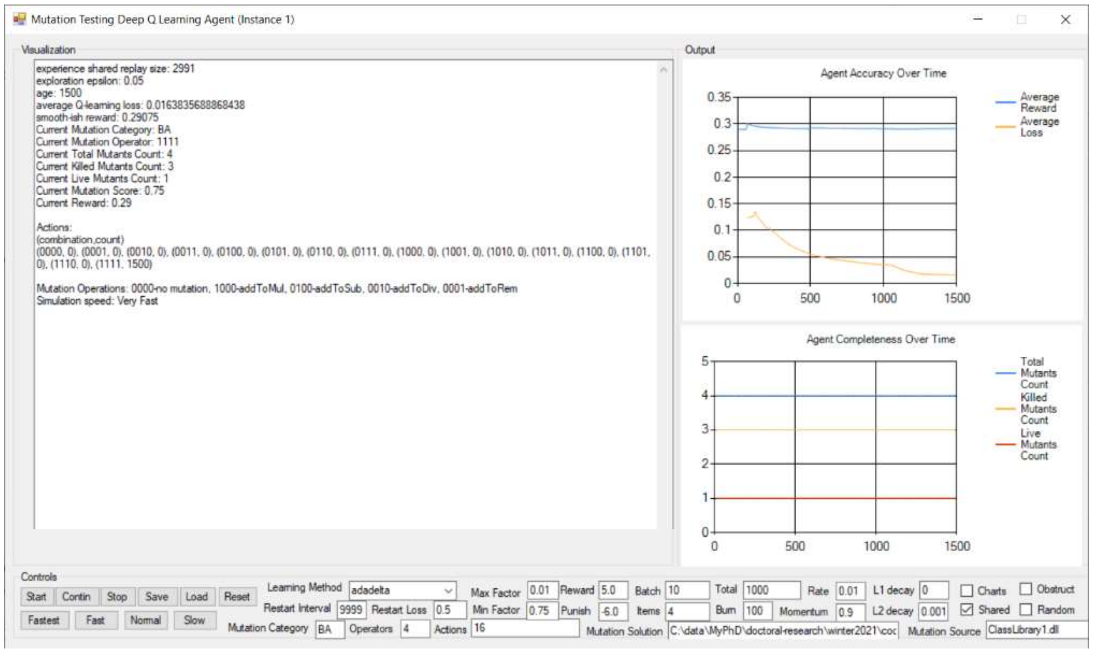
Policy: Reward 5.0; Punish -6.0; Max Factor 0.01; Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-04-14_06:40:15 | END(year-month-day_hours:mins:secs):2021-04-15_00:34:29 | ELAPSED(day.hours_mins:secs):00.17_54:14



```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0682334834488589
smooth-ish reward: 0.29225
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 3
Current Live Mutants Count: 1
Current Mutation Score: 0.75
Current Reward: 0.29

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 9999; Restart Loss: 0.5; Mutation Category: BA; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

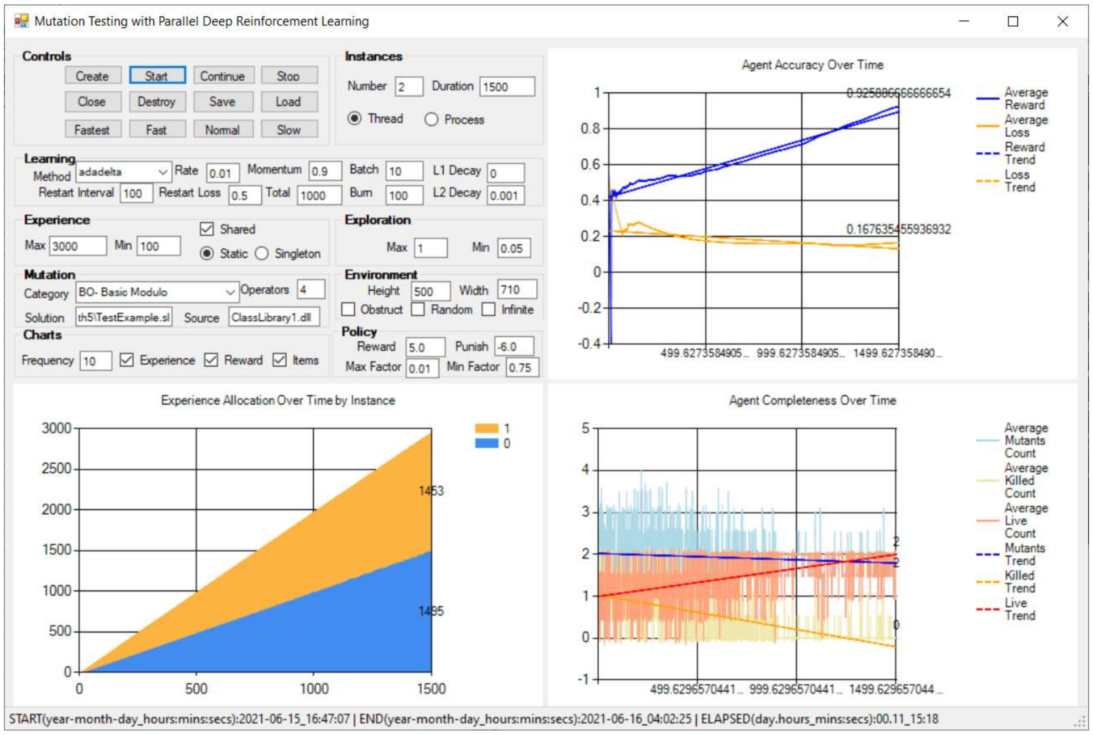


```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1500
average Q-learning loss: 0.0163835688868438
smooth-ish reward: 0.29075
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 3
Current Live Mutants Count: 1
Current Mutation Score: 0.75
Current Reward: 0.29

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1500)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 9999; Restart Loss: 0.5; Mutation Category: BA; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

Experiment 2: Learning Mutation Testing with Multiple Live Mutations

Machine Learning selection of mutation operators:



Instances: Number 2, Duration 1500; [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0; Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100; [✓ Shared]; [✓ Static] Singleton

Exploration: Max 1, Min 0.05

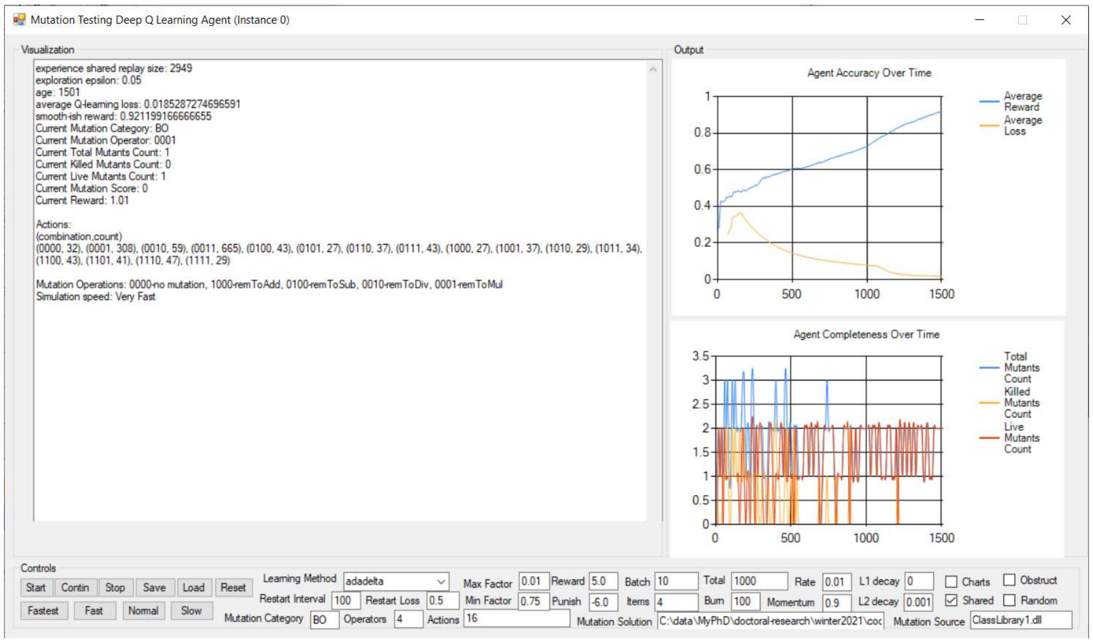
Mutation: Category BO- Basic Modulo, Operators 4; Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710; ☐ Obstruct ☐ Random ☐ Infinite

Charts: Frequency 10; ☑ Experience ☑ Reward ☑ Items

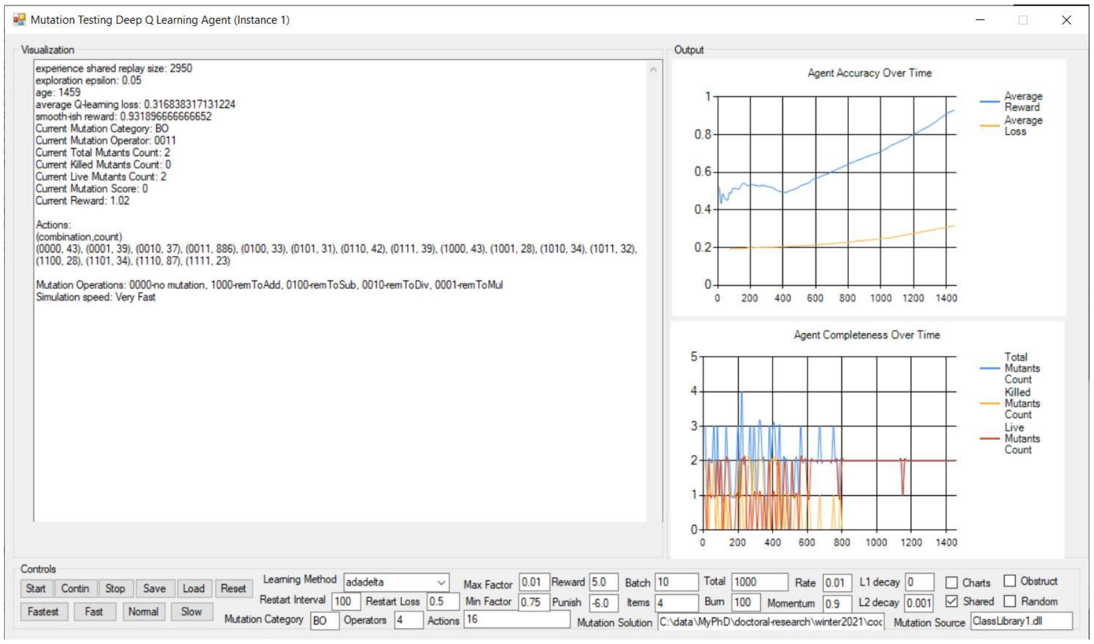
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-15_16:47:07 | END(year-month-day_hours:mins:secs):2021-06-16_04:02:25 | ELAPSED(day.hours_mins:secs):00.11_15:18



```
experience shared replay size: 2949
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0185287274696591
smooth-ish reward: 0.921199166666655
Current Mutation Category: BO
Current Mutation Operator: 0001
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 32), (0001, 308), (0010, 59), (0011, 665), (0100, 43), (0101, 27), (0110, 37), (0111, 43), (1000, 27), (1001, 37), (1010, 29), (1011, 34),
(1100, 43), (1101, 41), (1110, 47), (1111, 29)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

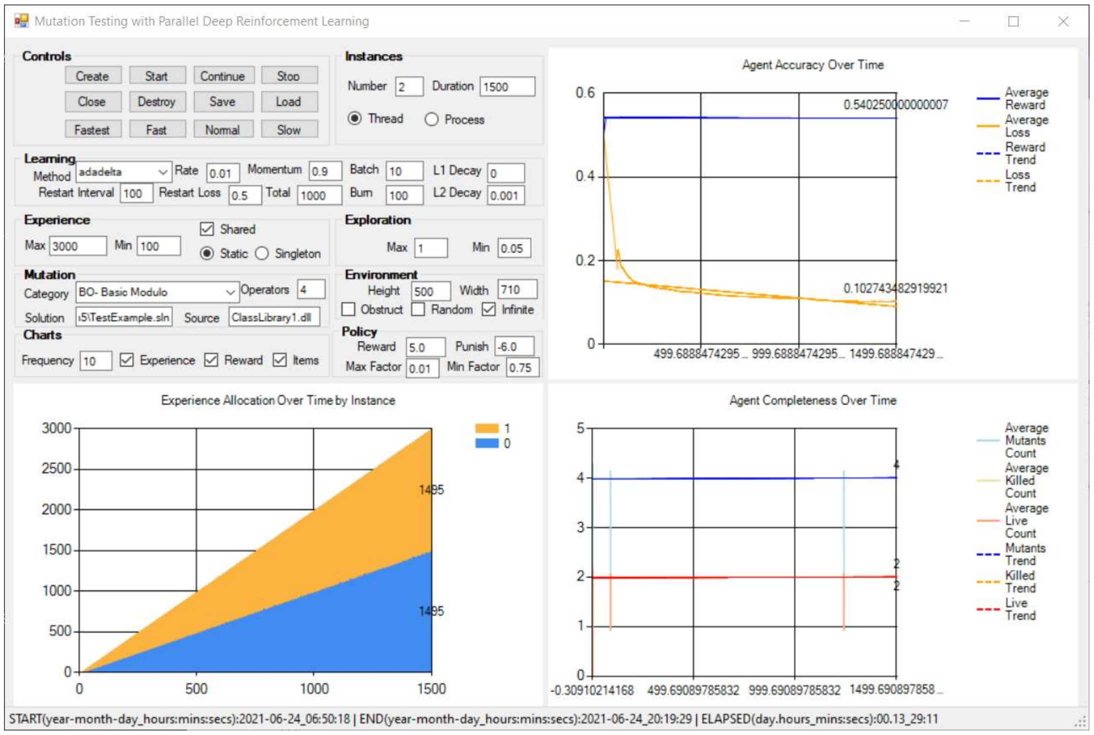
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, ☐ Charts, ☐ Obstruct; Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, ☑ Shared, ☐ Random; Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2950
exploration epsilon: 0.05
age: 1459
average Q-learning loss: 0.316838317131224
smooth-ish reward: 0.931896666666652
Current Mutation Category: BO
Current Mutation Operator: 0011
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 43), (0001, 39), (0010, 37), (0011, 886), (0100, 33), (0101, 31), (0110, 42), (0111, 39), (1000, 43), (1001, 28), (1010, 34), (1011, 32),
(1100, 28), (1101, 34), (1110, 87), (1111, 23)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Mutation Category: BO; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

Selection of all mutation operators:



Instances: Number 2; Duration 1500; [✓ Thread] Process

Learning: Method adadelta; Rate 0.01; Momentum 0.9; Batch 10; L1 Decay 0; Restart Interval 100; Restart Loss 0.5; Total 1000; Burn 100; L2 Decay 0.001

Experience: Max 3000; Min 100; ☑ Shared; [✓ Static] Singleton

Exploration: Max 1; Min 0.05

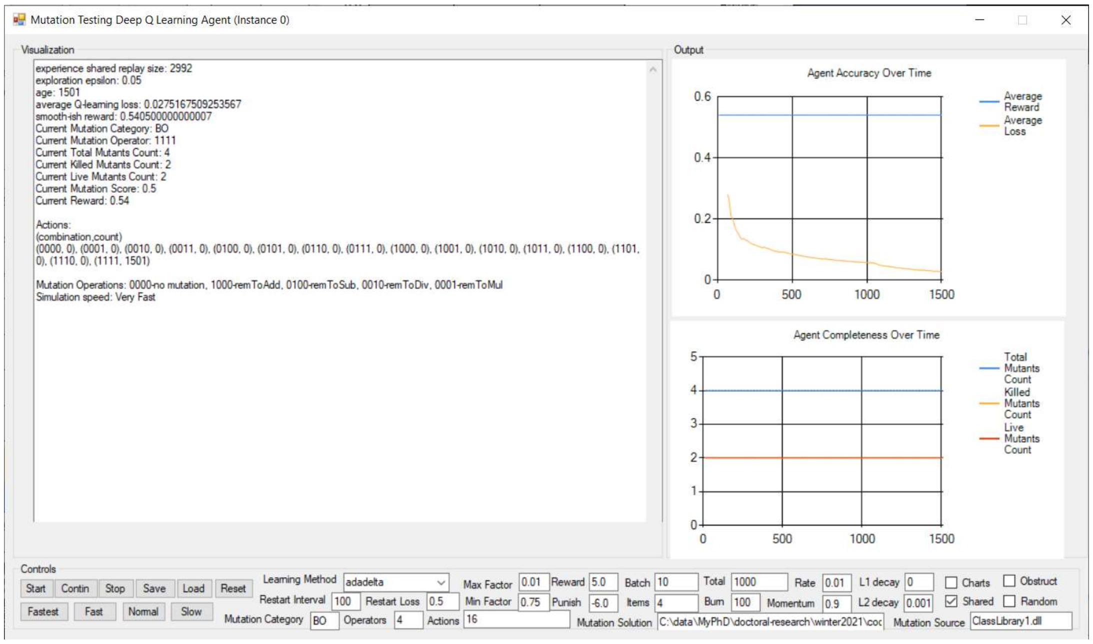
Mutation: Category BO- Basic Modulo; Operators 4; Solution \5\TestExample.sln; Source ClassLibrary1.dll

Environment: Height 500; Width 710; ☐ Obstruct; ☐ Random; ☑ Infinite

Charts: Frequency 10; ☑ Experience; ☑ Reward; ☑ Items

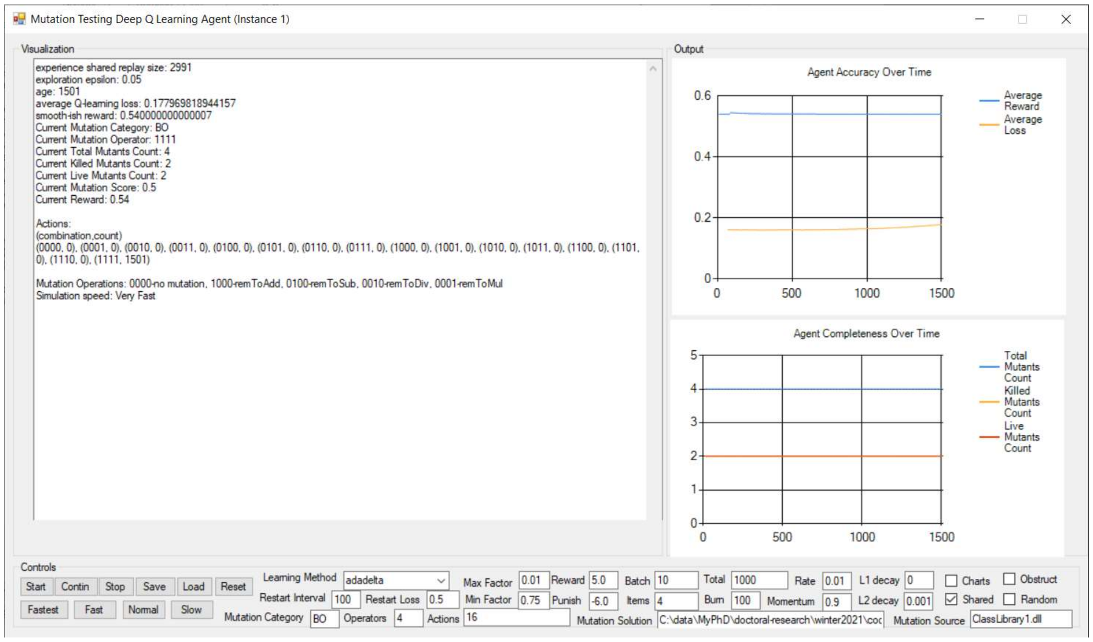
Policy: Reward 5.0; Punish -6.0; Max Factor 0.01; Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-24_06:50:18 | END(year-month-day_hours:mins:secs):2021-06-24_20:19:29 | ELAPSED(day.hours_mins:secs):00.13_29:11



```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0275167509253567
smooth-ish reward: 0.540500000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Mutation Category: BO; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

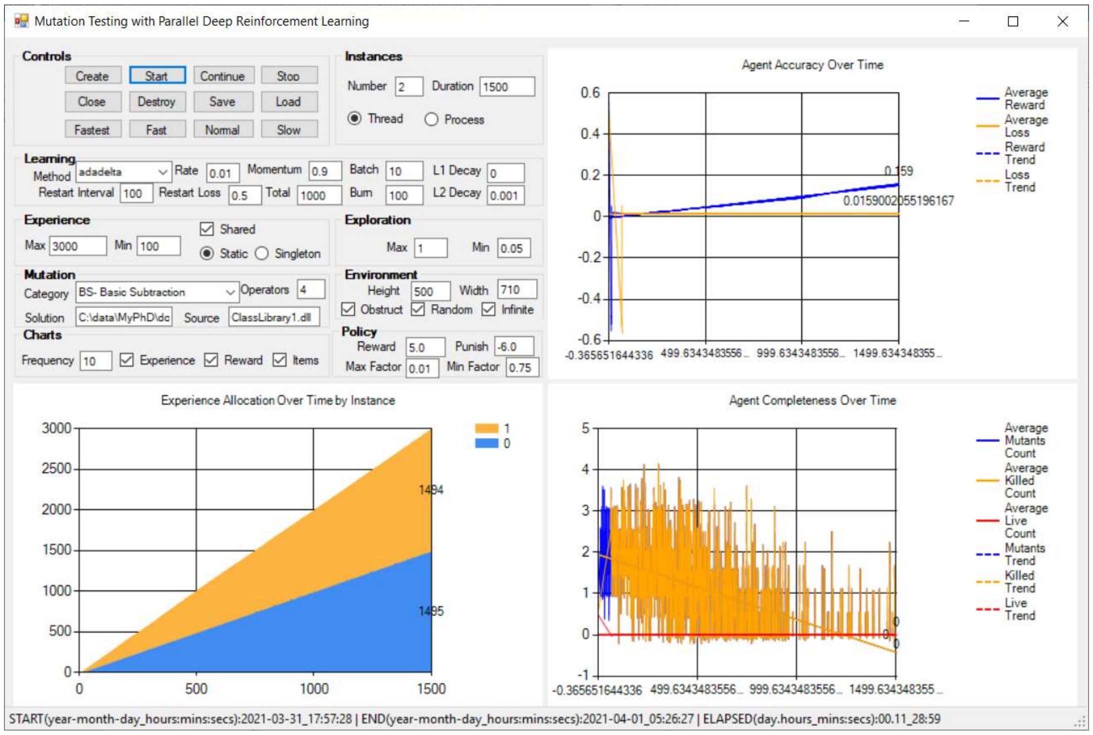


```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.177969818944157
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

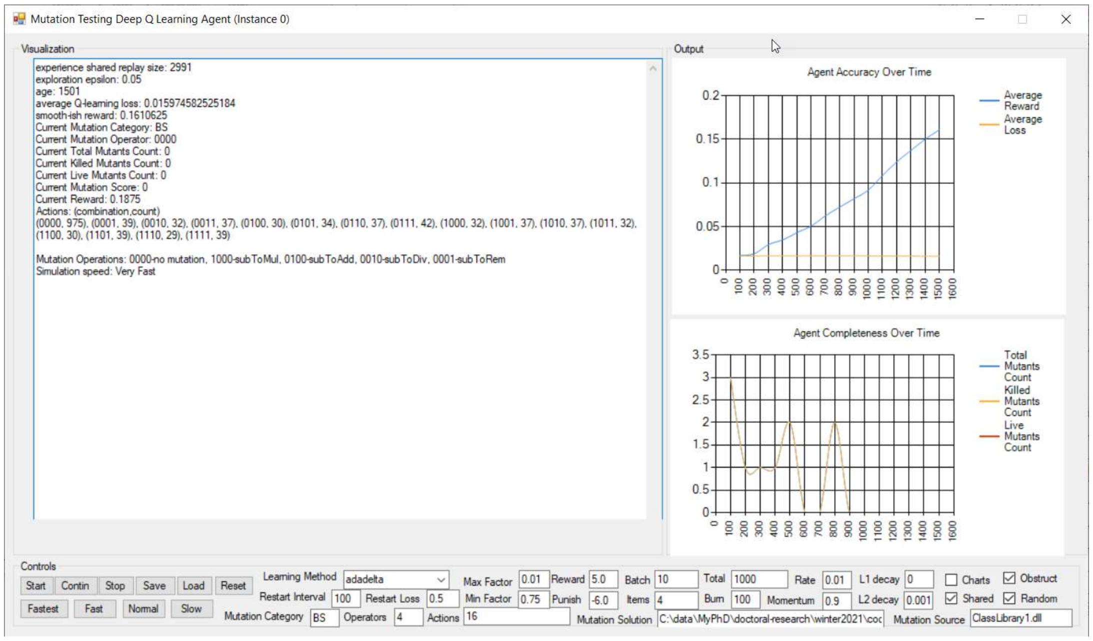
Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Mutation Category: BO; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

Experiment 3: Learning Mutation Testing with No Live Mutations

Machine Learning selection of mutation operators:


Mutation Testing with Parallel Deep Reinforcement Learning

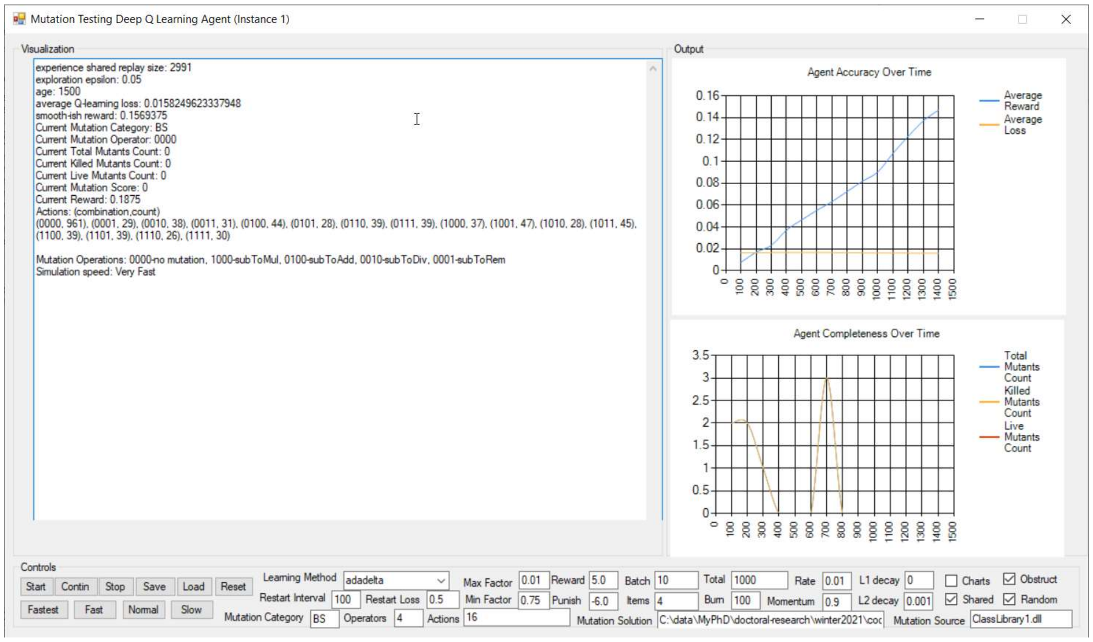
```
Controls: Create  Start  Continue  Stop  Close  Destroy  Save  Load  Fastest  Fast  Normal  Slow
Instances: Number 2   Duration 1500   (●) Thread  ( ) Process
Learning: Method adadelta   Rate 0.01   Momentum 0.9   Batch 10   L1 Decay 0
          Restart Interval 100   Restart Loss 0.5   Total 1000   Burn 100   L2 Decay 0.001
Experience: Max 3000   Min 100   [✓] Shared   (●) Static  ( ) Singleton
Exploration: Max 1   Min 0.05
Mutation: Category BS- Basic Subtraction   Operators 4
          Solution C:\data\MyPhD\dc   Source ClassLibrary1.dll
Environment: Height 500   Width 710   [✓] Obstruct  [✓] Random  [✓] Infinite
Charts: Frequency 10   [✓] Experience  [✓] Reward  [✓] Items
Policy: Reward 5.0   Punish -6.0   Max Factor 0.01   Min Factor 0.75
START(year-month-day_hours:mins:secs):2021-03-31_17:57:28 | END(year-month-day_hours:mins:secs):2021-04-01_05:26:27 | ELAPSED(day.hours_mins:secs):00.11_28:59
```


Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.015974582525184
smooth-ish reward: 0.1610625
Current Mutation Category: BS
Current Mutation Operator: 0000
Current Total Mutants Count: 0
Current Killed Mutants Count: 0
Current Live Mutants Count: 0
Current Mutation Score: 0
Current Reward: 0.1875
Actions: (combination,count)
(0000, 975), (0001, 39), (0010, 32), (0011, 37), (0100, 30), (0101, 34), (0110, 37), (0111, 42), (1000, 32), (1001, 37), (1010, 37), (1011, 32),
(1100, 30), (1101, 39), (1110, 29), (1111, 39)

Mutation Operations: 0000-no mutation, 1000-subToMul, 0100-subToAdd, 0010-subToDiv, 0001-subToRem
Simulation speed: Very Fast
```

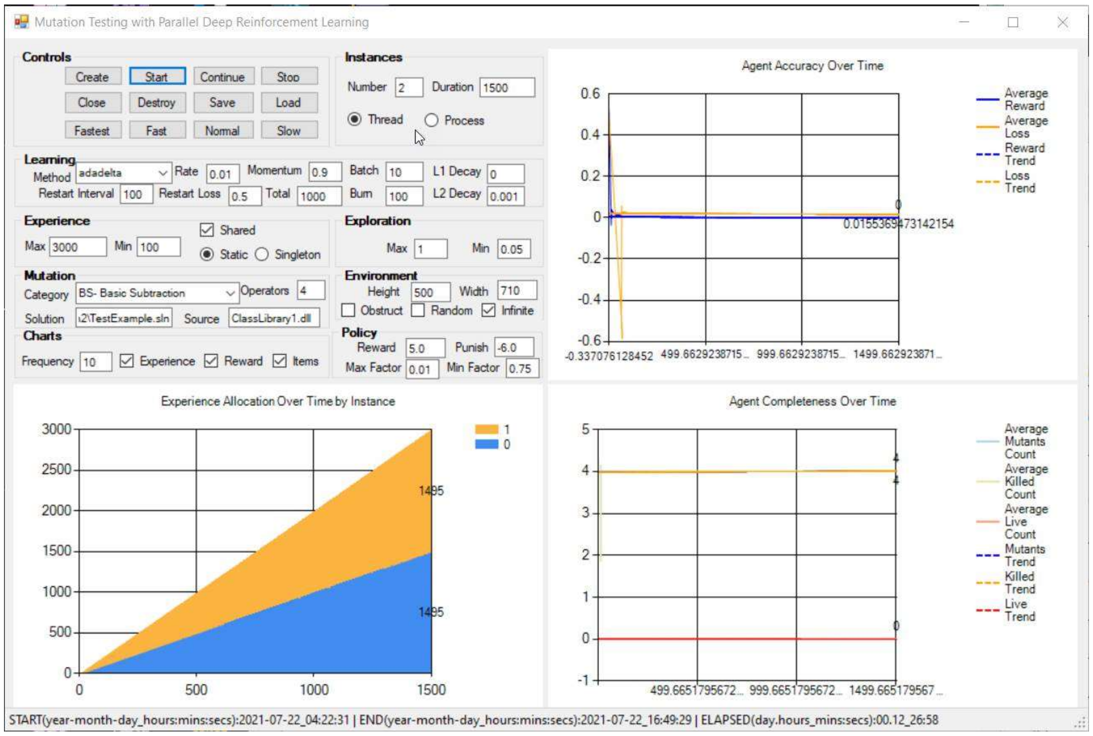
```
Controls: Start  Contin  Stop  Save  Load  Reset  Fastest  Fast  Normal  Slow
Learning Method adadelta   Max Factor 0.01   Reward 5.0   Batch 10   Total 1000   Rate 0.01   L1 decay 0     [ ] Charts  [✓] Obstruct
Restart Interval 100   Restart Loss 0.5   Min Factor 0.75   Punish -6.0   Items 4   Burn 100   Momentum 0.9   L2 decay 0.001   [✓] Shared  [✓] Random
Mutation Category BS   Operators 4   Actions 16   Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc   Mutation Source ClassLibrary1.dll
```



```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1500
average Q-learning loss: 0.0158249623337948
smooth-ish reward: 0.1569375
Current Mutation Category: BS
Current Mutation Operator: 0000
Current Total Mutants Count: 0
Current Killed Mutants Count: 0
Current Live Mutants Count: 0
Current Mutation Score: 0
Current Reward: 0.1875
Actions: (combination,count)
(0000, 961), (0001, 29), (0010, 38), (0011, 31), (0100, 44), (0101, 28), (0110, 39), (0111, 39), (1000, 37), (1001, 47), (1010, 28), (1011, 45),
(1100, 39), (1101, 39), (1110, 26), (1111, 30)

Mutation Operations: 0000-no mutation, 1000-subToMul, 0100-subToAdd, 0010-subToDiv, 0001-subToRem
Simulation speed: Very Fast
```

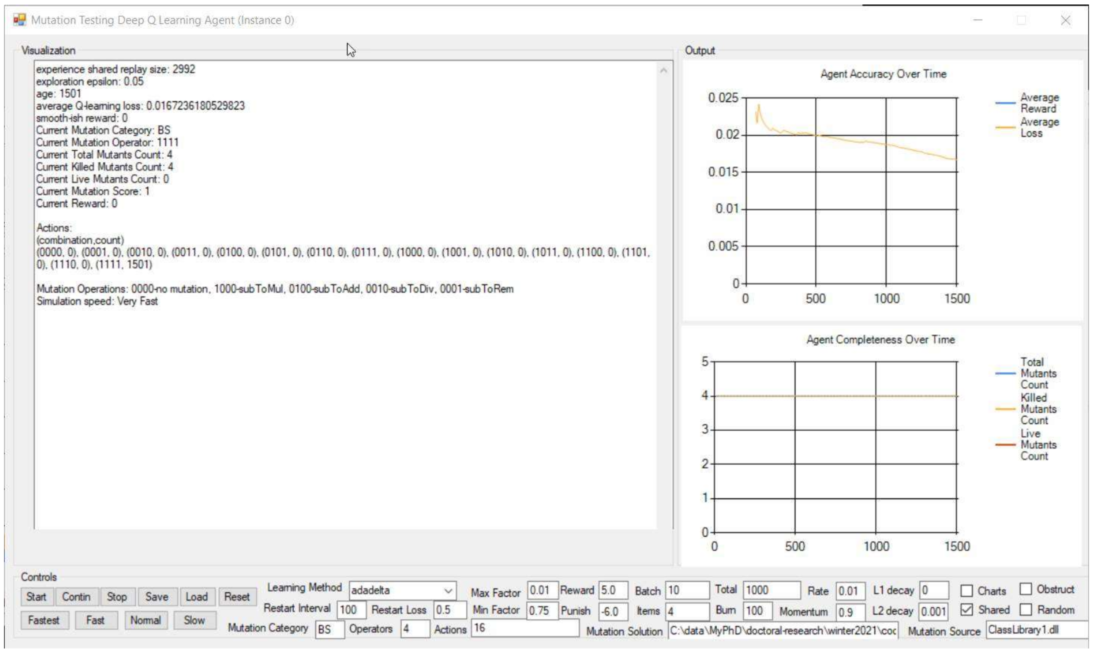
Learning Method: adadelta; Max Factor: 0.01; Reward: 5.0; Batch: 10; Total: 1000; Rate: 0.01; L1 decay: 0; [ ] Charts; [✓] Obstruct
Restart Interval: 100; Restart Loss: 0.5; Min Factor: 0.75; Punish: -6.0; Items: 4; Burn: 100; Momentum: 0.9; L2 decay: 0.001; [✓] Shared; [✓] Random
Mutation Category: BS; Operators: 4; Actions: 16; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

Selection of all mutation operators:



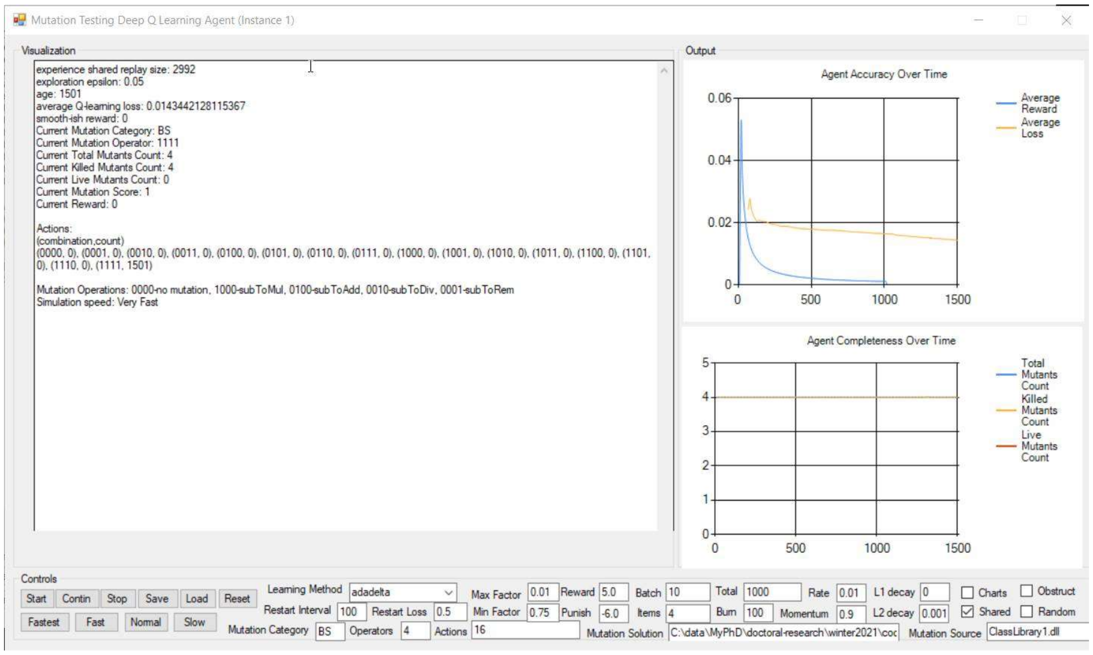
Instances — Number: 2; Duration: 1500; [✓ Thread] Process
Learning — Method: adadelta; Rate: 0.01; Momentum: 0.9; Batch: 10; L1 Decay: 0; Restart Interval: 100; Restart Loss: 0.5; Total: 1000; Burn: 100; L2 Decay: 0.001
Experience — Max: 3000; Min: 100; [✓] Shared; [✓ Static] Singleton
Exploration — Max: 1; Min: 0.05
Mutation — Category: BS- Basic Subtraction; Operators: 4; Solution: i2\TestExample.sln; Source: ClassLibrary1.dll
Environment — Height: 500; Width: 710; [ ] Obstruct; [ ] Random; [✓] Infinite
Charts — Frequency: 10; [✓] Experience; [✓] Reward; [✓] Items
Policy — Reward: 5.0; Punish: -6.0; Max Factor: 0.01; Min Factor: 0.75

START(year-month-day_hours:mins:secs):2021-07-22_04:22:31 | END(year-month-day_hours:mins:secs):2021-07-22_16:49:29 | ELAPSED(day.hours_mins:secs):00.12_26:58



```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0167236180529823
smooth-ish reward: 0
Current Mutation Category: BS
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 4
Current Live Mutants Count: 0
Current Mutation Score: 1
Current Reward: 0

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-subToMul, 0100-subToAdd, 0010-subToDiv, 0001-subToRem
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Mutation Category: BS; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll



```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0143442128115367
smooth-ish reward: 0
Current Mutation Category: BS
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 4
Current Live Mutants Count: 0
Current Mutation Score: 1
Current Reward: 0

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-subToMul, 0100-subToAdd, 0010-subToDiv, 0001-subToRem
Simulation speed: Very Fast
```

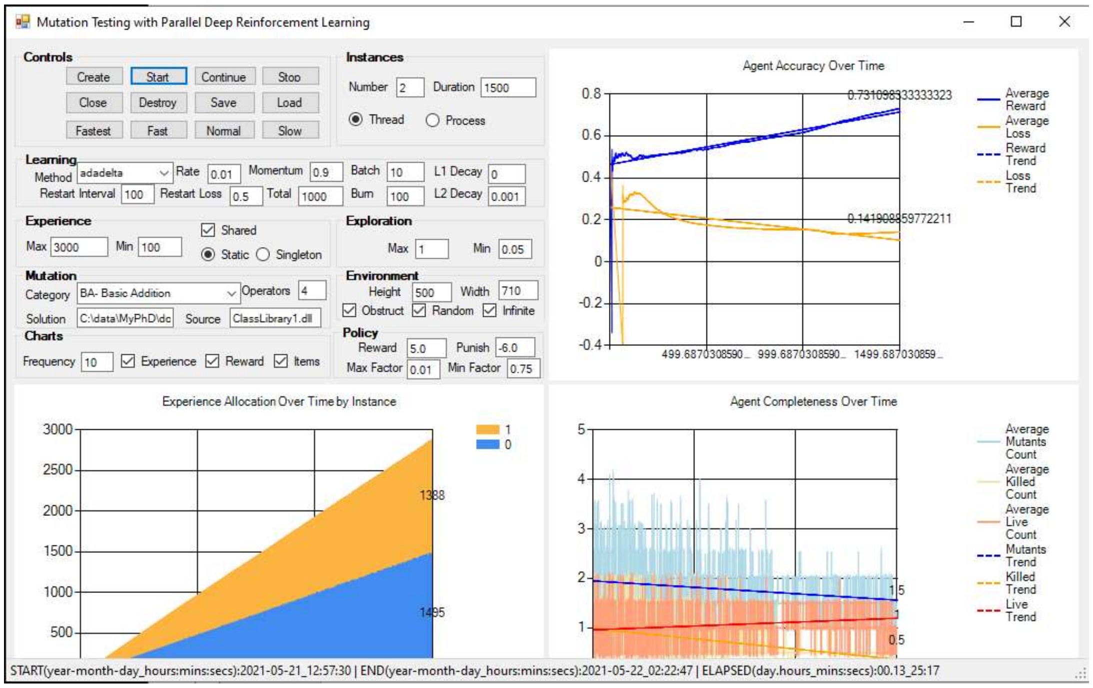
Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Mutation Category: BS; Operators: 4; Actions: 16; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; ☐ Charts; ☑ Shared; ☐ Obstruct; ☐ Random; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

Experiment 4: Comparing Mutation Testing Approaches with Two Cores

Machine Learning selection of mutation operators:

Results						
Test Metrics	**Test Average**	**Test Results**				
test runs		1	2	3	4	5
avg elapsed time (hh:mm:s	13:52:11	13:25:17	16:41:21	13:37:52	12:27:27	13:08:59
maximum action	1100	1000	1110	1100	0000	1000
average mutation score	0.21					
average mutant total	1.400000000000000	1.50	2.50	1.00	1.00	1.00
average killed count	0.300000000000000	0.50	1.00	0.00	0.00	0.00
average live count	1.100000000000000	1.00	1.50	1.00	1.00	1.00
average Q-learn loss	0.152961443509268	0.14190886	0.23263098	0.16998208	0.18783096	0.03245434
smooth-ish reward	0.798272916666644	0.73109833	0.68447875	0.82420500	0.82368542	0.92789708
Test Metrics	**Test Average**	**Test Results**				
test runs - instance0		1	2	3	4	5
maximum action	1100	1000	1110	1100	0100	1000
average Q-learn loss	0.129909970188984	0.02063952	0.23266643	0.02208906	0.33723121	0.03692364
smooth-ish reward	0.877912166666663	0.93731750	0.68755750	0.92718417	0.91432917	0.92317250
test runs - instance1		1	2	3	4	5
maximum action	1100	1010	1110	1100	1000	1100
average Q-learn loss	0.176076553121387	0.26340732	0.23251918	0.31798466	0.03871289	0.02775872
smooth-ish reward	0.759268833333329	0.52539917	0.68189000	0.72255917	0.93304167	0.93345417
	Test Summary	1	2	3	4	5
maximum action	**1100**					
average Q-learn loss	**0.152993261655185**	0.14202342	0.2325928	0.17003686	0.18797205	0.03234118
smooth-ish reward	**0.818590499999996**	0.73135833	0.68472375	0.82487167	0.92368542	0.92831333

1


Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, (●) Thread ( ) Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, (●) Static ( ) Singleton

Exploration: Max 1, Min 0.05

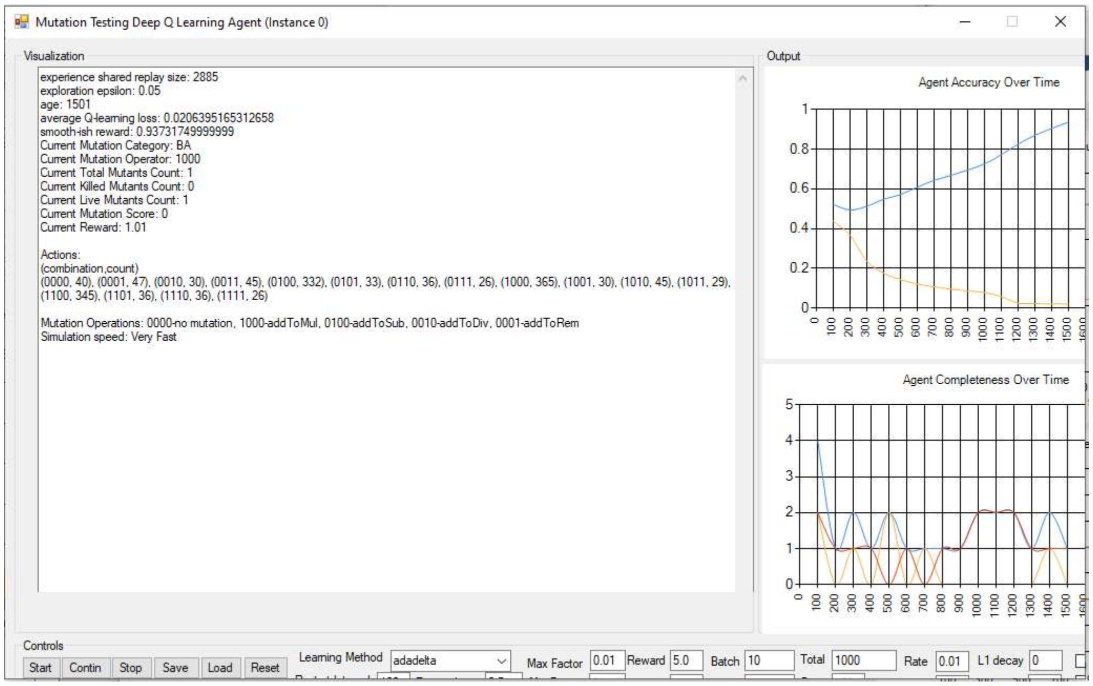
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☑ Obstruct ☑ Random ☑ Infinite

Charts: Frequency 10, ☑ Experience ☑ Reward ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

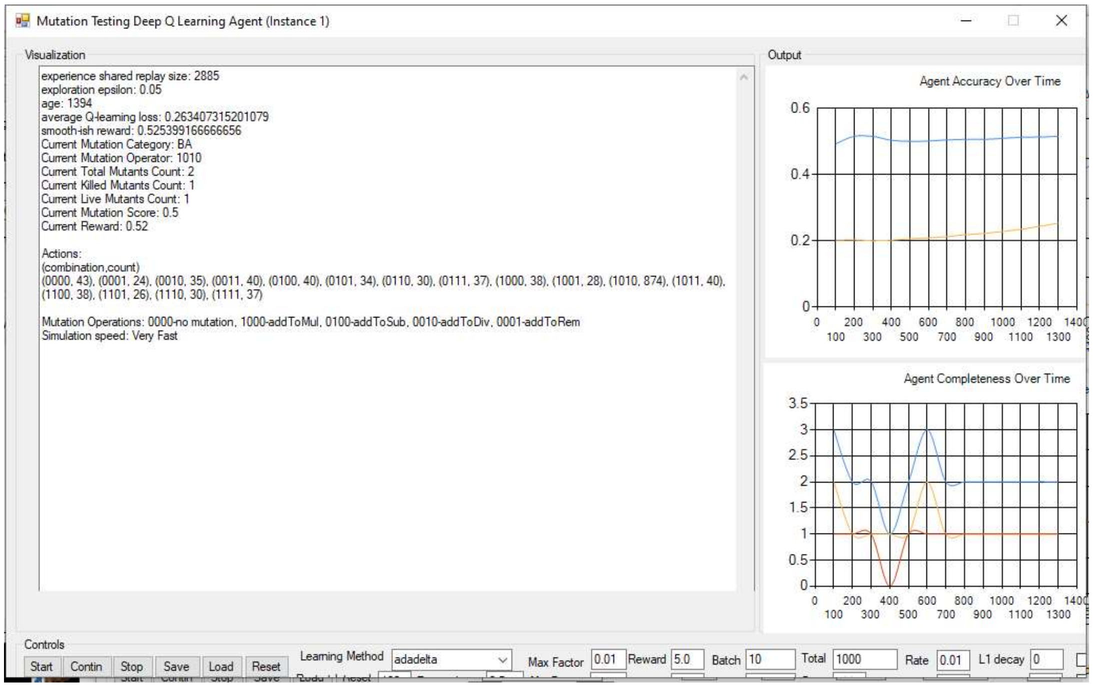
START(year-month-day_hours:mins:secs):2021-05-21_12:57:30 | END(year-month-day_hours:mins:secs):2021-05-22_02:22:47 | ELAPSED(day.hours_mins:secs):00.13_25:17


Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2885
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0206395165312658
smooth-ish reward: 0.93731749999999
Current Mutation Category: BA
Current Mutation Operator: 1000
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 40), (0001, 47), (0010, 30), (0011, 45), (0100, 332), (0101, 33), (0110, 36), (0111, 26), (1000, 365), (1001, 30), (1010, 45), (1011, 29),
(1100, 345), (1101, 36), (1110, 36), (1111, 26)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

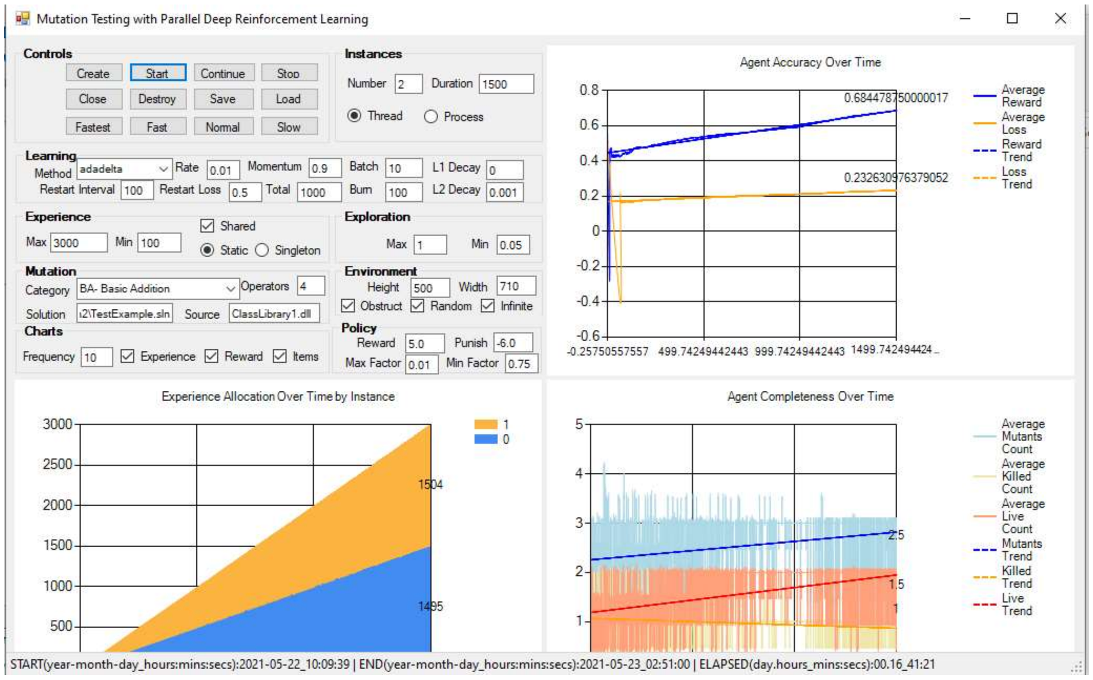
Controls: Start, Contin, Stop, Save, Load, Reset, Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2885
exploration epsilon: 0.05
age: 1394
average Q-learning loss: 0.263407315201079
smooth-ish reward: 0.525399166666656
Current Mutation Category: BA
Current Mutation Operator: 1010
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 43), (0001, 24), (0010, 35), (0011, 40), (0100, 40), (0101, 34), (0110, 30), (0111, 37), (1000, 38), (1001, 28), (1010, 874), (1011, 40),
(1100, 38), (1101, 26), (1110, 30), (1111, 37)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

2



Controls

Instances
Number 2 Duration 1500
[✓ Thread] Process

Learning
Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0
Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience
Max 3000 Min 100 [✓ Shared]
[✓ Static] Singleton

Exploration
Max 1 Min 0.05

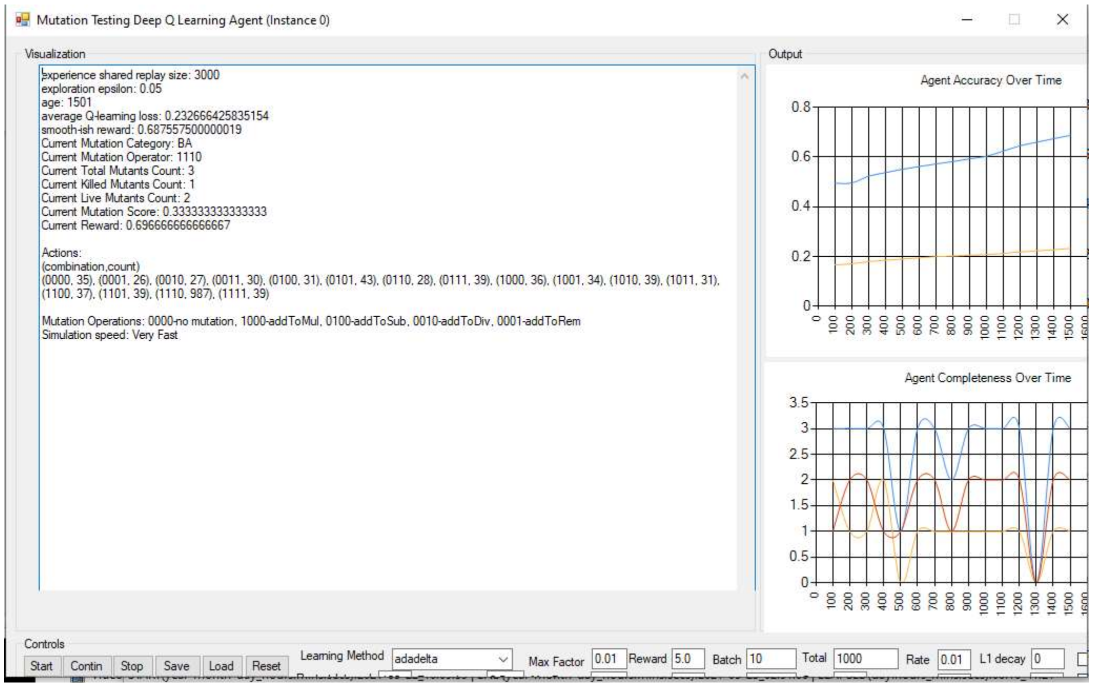
Mutation
Category BA- Basic Addition Operators 4
Solution \2\TestExample.sln Source ClassLibrary1.dll

Environment
Height 500 Width 710
[✓ Obstruct] [✓ Random] [✓ Infinite]

Charts
Frequency 10 [✓ Experience] [✓ Reward] [✓ Items]

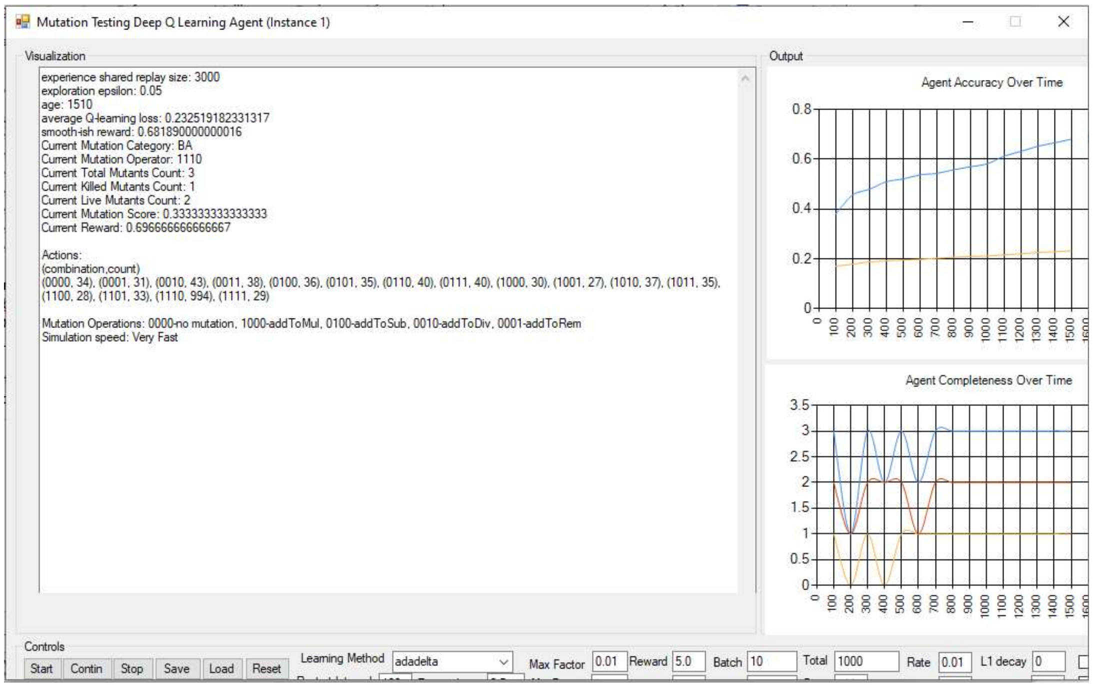
Policy
Reward 5.0 Punish -6.0
Max Factor 0.01 Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-05-22_10:09:39 | END(year-month-day_hours:mins:secs):2021-05-23_02:51:00 | ELAPSED(day.hours_mins:secs):00.16_41:21


Mutation Testing Deep Q Learning Agent (Instance 0)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 3000
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.232666425835154
smooth-ish reward: 0.687557500000019
Current Mutation Category: BA
Current Mutation Operator: 1110
Current Total Mutants Count: 3
Current Killed Mutants Count: 1
Current Live Mutants Count: 2
Current Mutation Score: 0.333333333333333
Current Reward: 0.696666666666667

Actions:
(combination,count)
(0000, 35), (0001, 26), (0010, 27), (0011, 30), (0100, 31), (0101, 43), (0110, 28), (0111, 39), (1000, 36), (1001, 34), (1010, 39), (1011, 31),
(1100, 37), (1101, 39), (1110, 987), (1111, 39)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

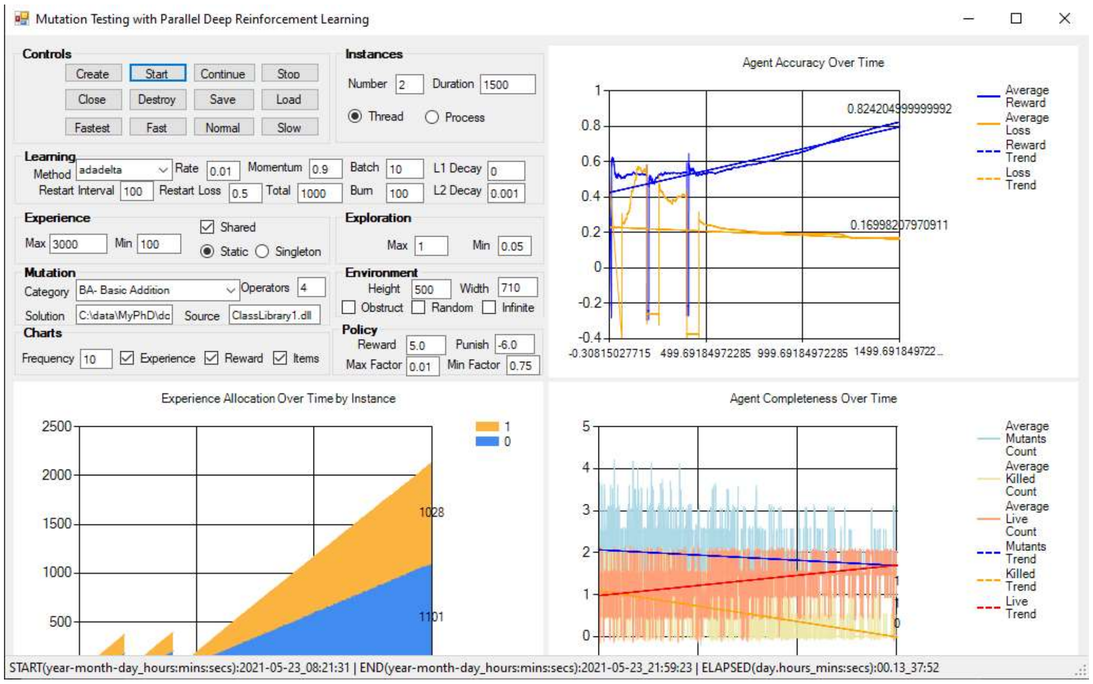
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 3000
exploration epsilon: 0.05
age: 1510
average Q-learning loss: 0.232519182331317
smooth-ish reward: 0.681890000000016
Current Mutation Category: BA
Current Mutation Operator: 1110
Current Total Mutants Count: 3
Current Killed Mutants Count: 1
Current Live Mutants Count: 2
Current Mutation Score: 0.333333333333333
Current Reward: 0.696666666666667

Actions:
(combination,count)
(0000, 34), (0001, 31), (0010, 43), (0011, 38), (0100, 36), (0101, 35), (0110, 40), (0111, 40), (1000, 30), (1001, 27), (1010, 37), (1011, 35),
(1100, 28), (1101, 33), (1110, 994), (1111, 29)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

3


Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, Shared ✓, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

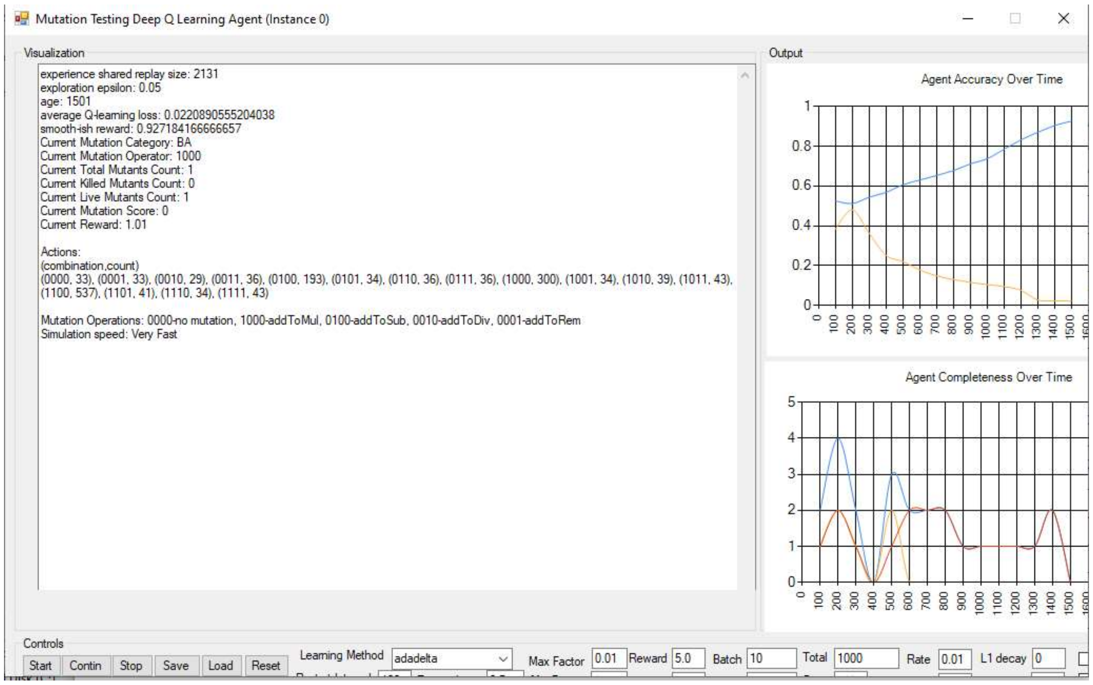
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, Obstruct ☐, Random ☐, Infinite ☐

Charts: Frequency 10, Experience ✓, Reward ✓, Items ✓

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

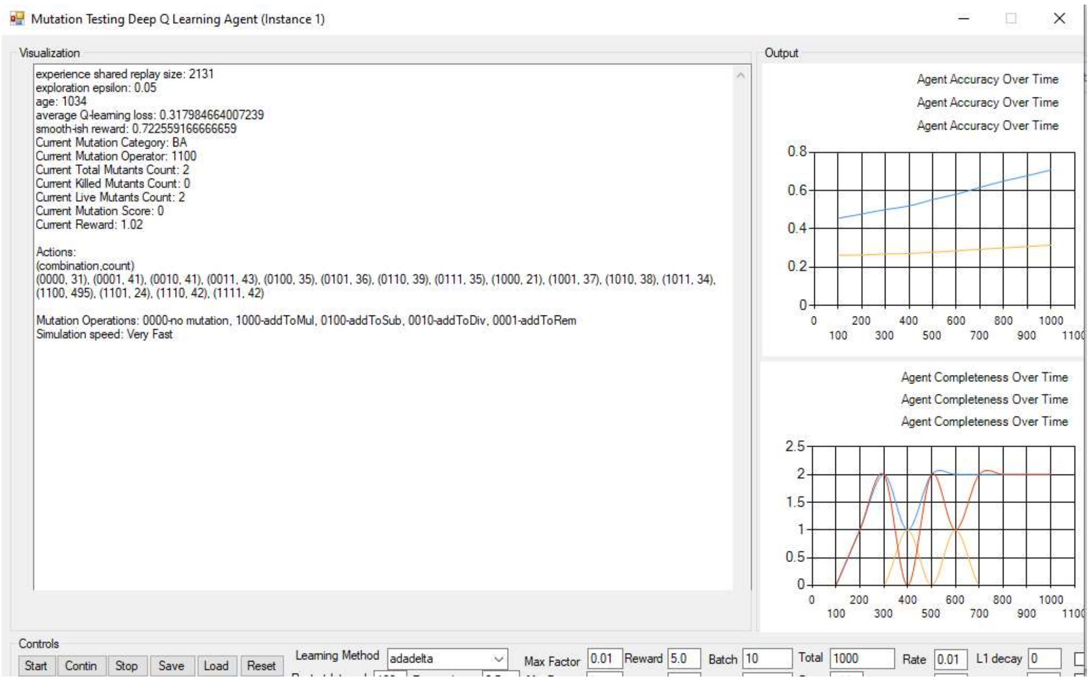
START(year-month-day_hours:mins:secs):2021-05-23_08:21:31 | END(year-month-day_hours:mins:secs):2021-05-23_21:59:23 | ELAPSED(day.hours_mins:secs):00.13_37:52


Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2131
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0220890555204038
smooth-ish reward: 0.927184166666657
Current Mutation Category: BA
Current Mutation Operator: 1000
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 33), (0001, 33), (0010, 29), (0011, 36), (0100, 193), (0101, 34), (0110, 36), (0111, 36), (1000, 300), (1001, 34), (1010, 39), (1011, 43),
(1100, 537), (1101, 41), (1110, 34), (1111, 43)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

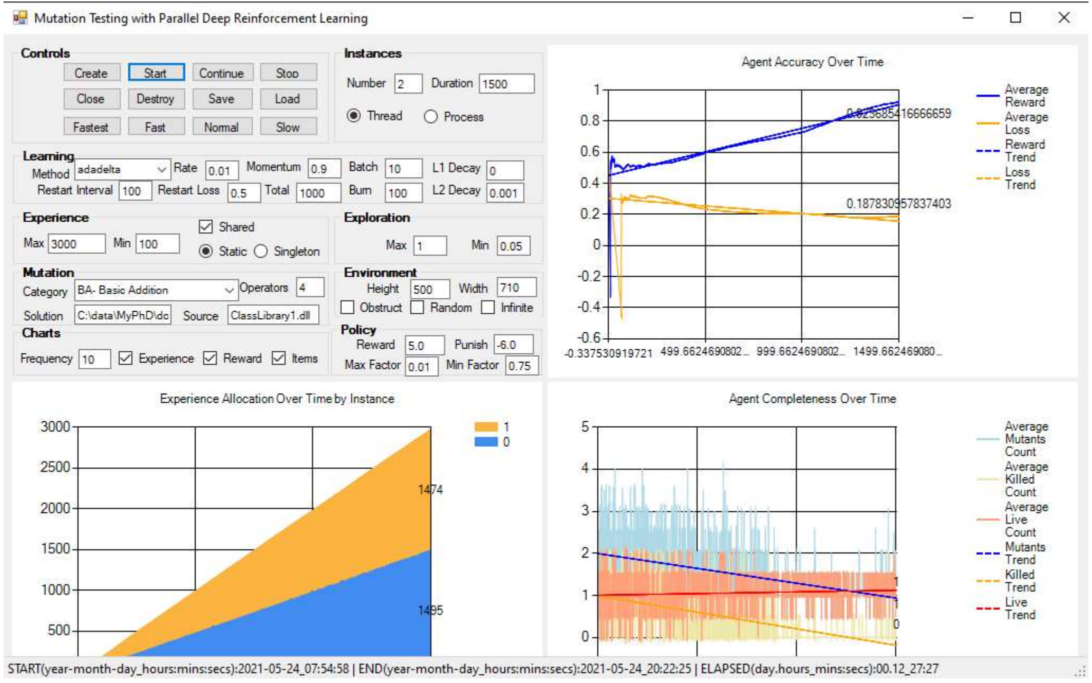
Controls: Start, Contin, Stop, Save, Load, Reset; Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)
Visualization
Output
Agent Accuracy Over Time
Agent Accuracy Over Time
Agent Accuracy Over Time
Agent Completeness Over Time
Agent Completeness Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2131
exploration epsilon: 0.05
age: 1034
average Q-learning loss: 0.317984664007239
smooth-ish reward: 0.722559166666659
Current Mutation Category: BA
Current Mutation Operator: 1100
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 31), (0001, 41), (0010, 41), (0011, 43), (0100, 35), (0101, 36), (0110, 39), (0111, 35), (1000, 21), (1001, 37), (1010, 38), (1011, 34),
(1100, 495), (1101, 24), (1110, 42), (1111, 42)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

4



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ✓ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

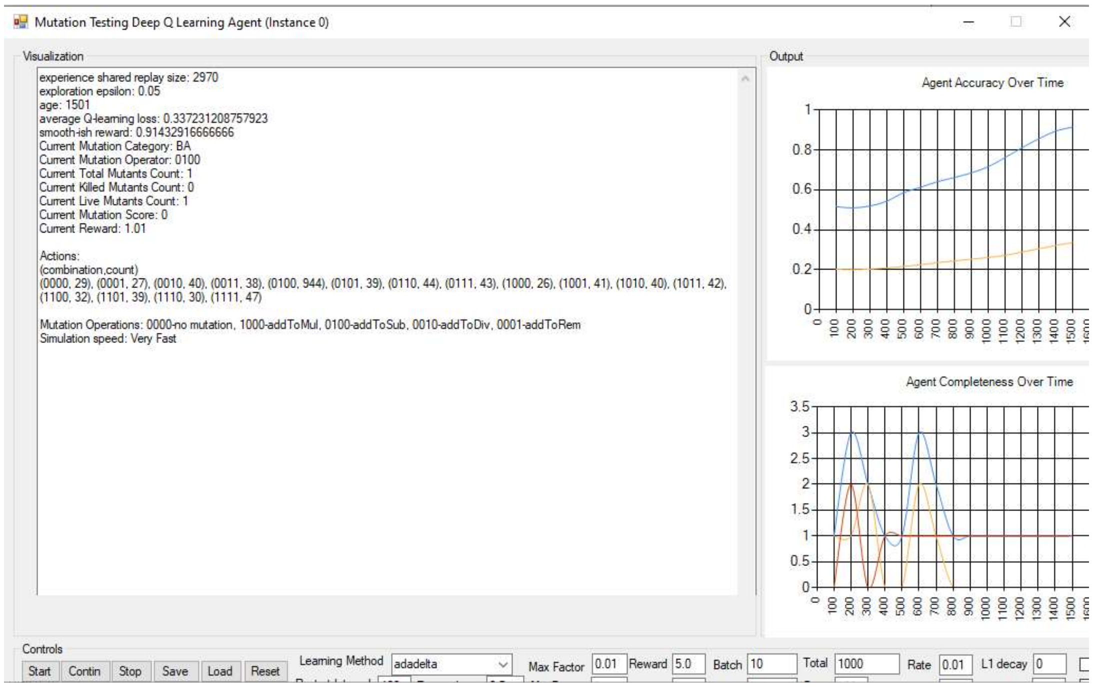
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☐ Infinite

Charts: Frequency 10, ✓ Experience, ✓ Reward, ✓ Items

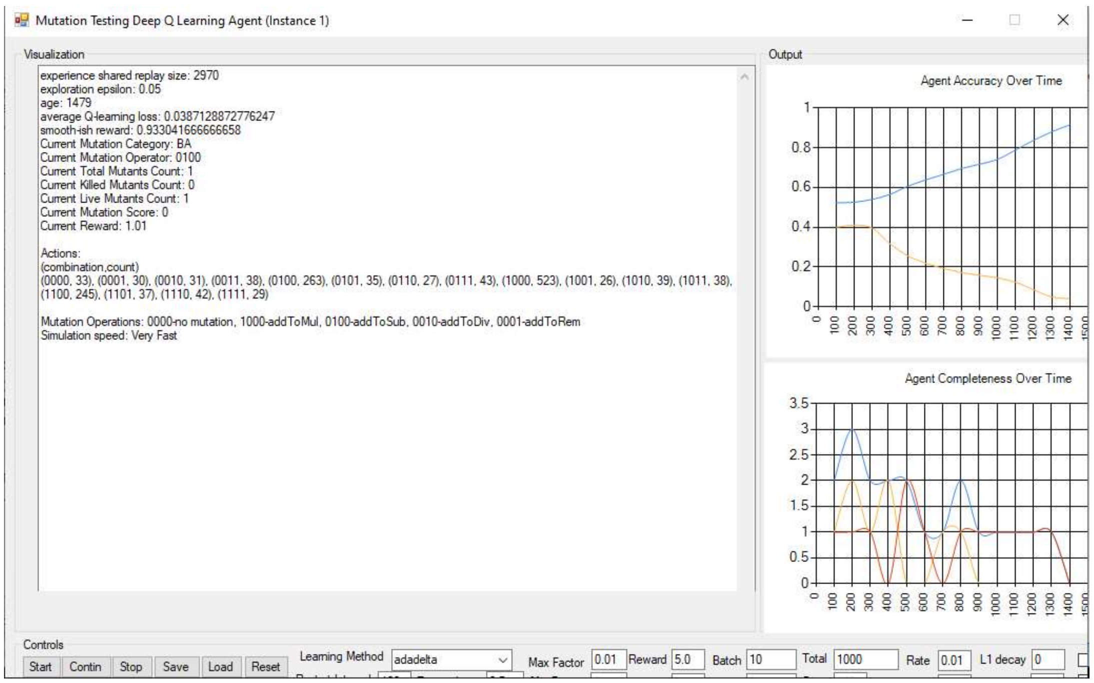
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-05-24_07:54:58 | END(year-month-day_hours:mins:secs):2021-05-24_20:22:25 | ELAPSED(day.hours_mins:secs):00.12_27:27


Mutation Testing Deep Q Learning Agent (Instance 0)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2970
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.337231208757923
smooth-ish reward: 0.91432916666666
Current Mutation Category: BA
Current Mutation Operator: 0100
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 29), (0001, 27), (0010, 40), (0011, 38), (0100, 944), (0101, 39), (0110, 44), (0111, 43), (1000, 26), (1001, 41), (1010, 40), (1011, 42),
(1100, 32), (1101, 39), (1110, 30), (1111, 47)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

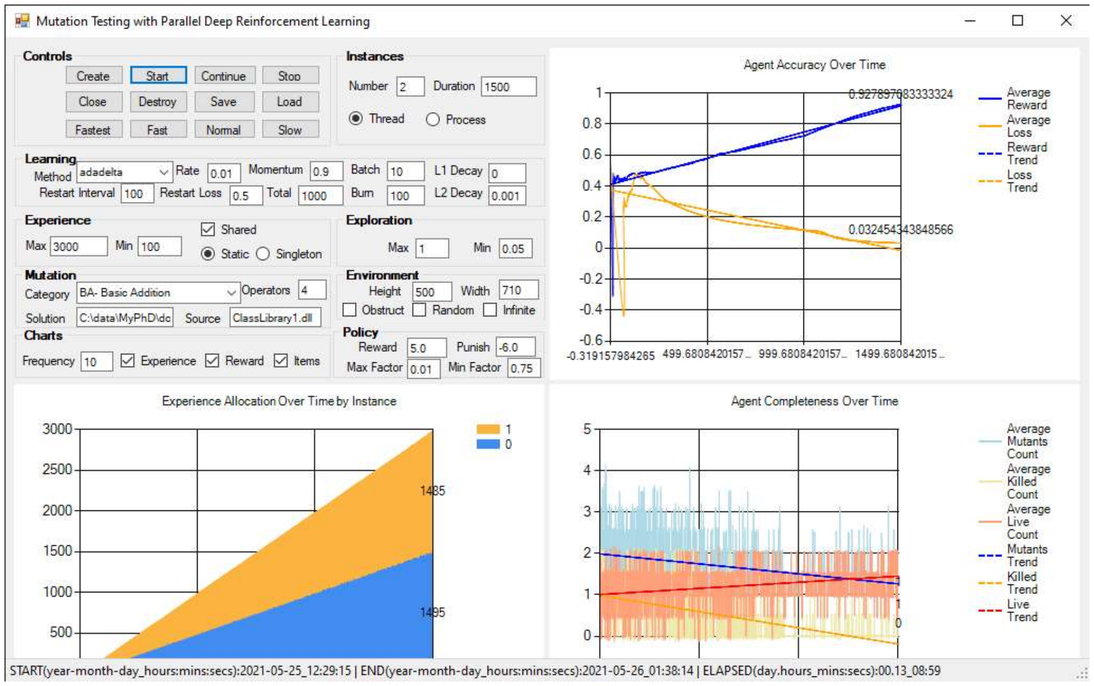
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0



```
experience shared replay size: 2970
exploration epsilon: 0.05
age: 1479
average Q-learning loss: 0.0387128872776247
smooth-ish reward: 0.933041666666658
Current Mutation Category: BA
Current Mutation Operator: 0100
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 33), (0001, 30), (0010, 31), (0011, 38), (0100, 263), (0101, 35), (0110, 27), (0111, 43), (1000, 523), (1001, 26), (1010, 39), (1011, 38),
(1100, 245), (1101, 37), (1110, 42), (1111, 29)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

5



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

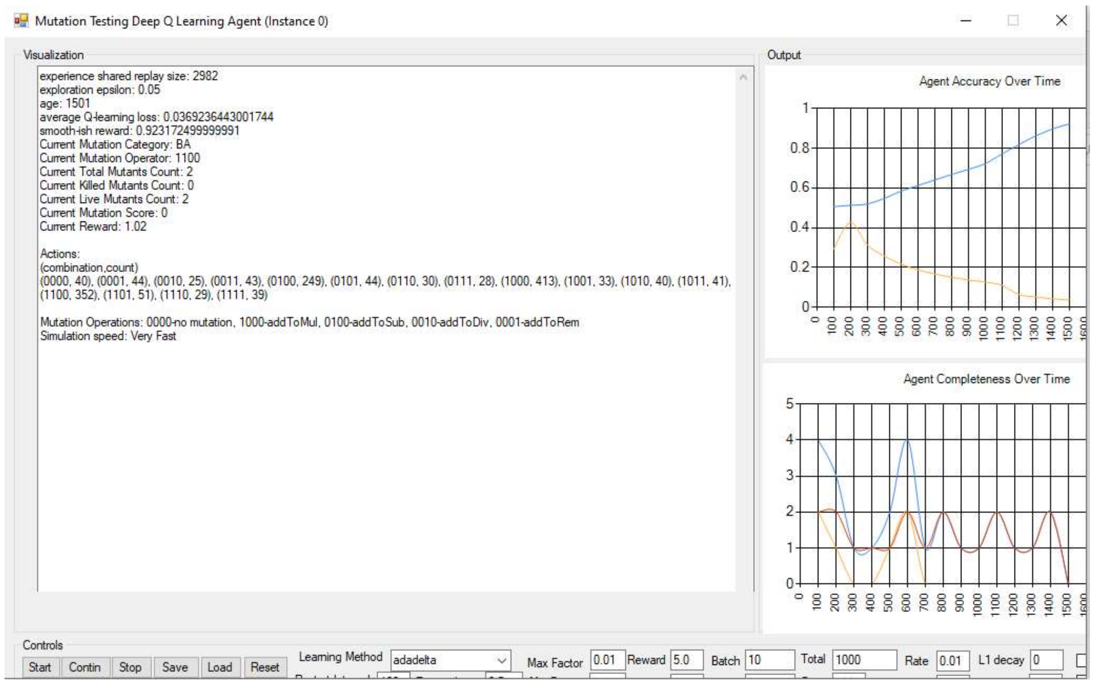
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

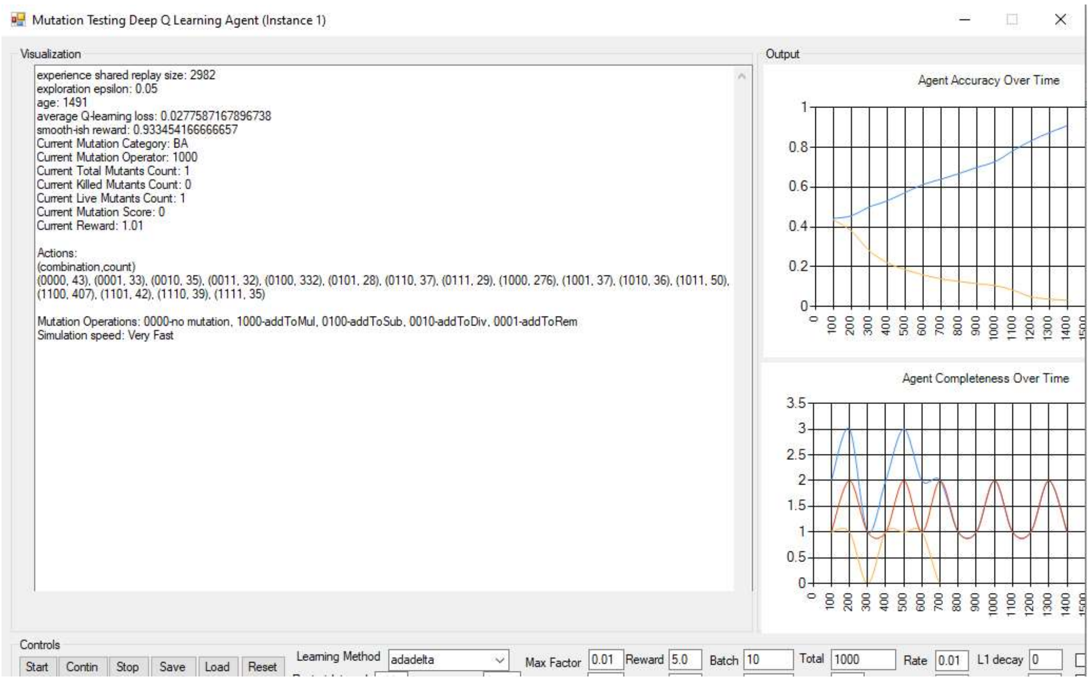
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-05-25_12:29:15 | END(year-month-day_hours:mins:secs):2021-05-26_01:38:14 | ELAPSED(day.hours_mins:secs):00.13_08:59


Mutation Testing Deep Q Learning Agent (Instance 0)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2982
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0369236443001744
smooth-ish reward: 0.923172499999991
Current Mutation Category: BA
Current Mutation Operator: 1100
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 40), (0001, 44), (0010, 25), (0011, 43), (0100, 249), (0101, 44), (0110, 30), (0111, 28), (1000, 413), (1001, 33), (1010, 40), (1011, 41),
(1100, 352), (1101, 51), (1110, 29), (1111, 39)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)

Visualization

```
experience shared replay size: 2982
exploration epsilon: 0.05
age: 1491
average Q-learning loss: 0.0277587167896738
smooth-ish reward: 0.933454166666657
Current Mutation Category: BA
Current Mutation Operator: 1000
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 43), (0001, 33), (0010, 35), (0011, 32), (0100, 332), (0101, 28), (0110, 37), (0111, 29), (1000, 276), (1001, 37), (1010, 36), (1011, 50),
(1100, 407), (1101, 42), (1110, 39), (1111, 35)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

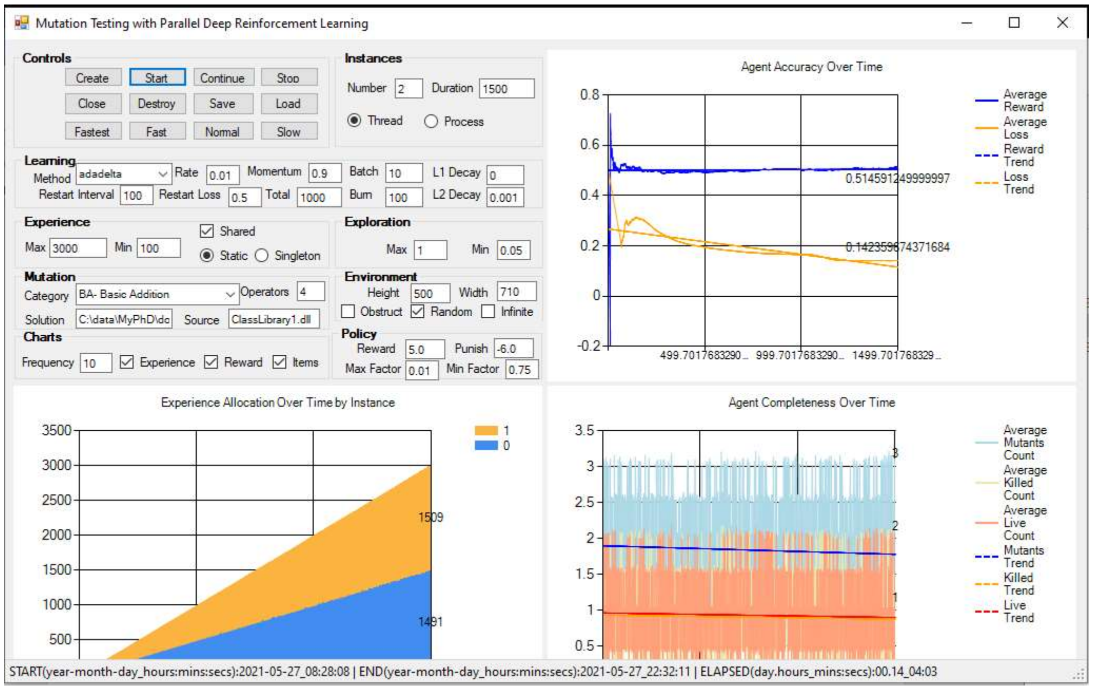
Controls

Learning Method: adadelta; Max Factor: 0.01; Reward: 5.0; Batch: 10; Total: 1000; Rate: 0.01; L1 decay: 0

Random selection of mutation operators:

Results						
Test Metrics	**Test Average**	**Test Results**				
test runs		1	2	3	4	5
avg elapsed time (hh:mm:s	16:25:42	14:04:03	17:18:30	15:43:58	16:37:36	18:24:25
maximum action	n/a					
average mutation score	0.68					
average mutant total	2.500000000000000	3.00	1.50	3.00	2.00	3.00
average killed count	1.700000000000000	2.00	1.00	2.00	1.50	2.00
average live count	0.800000000000000	1.00	0.50	1.00	0.50	1.00
average Q-learn loss	0.126778508637186	0.14235967	0.18430662	0.08087368	0.07788896	0.14846361
smooth-ish reward	0.504345666666664	0.51459125	0.49512833	0.50377042	0.50956583	0.49867250
Test Metrics	**Test Average**	**Test Results**				
test runs - instance0		1	2	3	4	5
maximum action	n/a					
average Q-learn loss	0.128149205757168	0.08421698	0.1836353	0.07598023	0.07868408	0.21822945
smooth-ish reward	0.496130999999998	0.49842083	0.49088250	0.49715583	0.50225583	0.49194000
test runs - instance1		1	2	3	4	5
maximum action	n/a					
average Q-learn loss	0.125384138198002	0.20050770	0.18492067	0.08572057	0.07710995	0.07866180
smooth-ish reward	0.513328999999998	0.53178167	0.49937417	0.51188500	0.51739583	0.50620833
	Test Summary	1	2	3	4	5
maximum action	**n/a**					
average Q-learn loss	**0.126766671977585**	0.14236234	0.18427798	0.0808504	0.07789702	0.14844562
smooth-ish reward	**0.504729999999998**	0.51510125	0.49512833	0.50452042	0.50982583	0.49907417

1


Mutation Testing with Parallel Deep Reinforcement Learning

Controls

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ✓ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

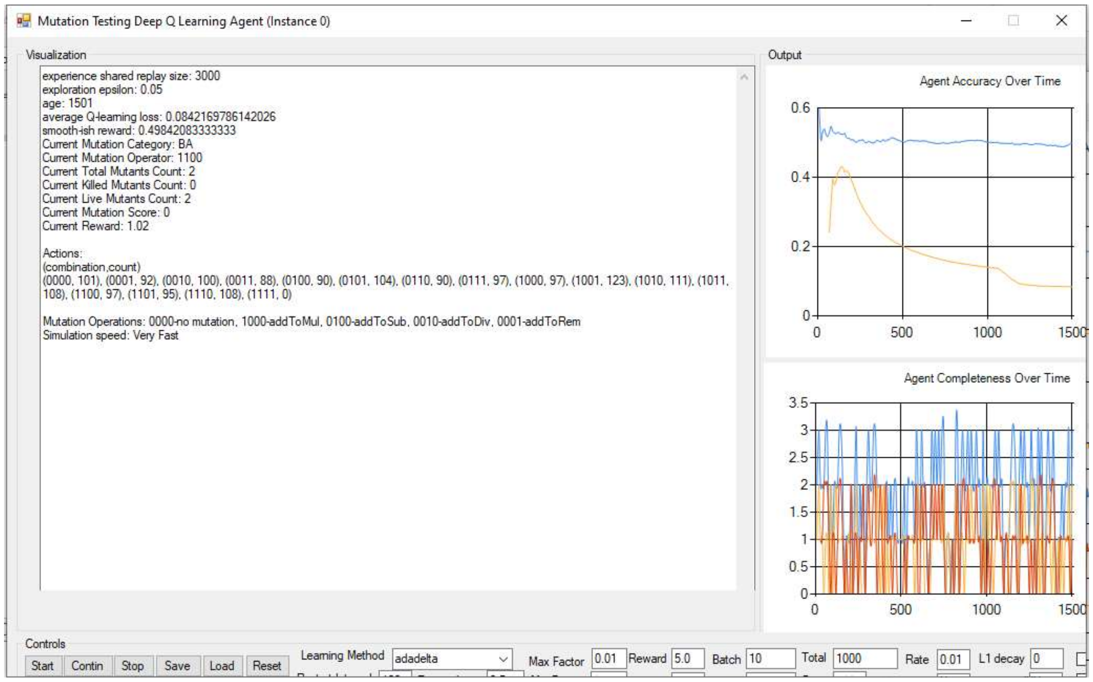
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ✓ Random, ☐ Infinite

Charts: Frequency 10, ✓ Experience, ✓ Reward, ✓ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

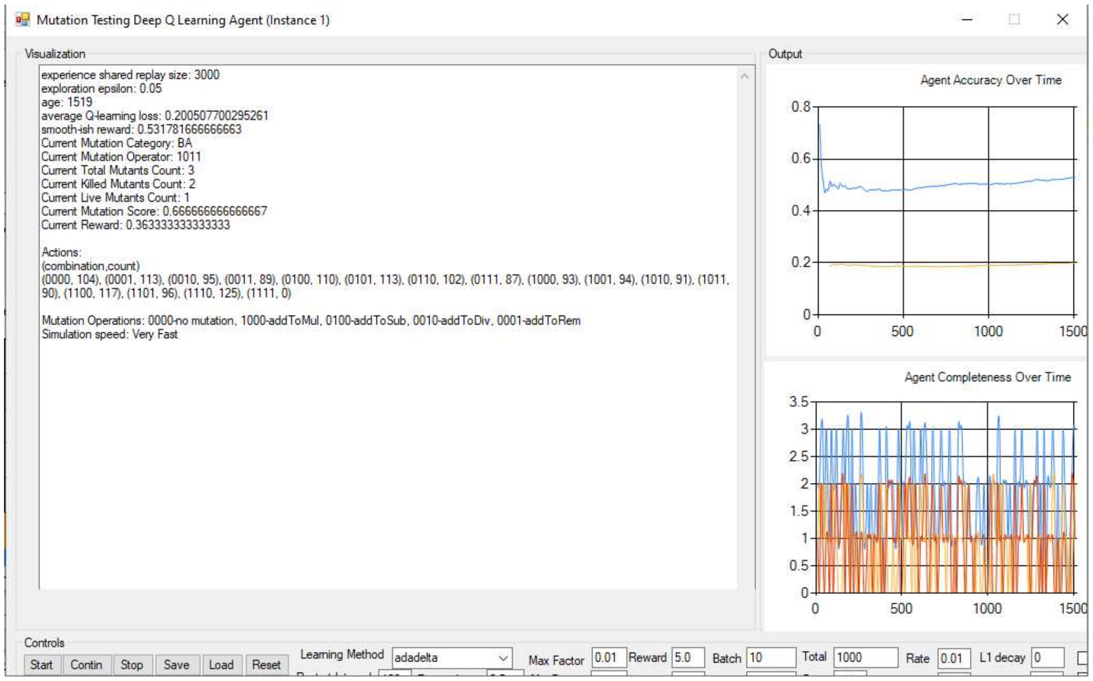
START(year-month-day_hours:mins:secs):2021-05-27_08:28:08 | END(year-month-day_hours:mins:secs):2021-05-27_22:32:11 | ELAPSED(day.hours_mins:secs):00.14_04:03


Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 3000
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0842169786142026
smooth-ish reward: 0.49842083333333
Current Mutation Category: BA
Current Mutation Operator: 1100
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 101), (0001, 92), (0010, 100), (0011, 88), (0100, 90), (0101, 104), (0110, 90), (0111, 97), (1000, 97), (1001, 123), (1010, 111), (1011,
108), (1100, 97), (1101, 95), (1110, 108), (1111, 0)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

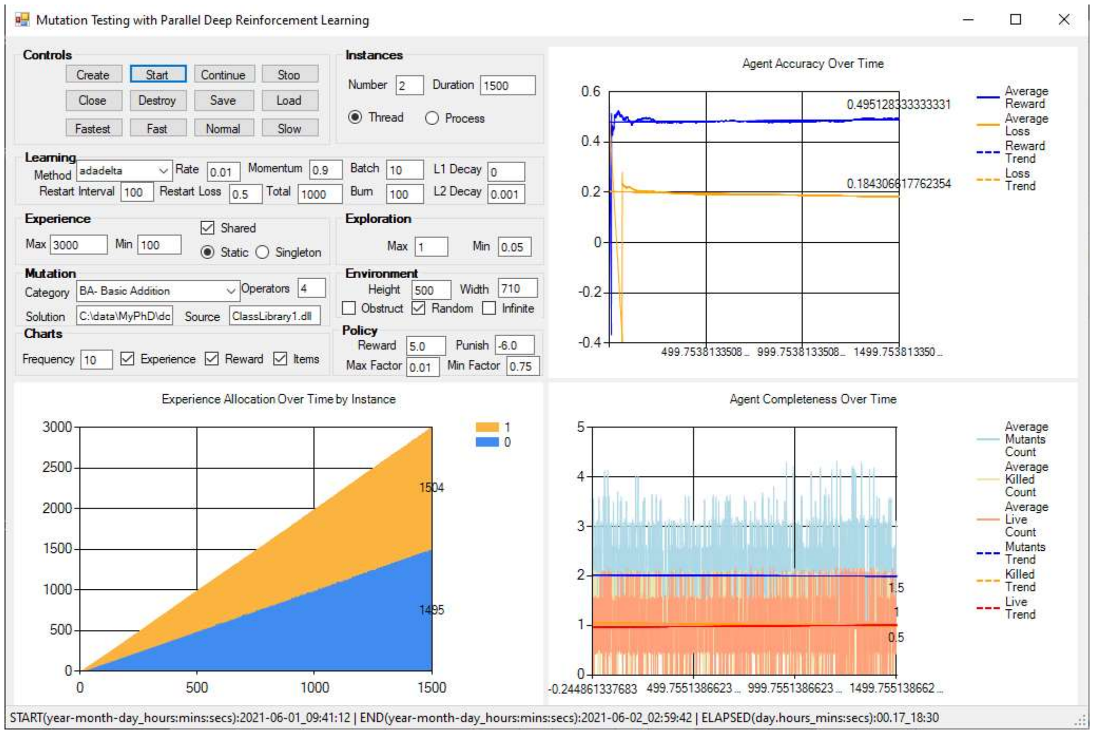
Controls: Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 3000
exploration epsilon: 0.05
age: 1519
average Q-learning loss: 0.200507700295261
smooth-ish reward: 0.531781666666663
Current Mutation Category: BA
Current Mutation Operator: 1011
Current Total Mutants Count: 3
Current Killed Mutants Count: 2
Current Live Mutants Count: 1
Current Mutation Score: 0.666666666666667
Current Reward: 0.363333333333333

Actions:
(combination,count)
(0000, 104), (0001, 113), (0010, 95), (0011, 89), (0100, 110), (0101, 113), (0110, 102), (0111, 87), (1000, 93), (1001, 94), (1010, 91), (1011,
90), (1100, 117), (1101, 96), (1110, 125), (1111, 0)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

2



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

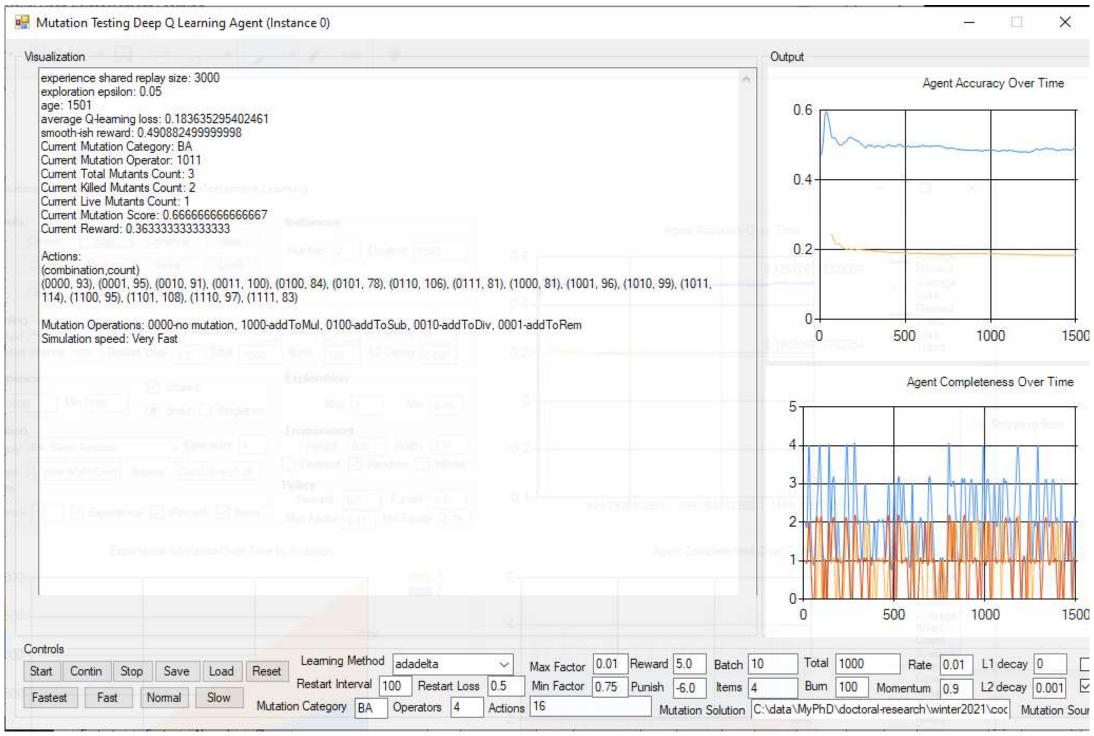
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☑ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

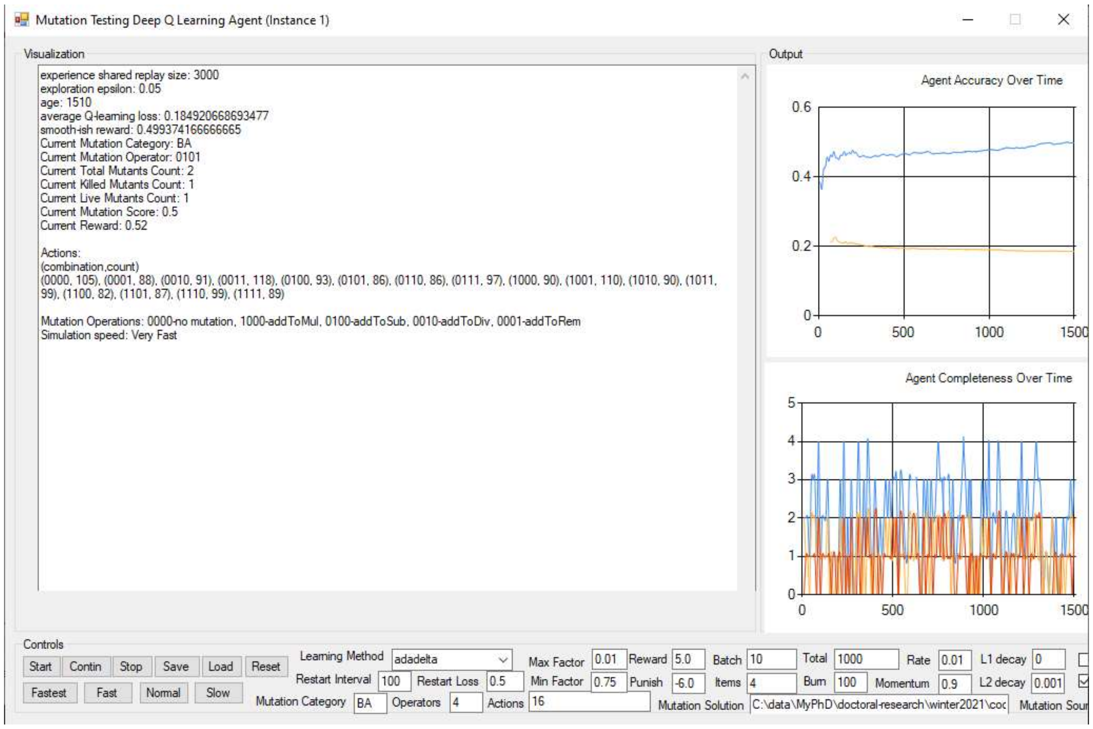
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-01_09:41:12 | END(year-month-day_hours:mins:secs):2021-06-02_02:59:42 | ELAPSED(day.hours_mins:secs):00.17_18:30



```
experience shared replay size: 3000
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.183635295402461
smooth-ish reward: 0.490882499999998
Current Mutation Category: BA
Current Mutation Operator: 1011
Current Total Mutants Count: 3
Current Killed Mutants Count: 2
Current Live Mutants Count: 1
Current Mutation Score: 0.666666666666667
Current Reward: 0.363333333333333

Actions:
(combination,count)
(0000, 93), (0001, 95), (0010, 91), (0011, 100), (0100, 84), (0101, 78), (0110, 106), (0111, 81), (1000, 81), (1001, 96), (1010, 99), (1011,
114), (1100, 95), (1101, 108), (1110, 97), (1111, 83)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

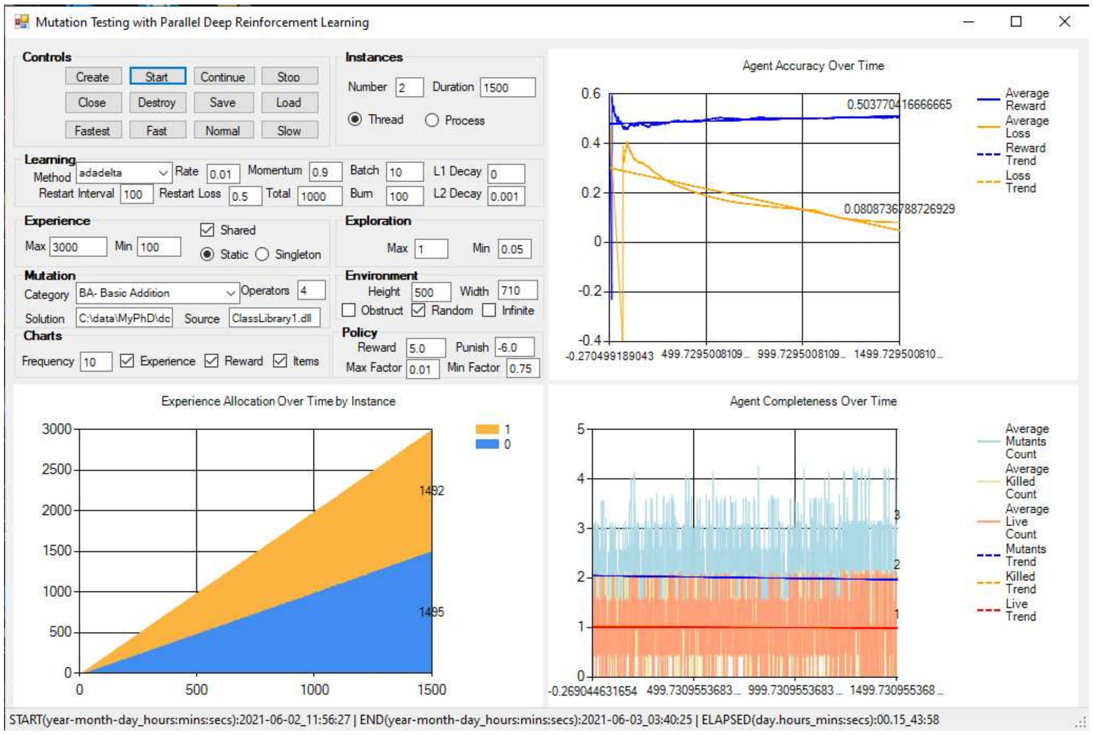
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, Mutation Category BA, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Sour



```
experience shared replay size: 3000
exploration epsilon: 0.05
age: 1510
average Q-learning loss: 0.184920668693477
smooth-ish reward: 0.499374166666665
Current Mutation Category: BA
Current Mutation Operator: 0101
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 105), (0001, 88), (0010, 91), (0011, 118), (0100, 93), (0101, 86), (0110, 86), (0111, 97), (1000, 90), (1001, 110), (1010, 90), (1011,
99), (1100, 82), (1101, 87), (1110, 99), (1111, 89)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Category: BA
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Sour

3



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, Shared ✓, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

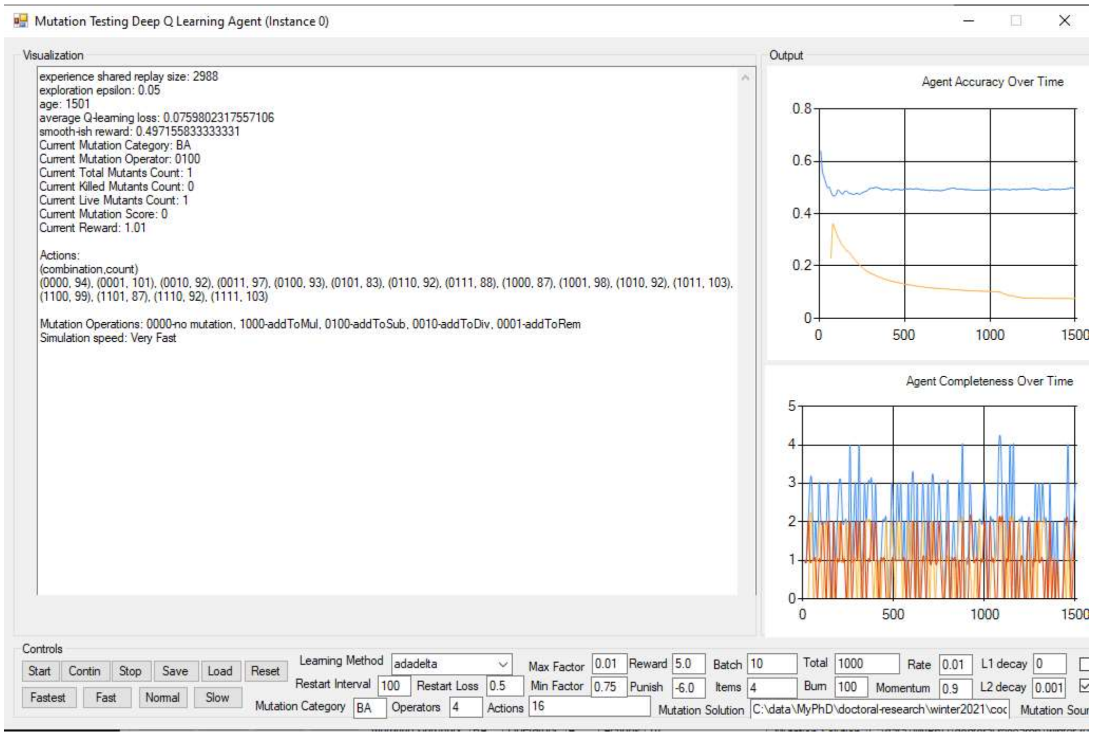
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, Obstruct ☐, Random ✓, Infinite ☐

Charts: Frequency 10, Experience ✓, Reward ✓, Items ✓

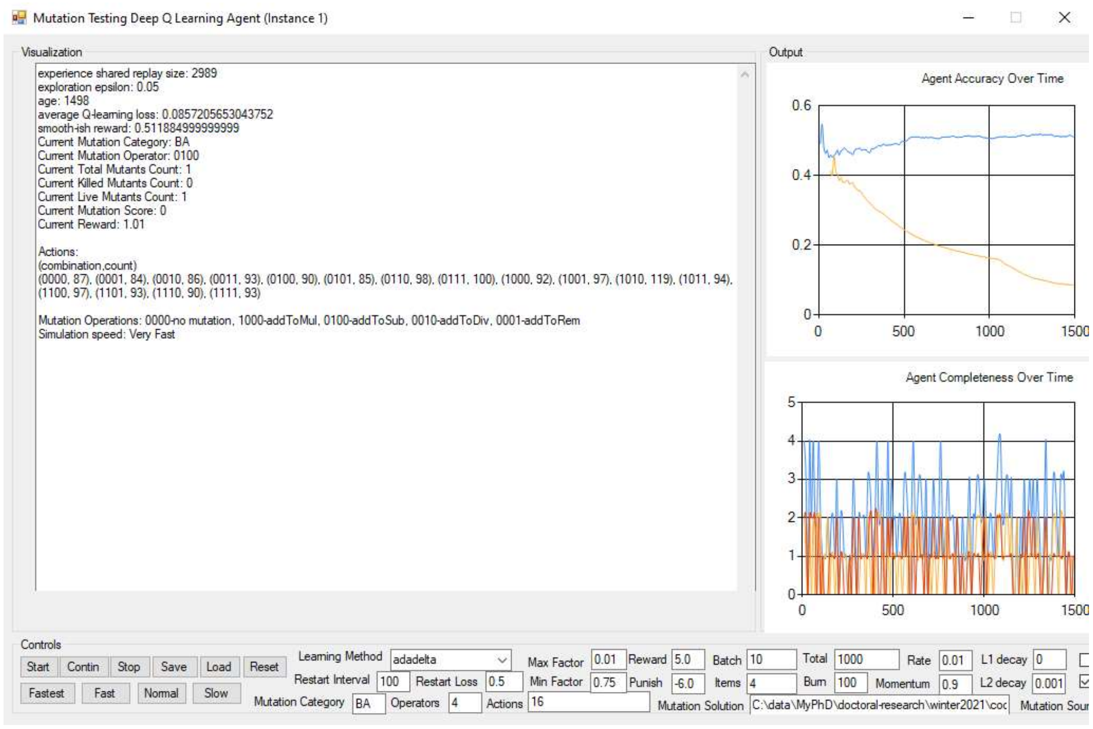
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-02_11:56:27 | END(year-month-day_hours:mins:secs):2021-06-03_03:40:25 | ELAPSED(day.hours_mins:secs):00.15_43:58



```
experience shared replay size: 2988
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0759802317557106
smooth-ish reward: 0.497155833333331
Current Mutation Category: BA
Current Mutation Operator: 0100
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 94), (0001, 101), (0010, 92), (0011, 97), (0100, 93), (0101, 83), (0110, 92), (0111, 88), (1000, 87), (1001, 98), (1010, 92), (1011, 103),
(1100, 99), (1101, 87), (1110, 92), (1111, 103)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

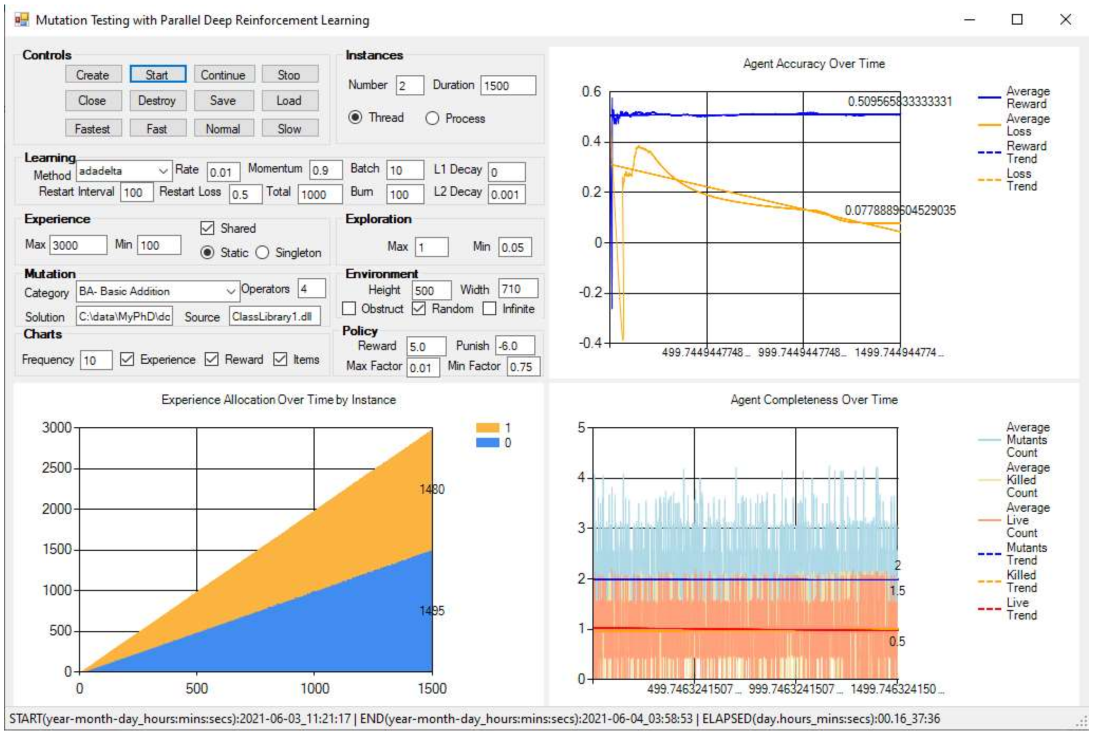
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, Mutation Category BA, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Sour



```
experience shared replay size: 2989
exploration epsilon: 0.05
age: 1498
average Q-learning loss: 0.0857205653043752
smooth-ish reward: 0.511884999999999
Current Mutation Category: BA
Current Mutation Operator: 0100
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 87), (0001, 84), (0010, 86), (0011, 93), (0100, 90), (0101, 85), (0110, 98), (0111, 100), (1000, 92), (1001, 97), (1010, 119), (1011, 94),
(1100, 97), (1101, 93), (1110, 90), (1111, 93)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BA
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Sour

4



Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

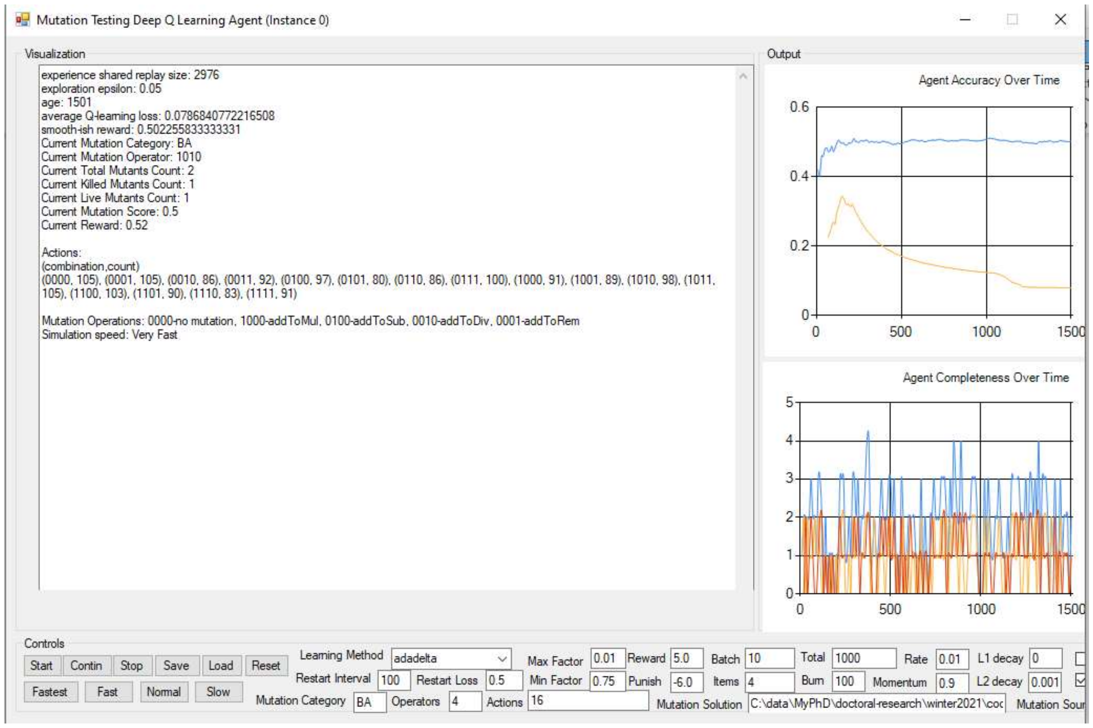
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☑ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-03_11:21:17 | END(year-month-day_hours:mins:secs):2021-06-04_03:58:53 | ELAPSED(day.hours_mins:secs):00.16_37:36



Mutation Testing Deep Q Learning Agent (Instance 0)

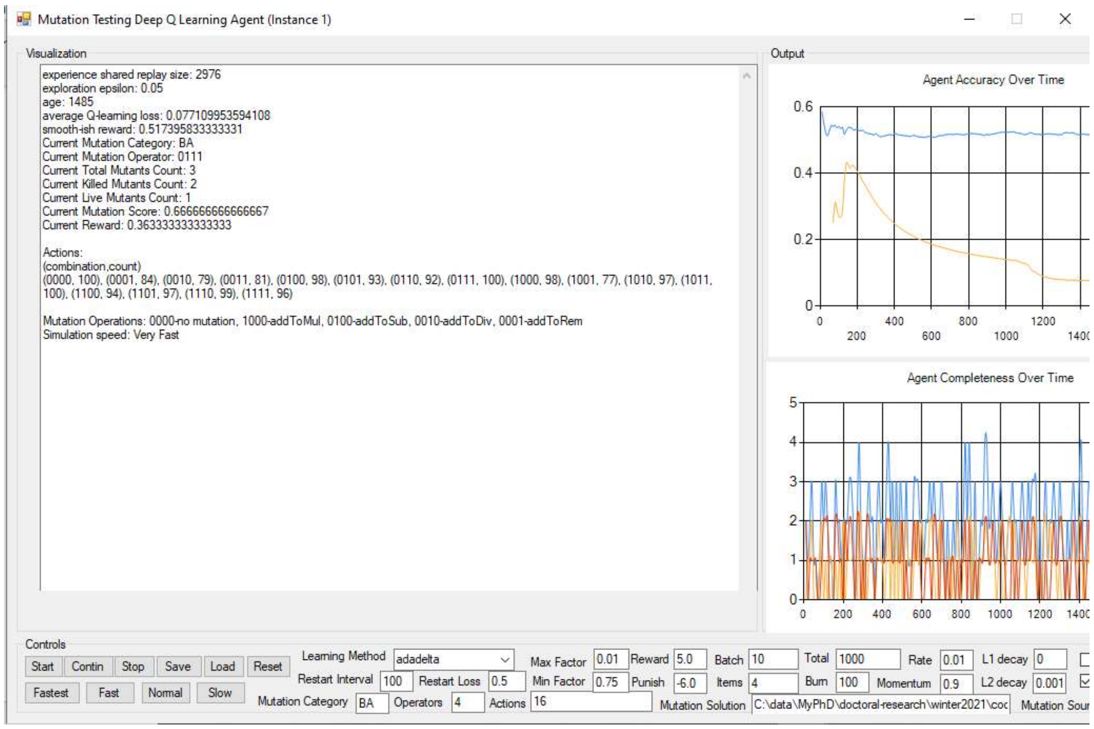
Visualization

```
experience shared replay size: 2976
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0786840772216508
smooth-ish reward: 0.502255833333331
Current Mutation Category: BA
Current Mutation Operator: 1010
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 105), (0001, 105), (0010, 86), (0011, 92), (0100, 97), (0101, 80), (0110, 86), (0111, 100), (1000, 91), (1001, 89), (1010, 98), (1011,
105), (1100, 103), (1101, 90), (1110, 83), (1111, 91)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Output

Controls: Start, Contin, Stop, Save, Load, Reset, Fastest, Fast, Normal, Slow

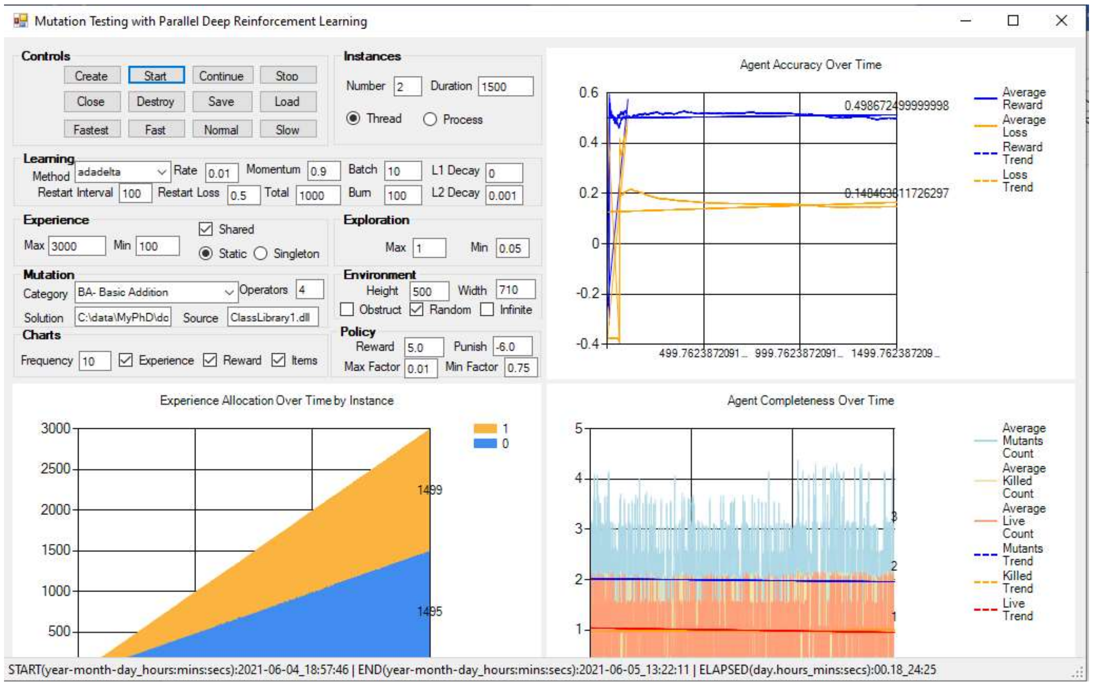
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, Mutation Category BA, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Sour



```
experience shared replay size: 2976
exploration epsilon: 0.05
age: 1485
average Q-learning loss: 0.077109953594108
smooth-ish reward: 0.517395833333331
Current Mutation Category: BA
Current Mutation Operator: 0111
Current Total Mutants Count: 3
Current Killed Mutants Count: 2
Current Live Mutants Count: 1
Current Mutation Score: 0.666666666666667
Current Reward: 0.363333333333333

Actions:
(combination,count)
(0000, 100), (0001, 84), (0010, 79), (0011, 81), (0100, 98), (0101, 93), (0110, 92), (0111, 100), (1000, 98), (1001, 77), (1010, 97), (1011,
100), (1100, 94), (1101, 97), (1110, 99), (1111, 96)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Category: BA
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Sour

5



Controls

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

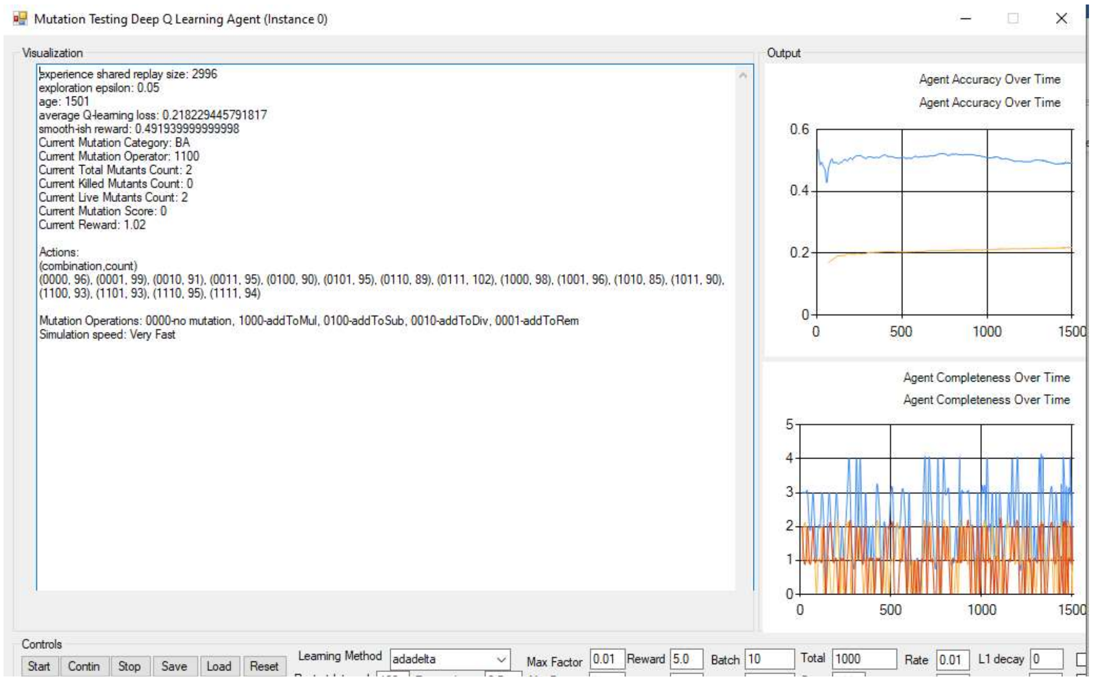
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☑ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

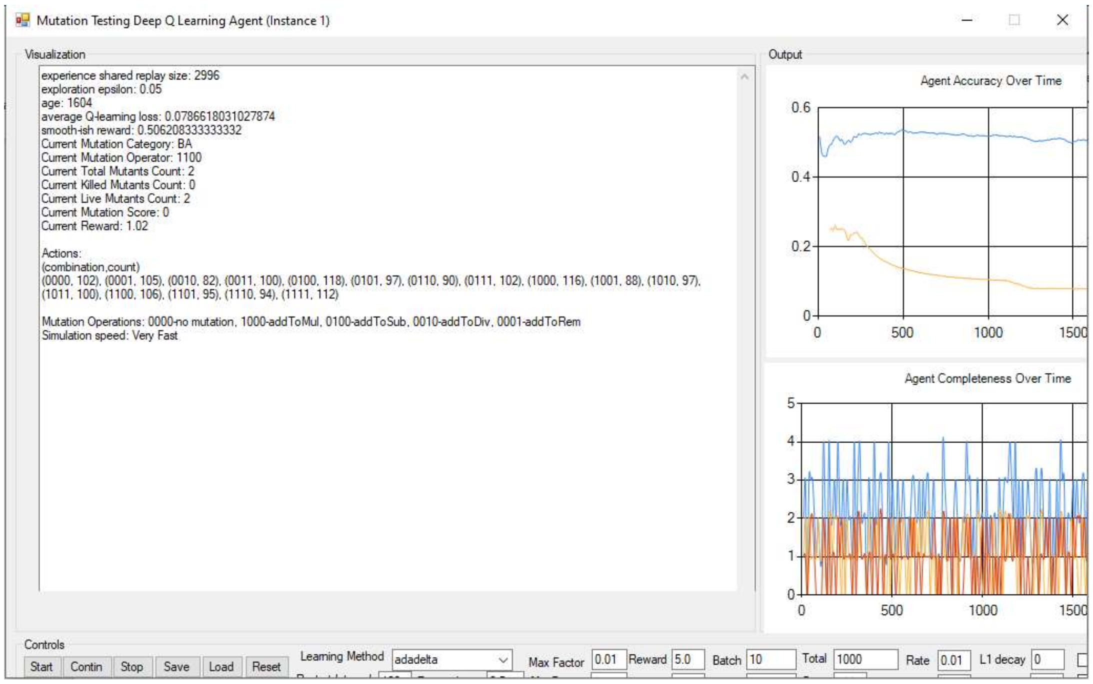
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-04_18:57:46 | END(year-month-day_hours:mins:secs):2021-06-05_13:22:11 | ELAPSED(day.hours_mins:secs):00.18_24:25


Mutation Testing Deep Q Learning Agent (Instance 0)
Visualization
Output
Agent Accuracy Over Time
Agent Accuracy Over Time
Agent Completeness Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.218229445791817
smooth-ish reward: 0.491939999999998
Current Mutation Category: BA
Current Mutation Operator: 1100
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 96), (0001, 99), (0010, 91), (0011, 95), (0100, 90), (0101, 95), (0110, 89), (0111, 102), (1000, 98), (1001, 96), (1010, 85), (1011, 90),
(1100, 93), (1101, 93), (1110, 95), (1111, 94)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

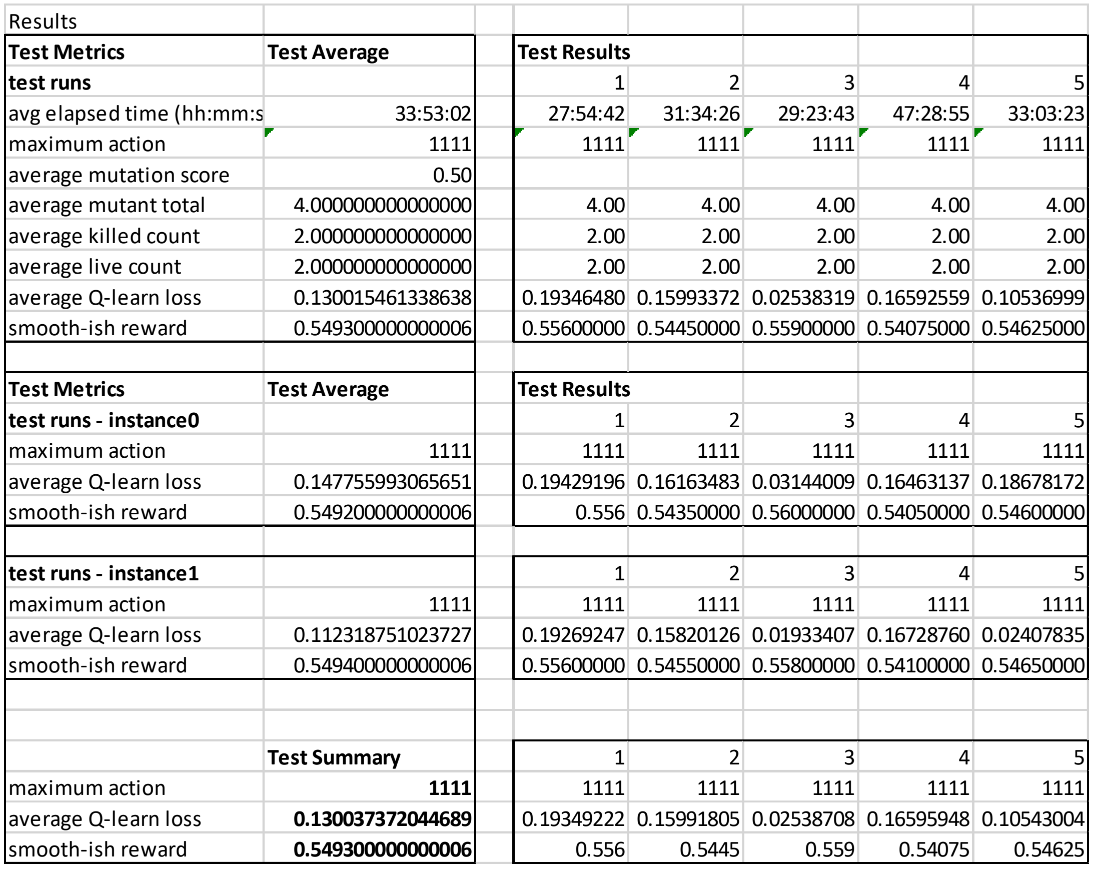
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1604
average Q-learning loss: 0.0786618031027874
smooth-ish reward: 0.506208333333332
Current Mutation Category: BA
Current Mutation Operator: 1100
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 102), (0001, 105), (0010, 82), (0011, 100), (0100, 118), (0101, 97), (0110, 90), (0111, 102), (1000, 116), (1001, 88), (1010, 97),
(1011, 100), (1100, 106), (1101, 95), (1110, 94), (1111, 112)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

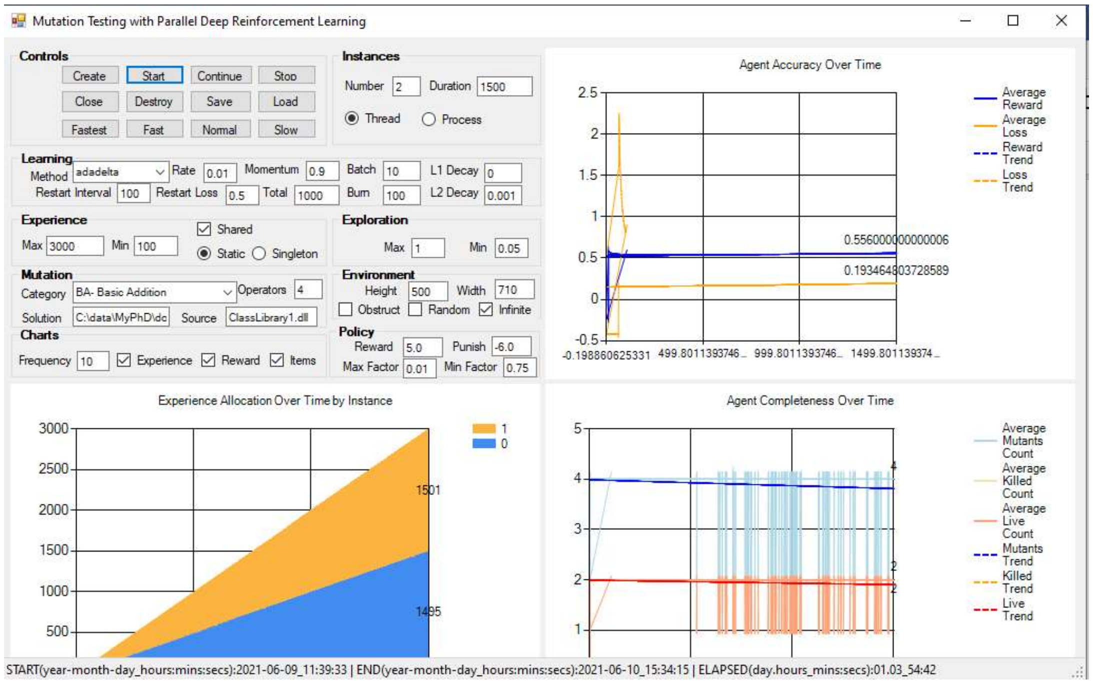
Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

Selection of all mutation operators:



Results						
Test Metrics	**Test Average**	**Test Results**				
test runs		1	2	3	4	5
avg elapsed time (hh:mm:s	33:53:02	27:54:42	31:34:26	29:23:43	47:28:55	33:03:23
maximum action	1111	1111	1111	1111	1111	1111
average mutation score	0.50					
average mutant total	4.000000000000000	4.00	4.00	4.00	4.00	4.00
average killed count	2.000000000000000	2.00	2.00	2.00	2.00	2.00
average live count	2.000000000000000	2.00	2.00	2.00	2.00	2.00
average Q-learn loss	0.130015461338638	0.19346480	0.15993372	0.02538319	0.16592559	0.10536999
smooth-ish reward	0.549300000000006	0.55600000	0.54450000	0.55900000	0.54075000	0.54625000
Test Metrics	**Test Average**	**Test Results**				
test runs - instance0		1	2	3	4	5
maximum action	1111	1111	1111	1111	1111	1111
average Q-learn loss	0.147755993065651	0.19429196	0.16163483	0.03144009	0.16463137	0.18678172
smooth-ish reward	0.549200000000006	0.556	0.54350000	0.56000000	0.54050000	0.54600000
test runs - instance1		1	2	3	4	5
maximum action	1111	1111	1111	1111	1111	1111
average Q-learn loss	0.112318751023727	0.19269247	0.15820126	0.01933407	0.16728760	0.02407835
smooth-ish reward	0.549400000000006	0.55600000	0.54550000	0.55800000	0.54100000	0.54650000
	Test Summary	1	2	3	4	5
maximum action	**1111**	1111	1111	1111	1111	1111
average Q-learn loss	**0.130037372044689**	0.19349222	0.15991805	0.02538708	0.16595948	0.10543004
smooth-ish reward	**0.549300000000006**	0.556	0.5445	0.559	0.54075	0.54625

1


Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, Thread [✓], Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, Shared [✓], Static [✓], Singleton

Exploration: Max 1, Min 0.05

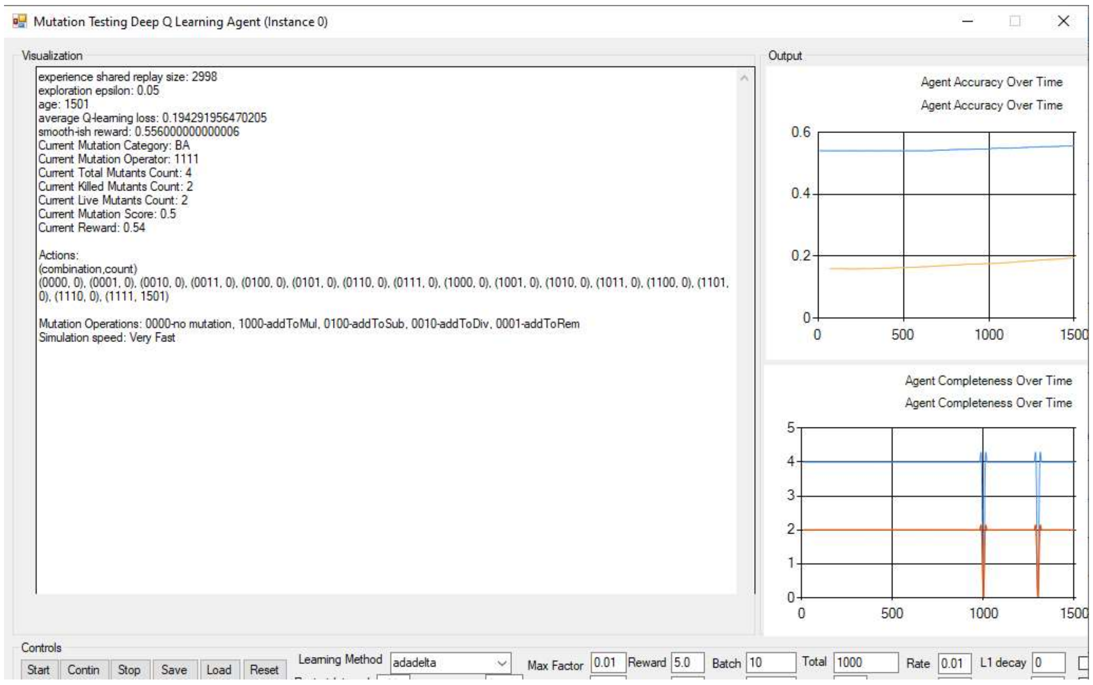
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, Obstruct, Random, Infinite [✓]

Charts: Frequency 10, Experience [✓], Reward [✓], Items [✓]

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

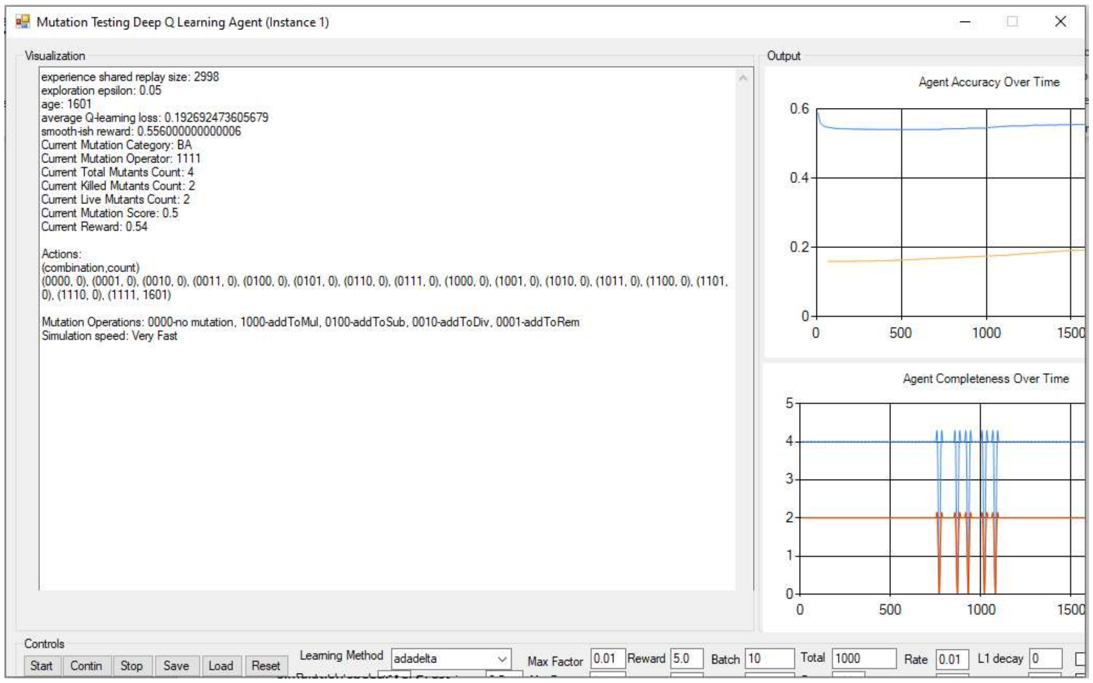
START(year-month-day_hours:mins:secs):2021-06-09_11:39:33 | END(year-month-day_hours:mins:secs):2021-06-10_15:34:15 | ELAPSED(day.hours_mins:secs):01.03_54:42


Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2998
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.194291956470205
smooth-ish reward: 0.556000000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

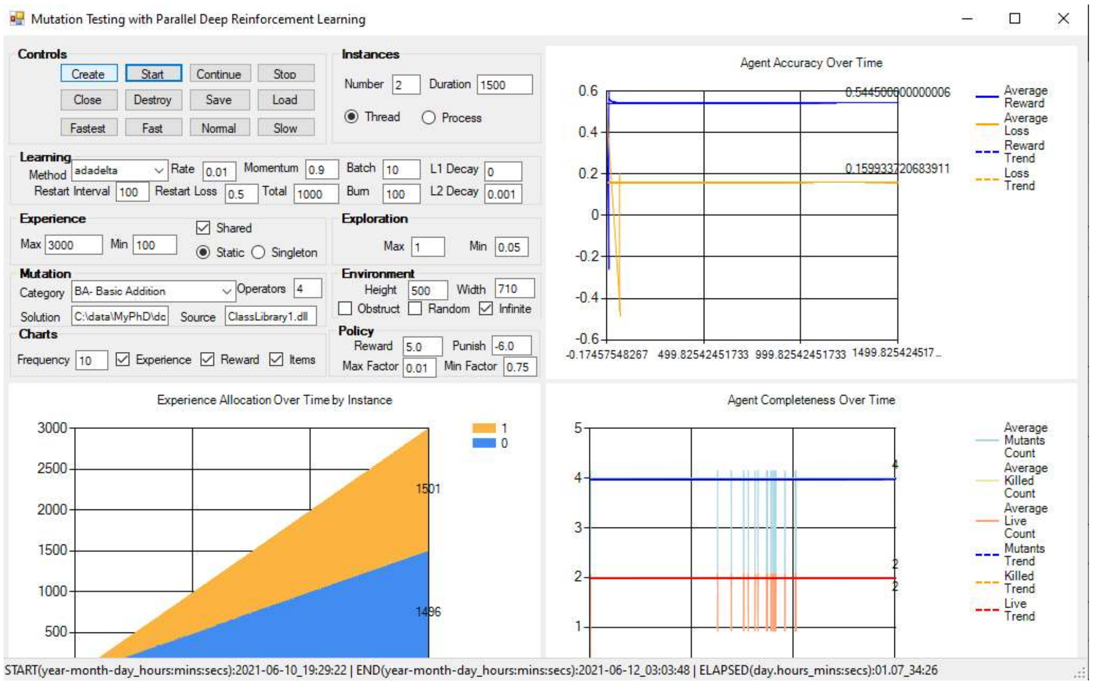
Controls: Start, Contin, Stop, Save, Load, Reset, Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0



```
experience shared replay size: 2998
exploration epsilon: 0.05
age: 1601
average Q-learning loss: 0.192692473605679
smooth-ish reward: 0.556000000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1601)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

2



Controls

Instances
Number 2 Duration 1500
[✓ Thread] Process

Learning
Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0
Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience
Max 3000 Min 100 [✓] Shared
[✓ Static] Singleton

Exploration
Max 1 Min 0.05

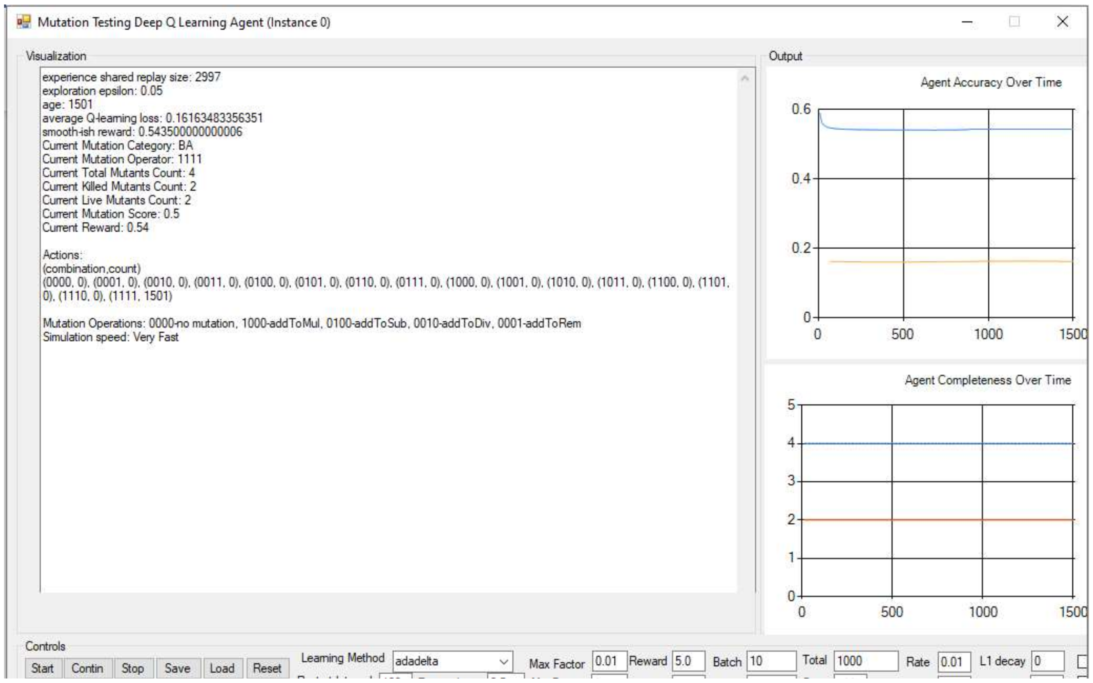
Mutation
Category BA- Basic Addition Operators 4
Solution C:\data\MyPhD\dc Source ClassLibrary1.dll

Environment
Height 500 Width 710
[ ] Obstruct [ ] Random [✓] Infinite

Charts
Frequency 10 [✓] Experience [✓] Reward [✓] Items

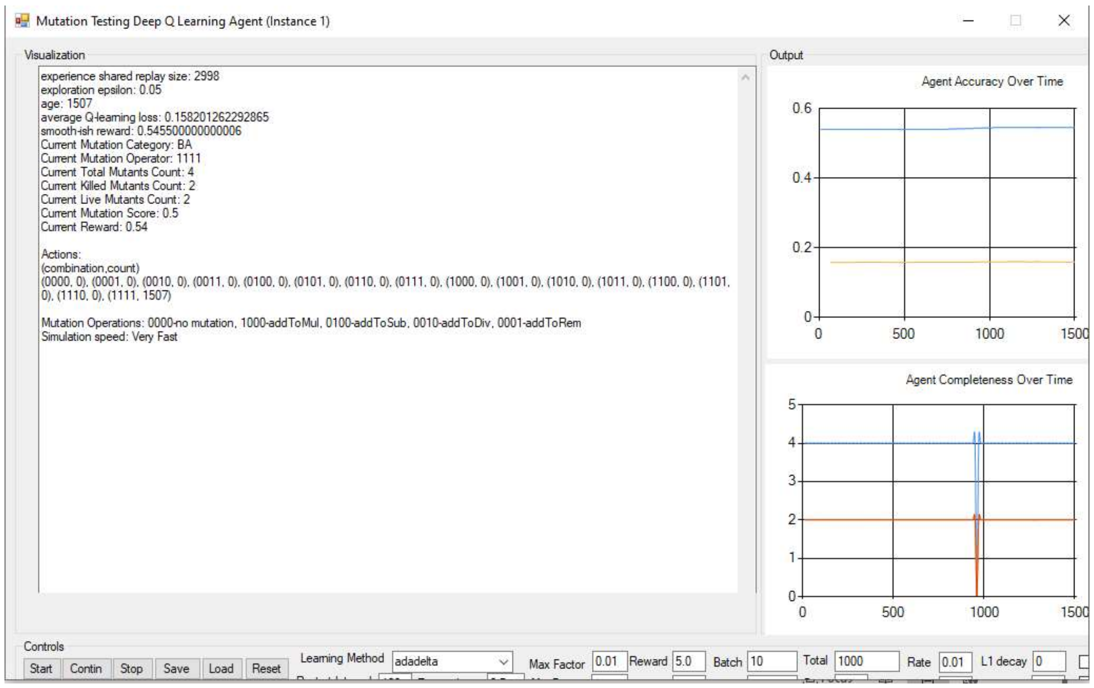
Policy
Reward 5.0 Punish -6.0
Max Factor 0.01 Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-10_19:29:22 | END(year-month-day_hours:mins:secs):2021-06-12_03:03:48 | ELAPSED(day.hours_mins:secs):01.07_34:26


Mutation Testing Deep Q Learning Agent (Instance 0)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2997
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.16163483356351
smooth-ish reward: 0.543500000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

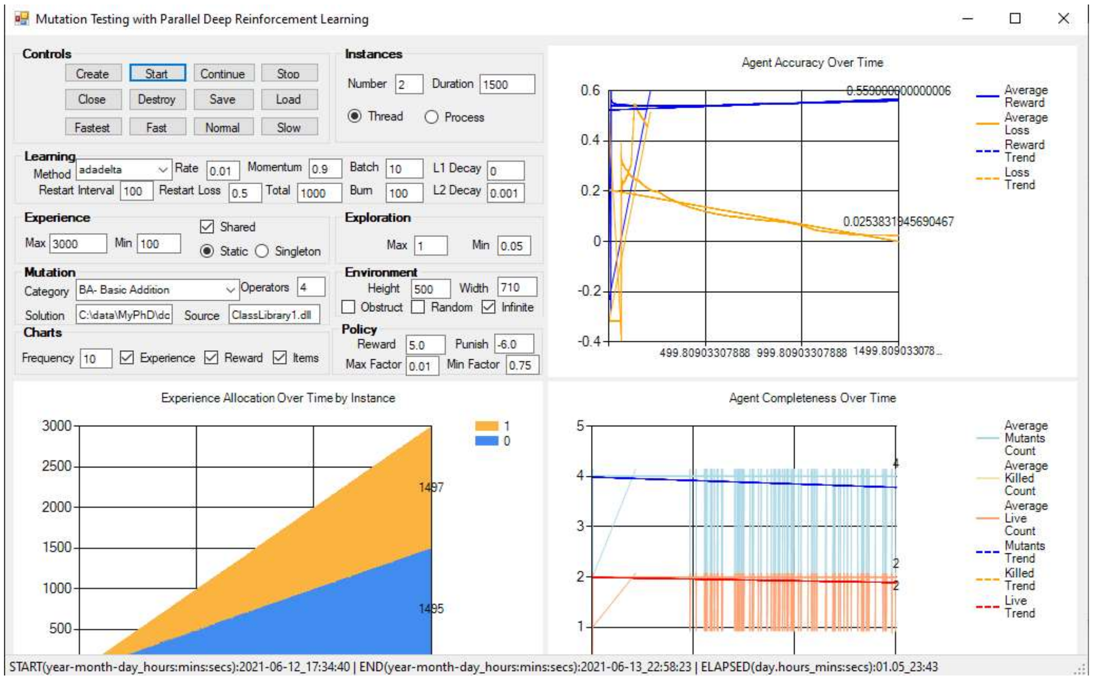
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0



```
experience shared replay size: 2998
exploration epsilon: 0.05
age: 1507
average Q-learning loss: 0.158201262292865
smooth-ish reward: 0.545500000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1507)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

3



Mutation Testing with Parallel Deep Reinforcement Learning

Controls

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ✓ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

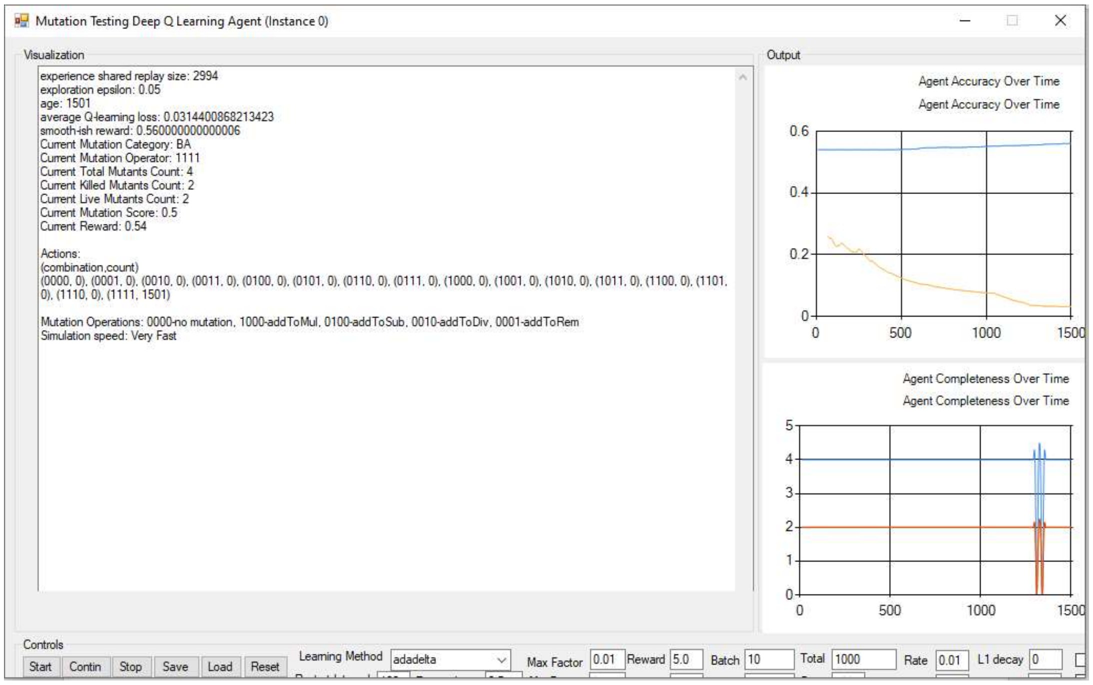
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ✓ Infinite

Charts: Frequency 10, ✓ Experience, ✓ Reward, ✓ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-12_17:34:40 | END(year-month-day_hours:mins:secs):2021-06-13_22:58:23 | ELAPSED(day.hours_mins:secs):01.05_23:43

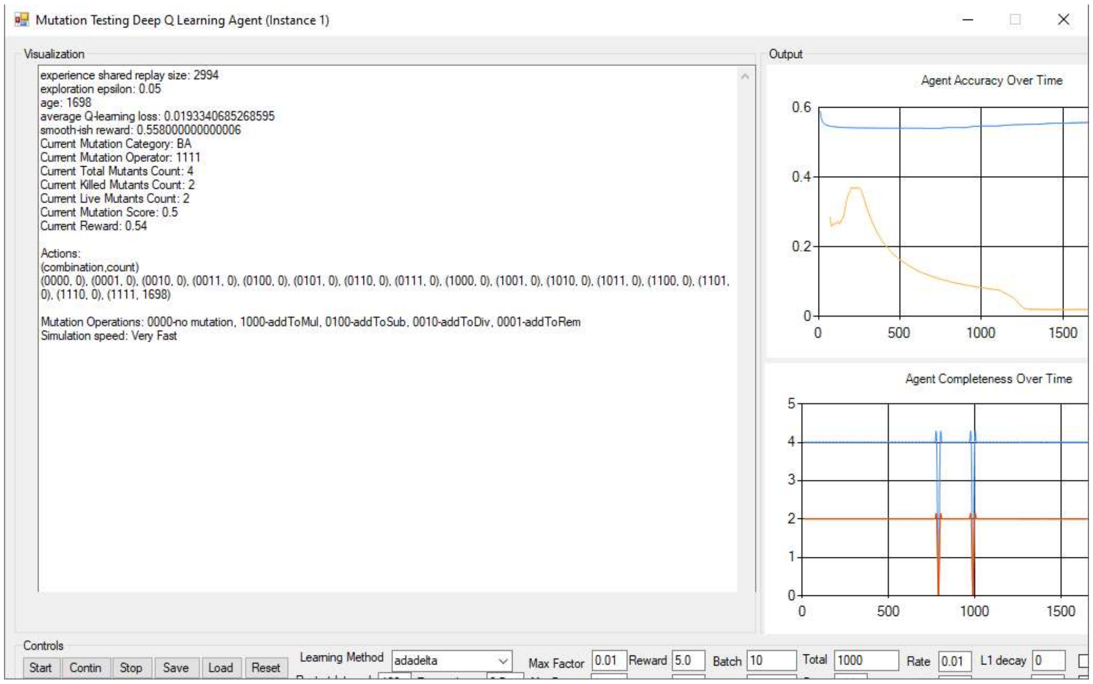


Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2994
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0314400868213423
smooth-ish reward: 0.560000000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

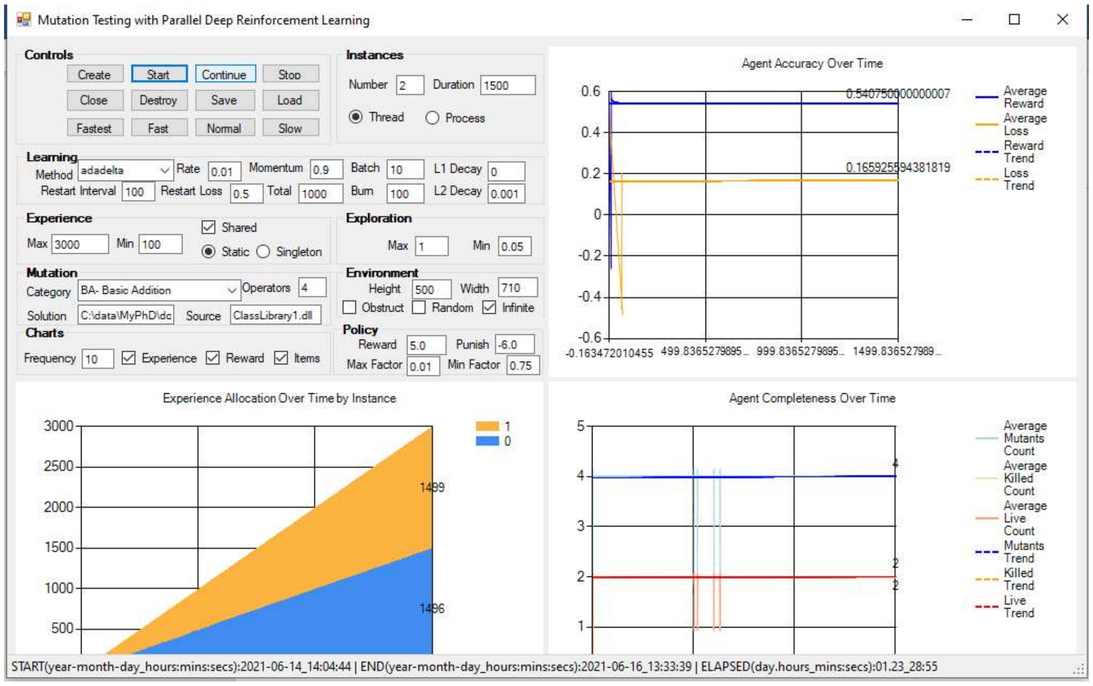
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0


Mutation Testing Deep Q Learning Agent (Instance 1)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2994
exploration epsilon: 0.05
age: 1698
average Q-learning loss: 0.0193340685268595
smooth-ish reward: 0.558000000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1698)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0

4



Instances

Number 2 Duration 1500

(● Thread) ○ Process

Learning

Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0

Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience

Max 3000 Min 100 ☑ Shared (● Static) ○ Singleton

Exploration

Max 1 Min 0.05

Mutation

Category BA- Basic Addition Operators 4

Solution C:\data\MyPhD\dc Source ClassLibrary1.dll

Environment

Height 500 Width 710 ☐ Obstruct ☐ Random ☑ Infinite

Charts

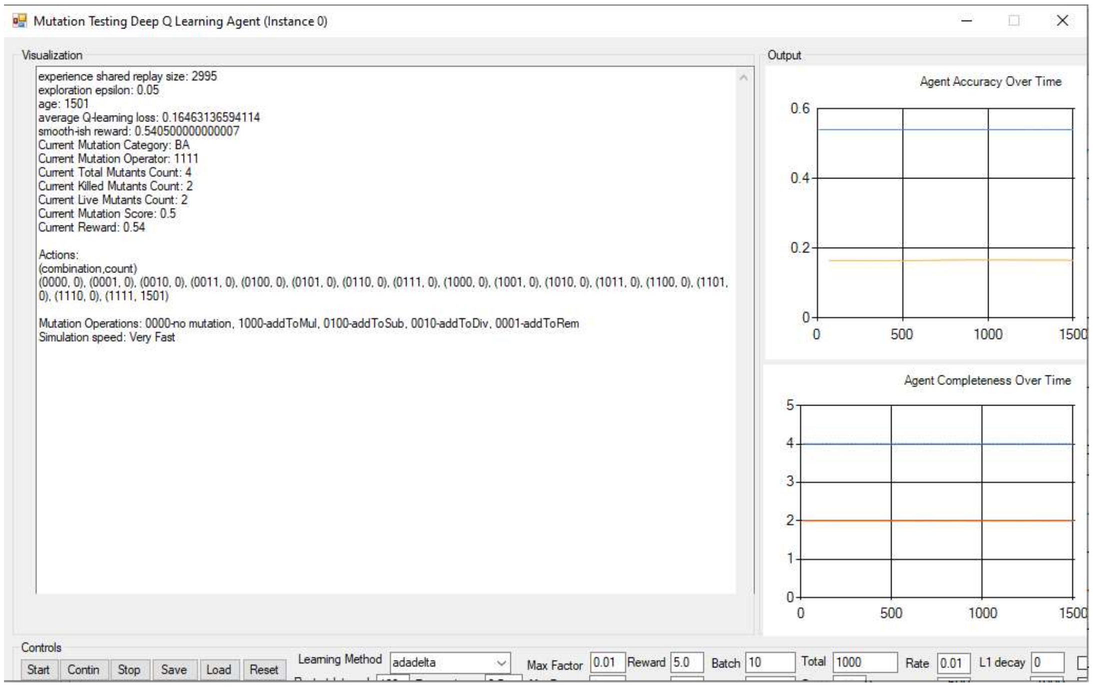
Frequency 10 ☑ Experience ☑ Reward ☑ Items

Policy

Reward 5.0 Punish -6.0

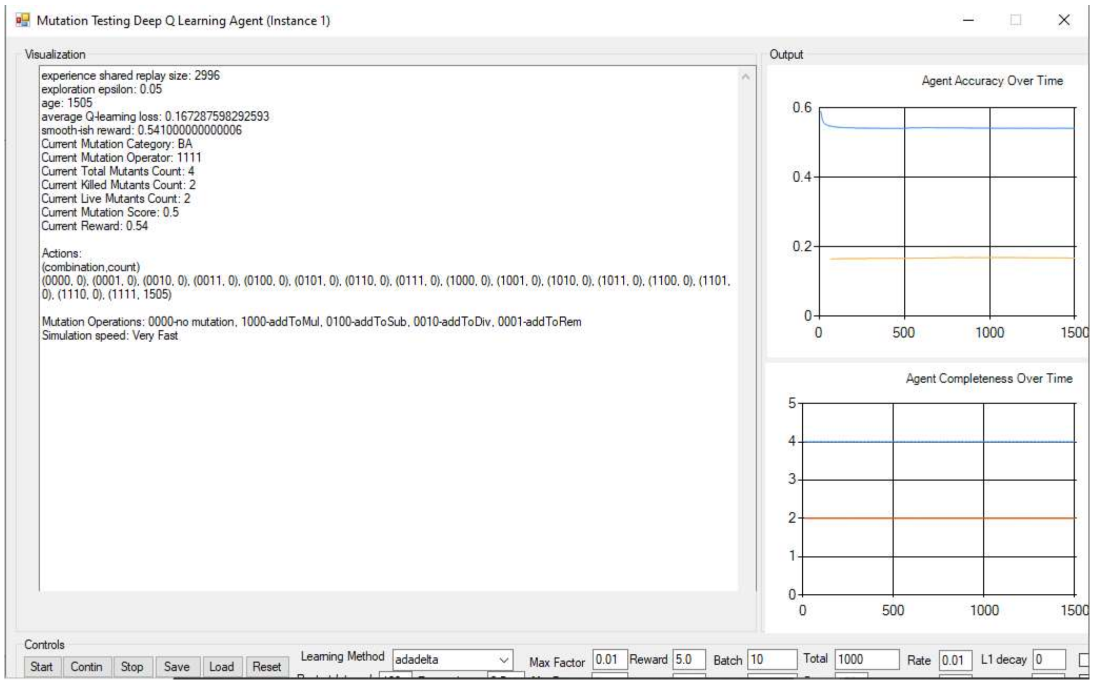
Max Factor 0.01 Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-14_14:04:44 | END(year-month-day_hours:mins:secs):2021-06-16_13:33:39 | ELAPSED(day.hours_mins:secs):01.23_28:55


Mutation Testing Deep Q Learning Agent (Instance 0)
Visualization
Output
Agent Accuracy Over Time
Agent Completeness Over Time
Controls
Start
Contin
Stop
Save
Load
Reset

```
experience shared replay size: 2995
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.16463136594114
smooth-ish reward: 0.540500000000007
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

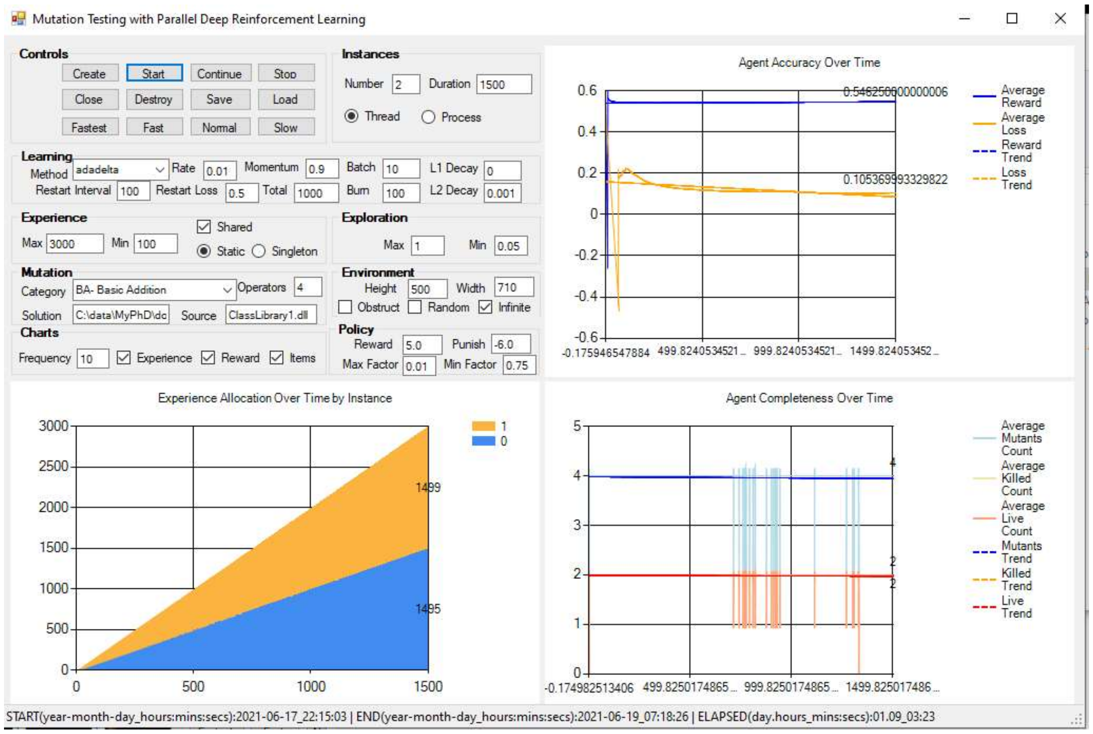
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0



```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1505
average Q-learning loss: 0.167287598292593
smooth-ish reward: 0.541000000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1505)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta | Max Factor: 0.01 | Reward: 5.0 | Batch: 10 | Total: 1000 | Rate: 0.01 | L1 decay: 0

5



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ✓ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

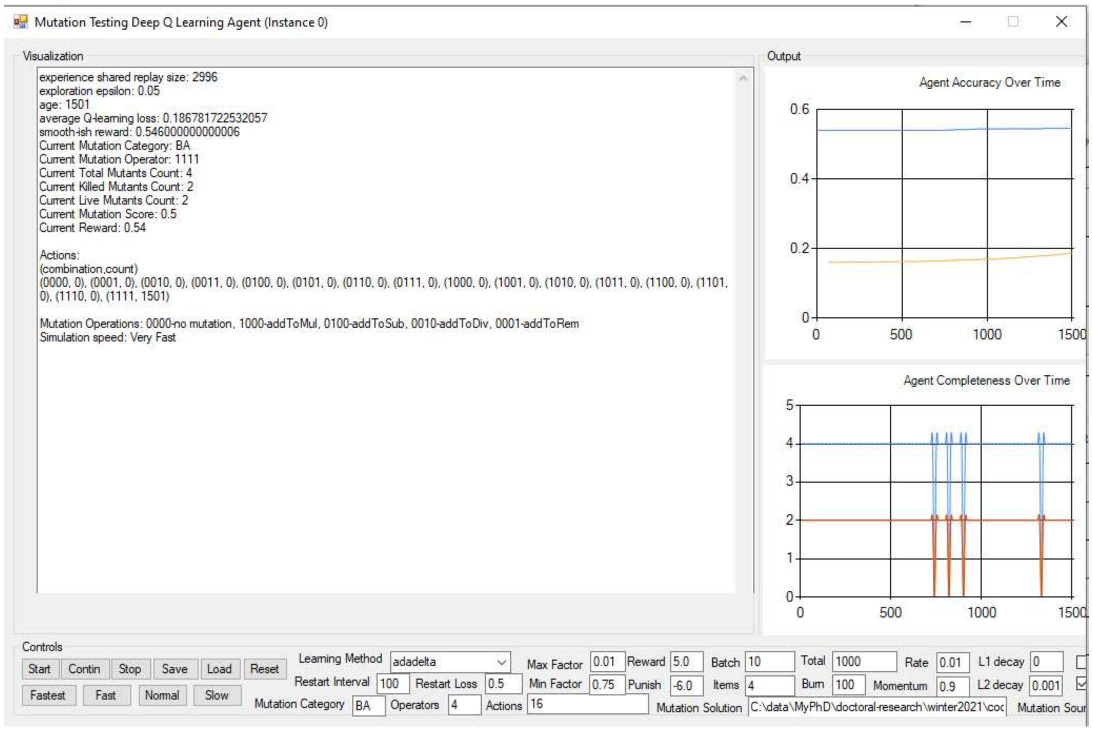
Mutation: Category BA- Basic Addition, Operators 4, Solution C:\data\MyPhD\dc, Source ClassLibrary1.dll

Environment: Height 500, Width 710, Obstruct, Random, ✓ Infinite

Charts: Frequency 10, ✓ Experience, ✓ Reward, ✓ Items

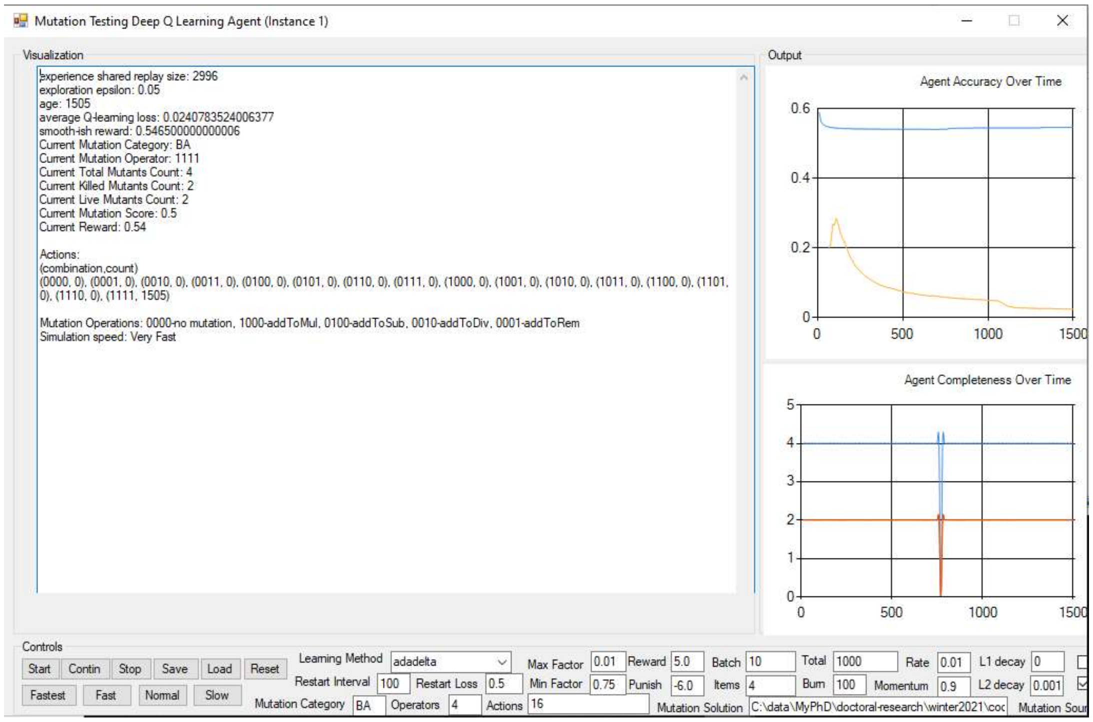
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-17_22:15:03 | END(year-month-day_hours:mins:secs):2021-06-19_07:18:26 | ELAPSED(day.hours_mins:secs):01.09_03:23



```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.186781722532057
smooth-ish reward: 0.546000000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0
Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001
Mutation Category BA, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Sour

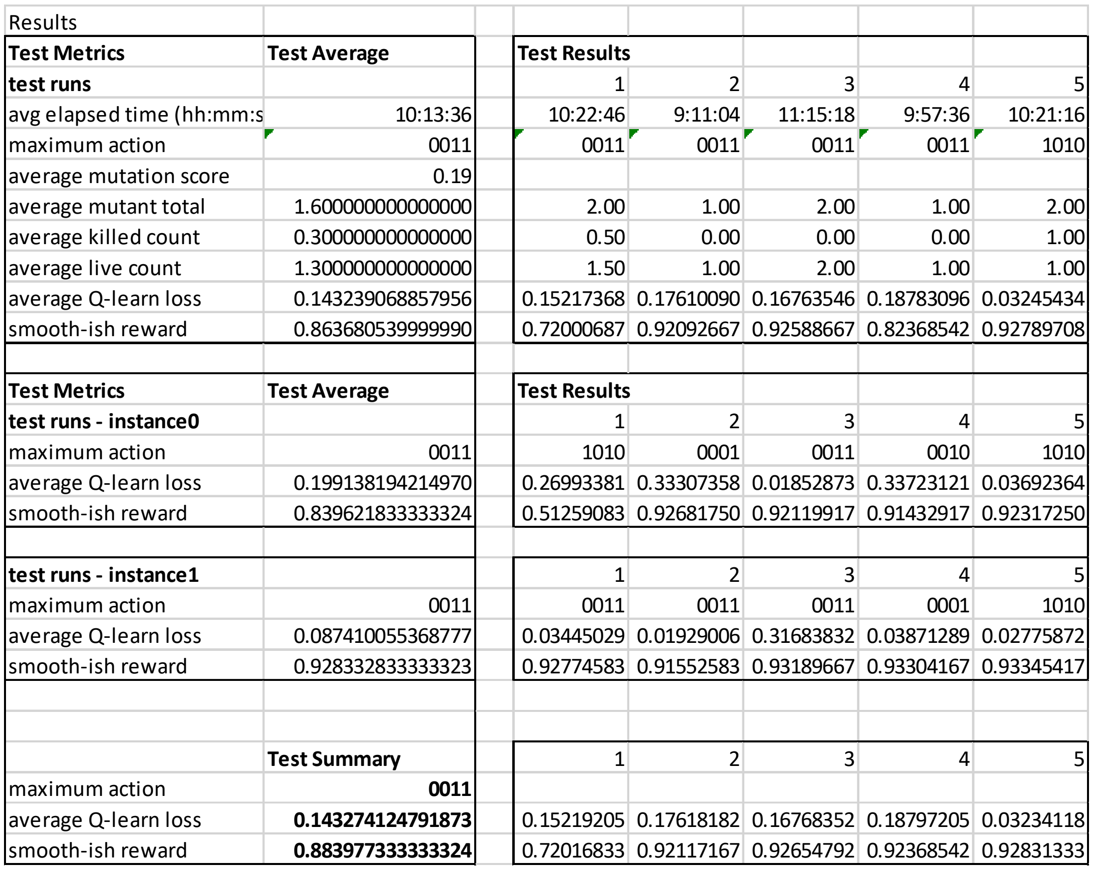


```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1505
average Q-learning loss: 0.0240783524006377
smooth-ish reward: 0.546500000000006
Current Mutation Category: BA
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1505)

Mutation Operations: 0000-no mutation, 1000-addToMul, 0100-addToSub, 0010-addToDiv, 0001-addToRem
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Category: BA
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Sour

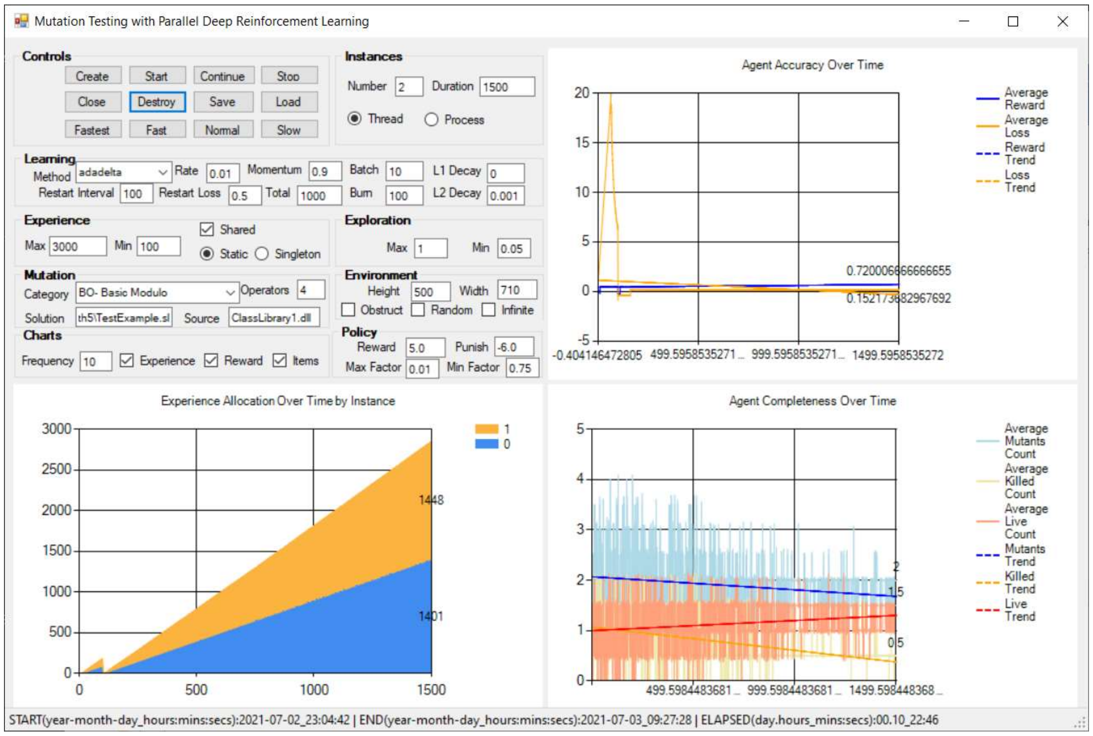
Experiment 5: Comparing Mutation Testing Approaches with Four Cores

Machine Learning selection of mutation operators:



Results						
Test Metrics	**Test Average**	**Test Results**				
test runs		1	2	3	4	5
avg elapsed time (hh:mm:s	10:13:36	10:22:46	9:11:04	11:15:18	9:57:36	10:21:16
maximum action	0011	0011	0011	0011	0011	1010
average mutation score	0.19					
average mutant total	1.600000000000000	2.00	1.00	2.00	1.00	2.00
average killed count	0.300000000000000	0.50	0.00	0.00	0.00	1.00
average live count	1.300000000000000	1.50	1.00	2.00	1.00	1.00
average Q-learn loss	0.143239068857956	0.15217368	0.17610090	0.16763546	0.18783096	0.03245434
smooth-ish reward	0.863680539999990	0.72000687	0.92092667	0.92588667	0.82368542	0.92789708
Test Metrics	**Test Average**	**Test Results**				
test runs - instance0		1	2	3	4	5
maximum action	0011	1010	0001	0011	0010	1010
average Q-learn loss	0.199138194214970	0.26993381	0.33307358	0.01852873	0.33723121	0.03692364
smooth-ish reward	0.839621833333324	0.51259083	0.92681750	0.92119917	0.91432917	0.92317250
test runs - instance1		1	2	3	4	5
maximum action	0011	0011	0011	0011	0001	1010
average Q-learn loss	0.087410055368777	0.03445029	0.01929006	0.31683832	0.03871289	0.02775872
smooth-ish reward	0.928332833333323	0.92774583	0.91552583	0.93189667	0.93304167	0.93345417
	Test Summary	1	2	3	4	5
maximum action	**0011**					
average Q-learn loss	**0.143274124791873**	0.15219205	0.17618182	0.16768352	0.18797205	0.03234118
smooth-ish reward	**0.883977333333324**	0.72016833	0.92117167	0.92654792	0.92368542	0.92831333

1



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

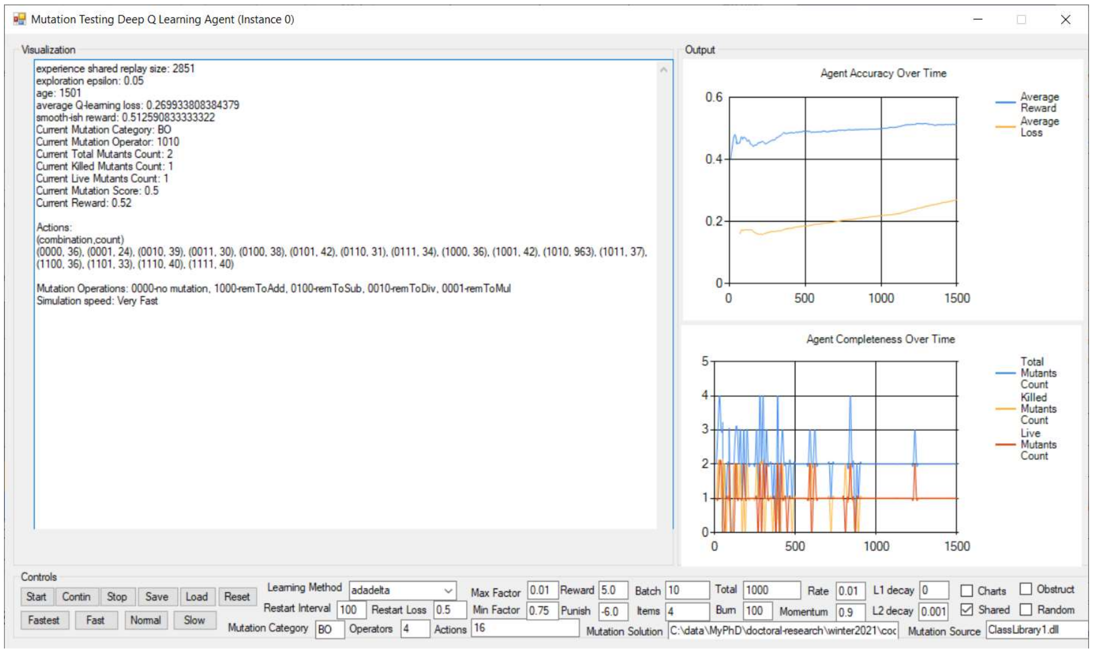
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

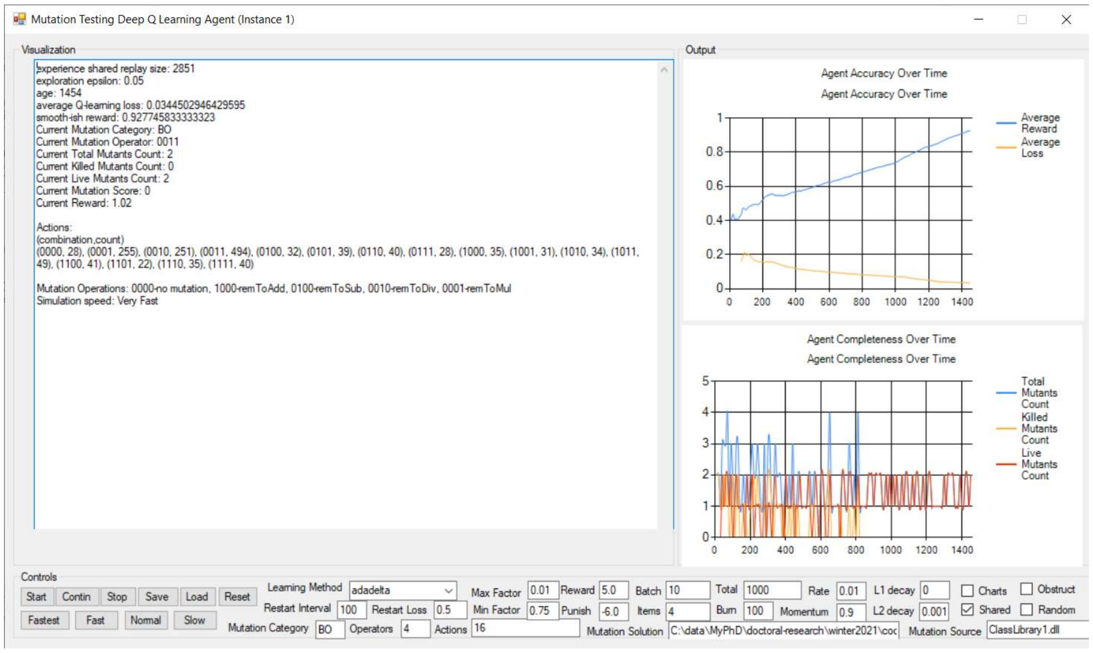
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

```
START(year-month-day_hours:mins:secs):2021-07-02_23:04:42 | END(year-month-day_hours:mins:secs):2021-07-03_09:27:28 | ELAPSED(day.hours_mins:secs):00.10_22:46
```



```
experience shared replay size: 2851
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.269933808384379
smooth-ish reward: 0.512590833333322
Current Mutation Category: BO
Current Mutation Operator: 1010
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 36), (0001, 24), (0010, 39), (0011, 30), (0100, 38), (0101, 42), (0110, 31), (0111, 34), (1000, 36), (1001, 42), (1010, 963), (1011, 37),
(1100, 36), (1101, 33), (1110, 40), (1111, 40)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Mutation Category BO, Operators 4, Actions 16, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, ☐ Charts, ☑ Shared, ☐ Obstruct, ☐ Random, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



Mutation Testing Deep Q Learning Agent (Instance 1)

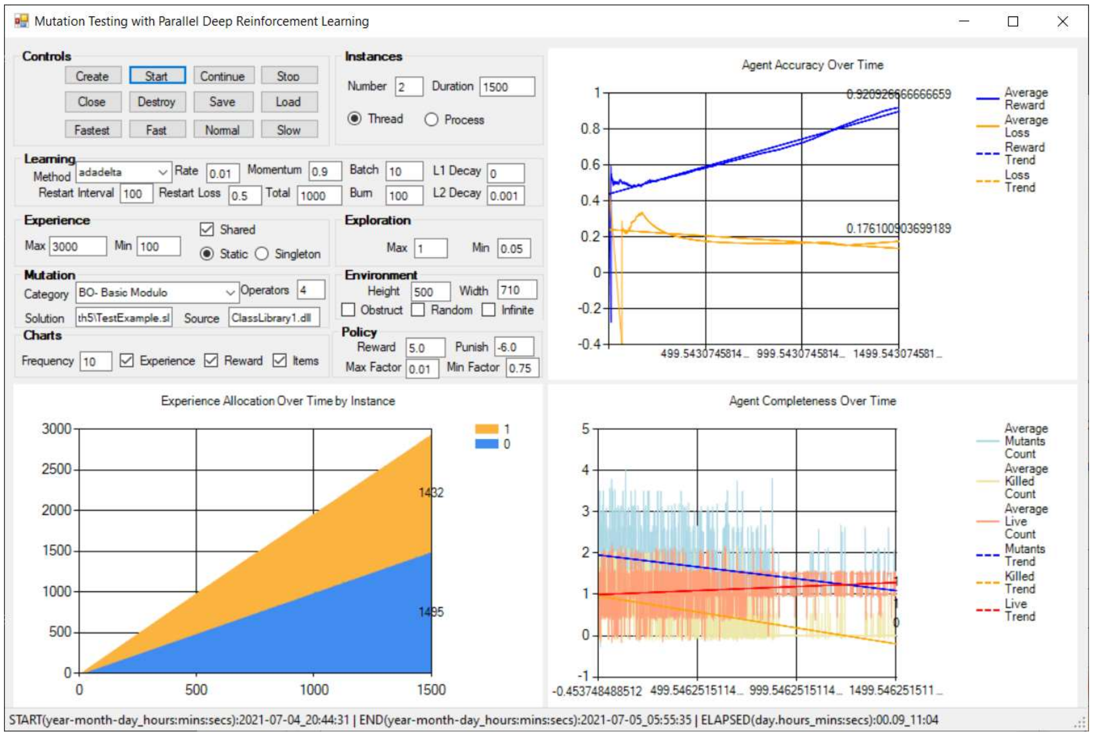
Visualization

```
experience shared replay size: 2851
exploration epsilon: 0.05
age: 1454
average Q-learning loss: 0.0344502946429595
smooth-ish reward: 0.927745833333323
Current Mutation Category: BO
Current Mutation Operator: 0011
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 28), (0001, 255), (0010, 251), (0011, 494), (0100, 32), (0101, 39), (0110, 40), (0111, 28), (1000, 35), (1001, 31), (1010, 34), (1011,
49), (1100, 41), (1101, 22), (1110, 35), (1111, 40)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Controls

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0
Restart Interval: 100
Restart Loss: 0.5
Min Factor: 0.75
Punish: -6.0
Items: 4
Burn: 100
Momentum: 0.9
L2 decay: 0.001
Charts [ ] Obstruct [ ] Shared [✓] Random [ ]
Mutation Category: BO
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

2



Controls

Instances Number 2 Duration 1500 [✓ Thread] Process

Learning Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0
Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience Max 3000 Min 100 ☑ Shared [✓ Static] Singleton

Exploration Max 1 Min 0.05

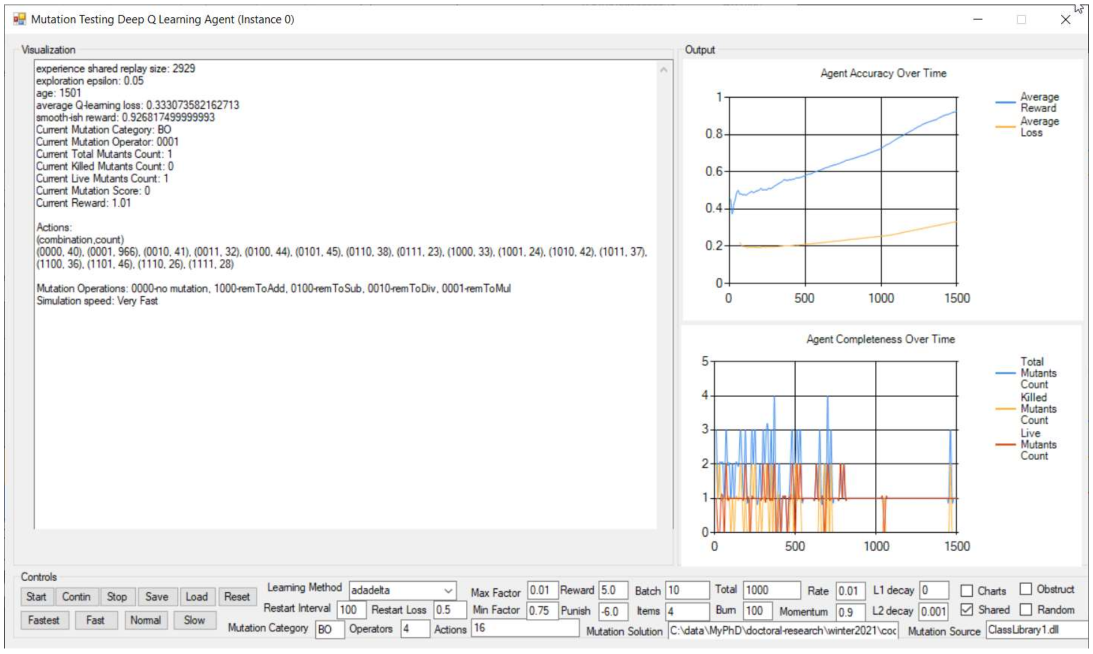
Mutation Category BO- Basic Modulo Operators 4
Solution th5\TestExample.sl Source ClassLibrary1.dll

Environment Height 500 Width 710 ☐ Obstruct ☐ Random ☐ Infinite

Charts Frequency 10 ☑ Experience ☑ Reward ☑ Items

Policy Reward 5.0 Punish -6.0 Max Factor 0.01 Min Factor 0.75

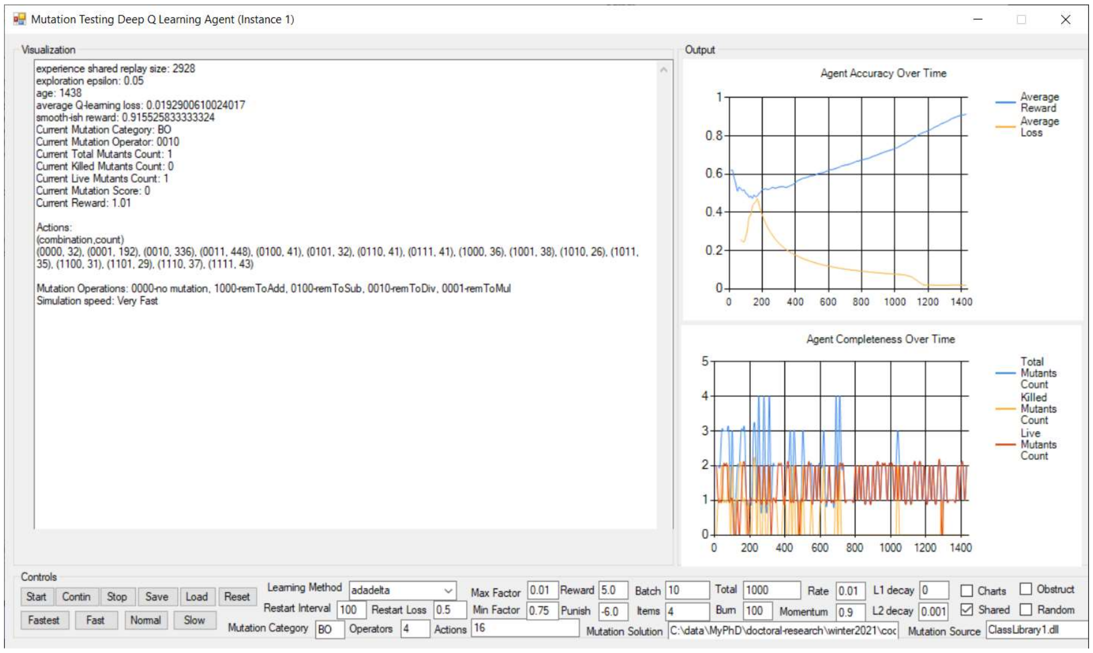
START(year-month-day_hours:mins:secs):2021-07-04_20:44:31 | END(year-month-day_hours:mins:secs):2021-07-05_05:55:35 | ELAPSED(day.hours_mins:secs):00.09_11:04



```
experience shared replay size: 2929
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.333073582162713
smooth-ish reward: 0.926817499999993
Current Mutation Category: BO
Current Mutation Operator: 0001
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 40), (0001, 966), (0010, 41), (0011, 32), (0100, 44), (0101, 45), (0110, 38), (0111, 23), (1000, 33), (1001, 24), (1010, 42), (1011, 37),
(1100, 36), (1101, 46), (1110, 26), (1111, 28)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Controls

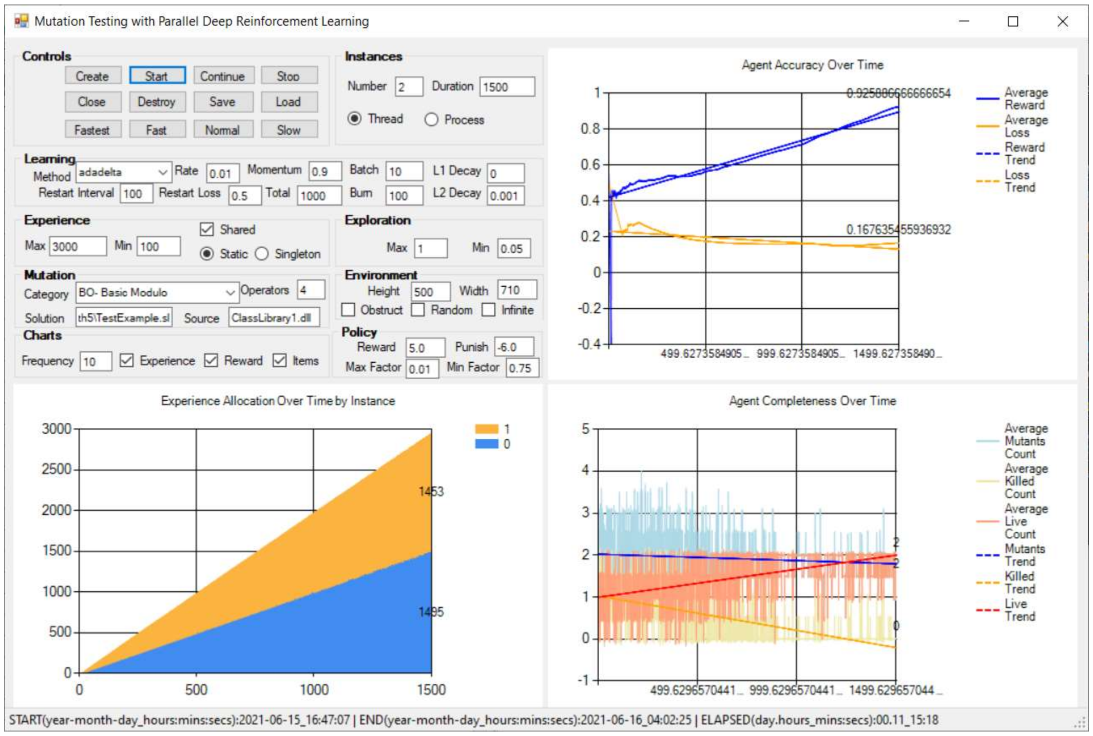
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0 ☐ Charts ☐ Obstruct
Restart Interval 100 Restart Loss 0.5 Min Factor 0.75 Punish -6.0 Items 4 Burn 100 Momentum 0.9 L2 decay 0.001 ☑ Shared ☐ Random
Mutation Category BO Operators 4 Actions 16 Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2928
exploration epsilon: 0.05
age: 1438
average Q-learning loss: 0.0192900610024017
smooth-ish reward: 0.915525833333324
Current Mutation Category: BO
Current Mutation Operator: 0010
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 32), (0001, 192), (0010, 336), (0011, 448), (0100, 41), (0101, 32), (0110, 41), (0111, 41), (1000, 36), (1001, 38), (1010, 26), (1011,
35), (1100, 31), (1101, 29), (1110, 37), (1111, 43)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
☐ Charts
[✓ Shared]
☐ Obstruct
☐ Random
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

3



Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

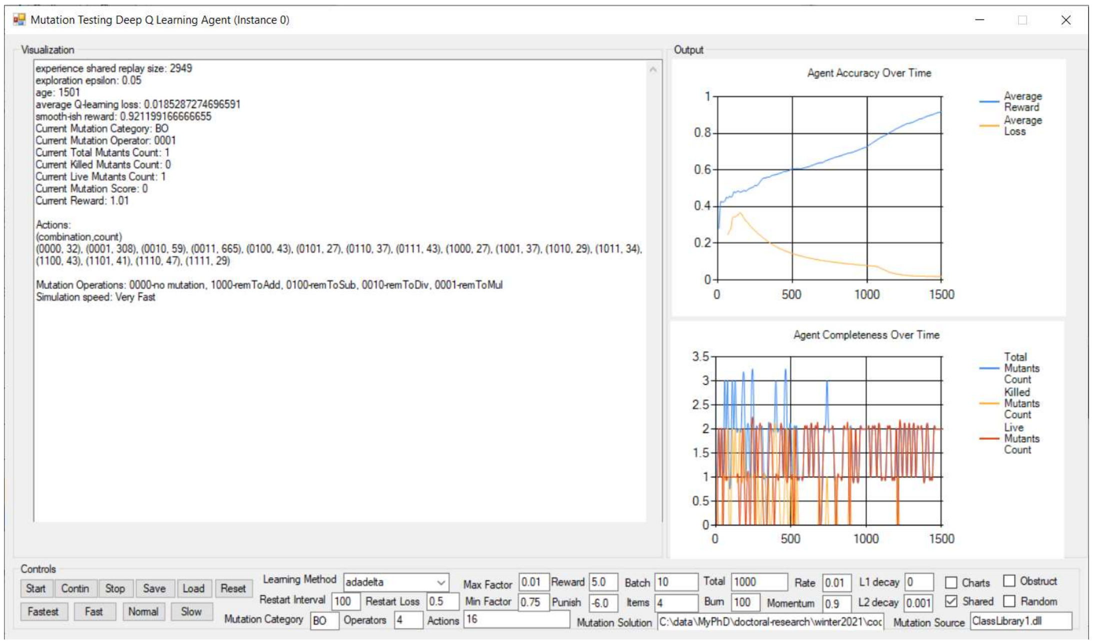
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-15_16:47:07 | END(year-month-day_hours:mins:secs):2021-06-16_04:02:25 | ELAPSED(day.hours_mins:secs):00.11_15:18



Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2949
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0185287274696591
smooth-ish reward: 0.921199166666655
Current Mutation Category: BO
Current Mutation Operator: 0001
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 32), (0001, 308), (0010, 59), (0011, 665), (0100, 43), (0101, 27), (0110, 37), (0111, 43), (1000, 27), (1001, 37), (1010, 29), (1011, 34),
(1100, 43), (1101, 41), (1110, 47), (1111, 29)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

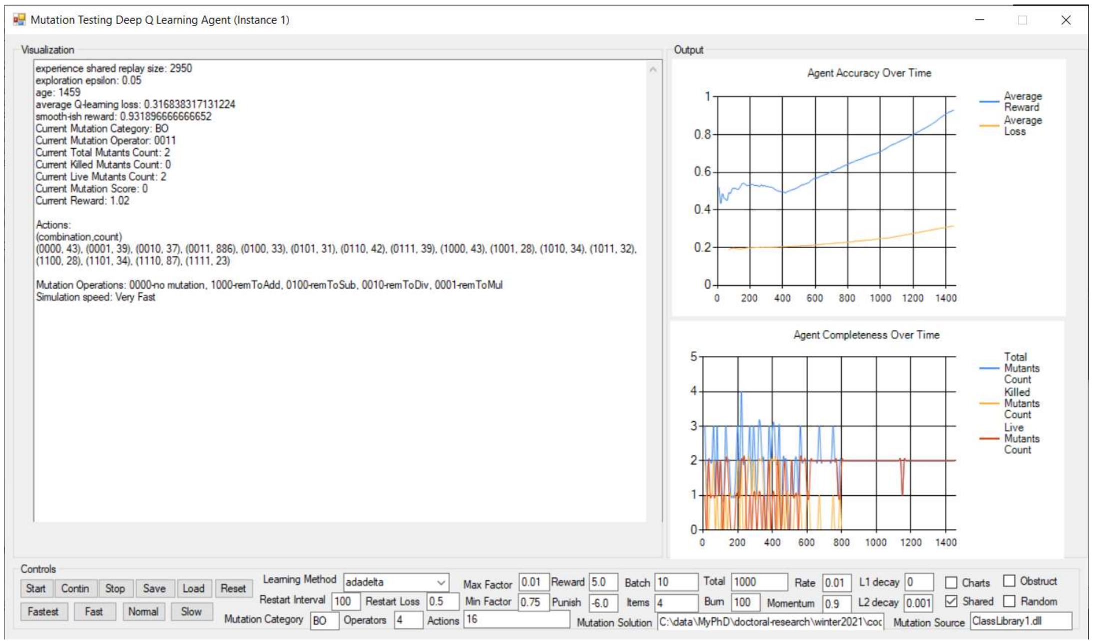
Output

Controls: Start, Contin, Stop, Save, Load, Reset, Fastest, Fast, Normal, Slow

Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, ☐ Charts, ☐ Obstruct

Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, ☑ Shared, ☐ Random

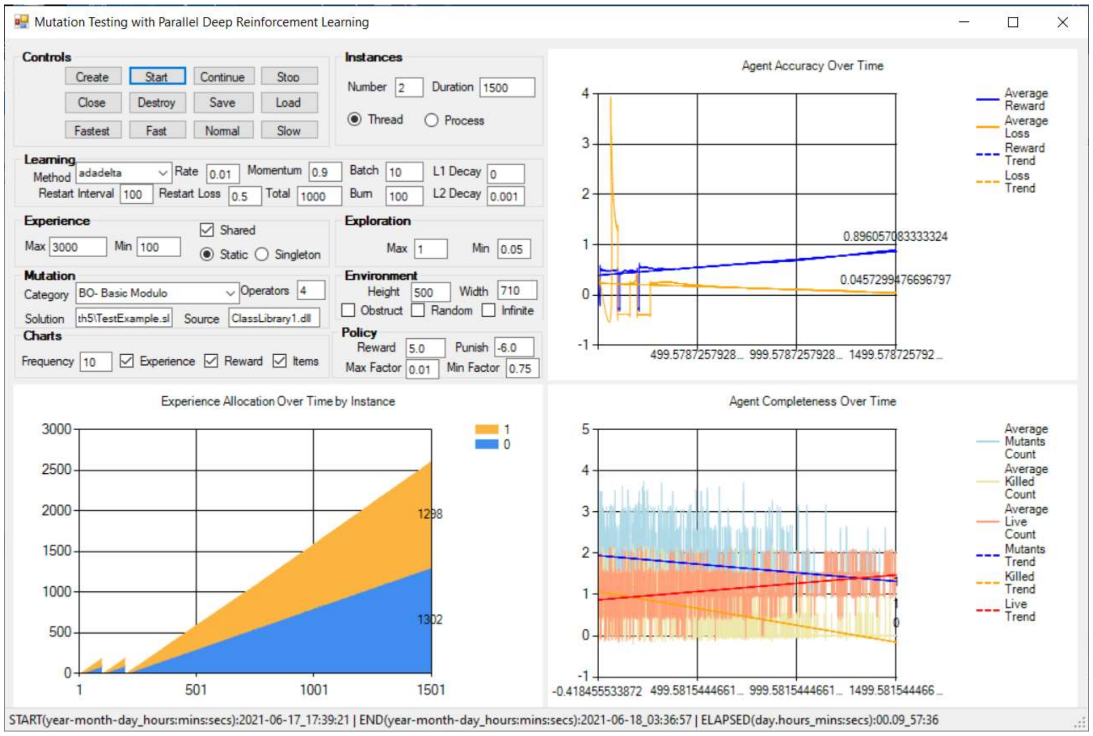
Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2950
exploration epsilon: 0.05
age: 1459
average Q-learning loss: 0.316838317131224
smooth-ish reward: 0.931896666666652
Current Mutation Category: BO
Current Mutation Operator: 0011
Current Total Mutants Count: 2
Current Killed Mutants Count: 0
Current Live Mutants Count: 2
Current Mutation Score: 0
Current Reward: 1.02

Actions:
(combination,count)
(0000, 43), (0001, 39), (0010, 37), (0011, 886), (0100, 33), (0101, 31), (0110, 42), (0111, 39), (1000, 43), (1001, 28), (1010, 34), (1011, 32),
(1100, 28), (1101, 34), (1110, 87), (1111, 23)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll
[ ] Charts [✓ Shared] [ ] Obstruct [ ] Random

4



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, [✓ Shared], [✓ Static] Singleton

Exploration: Max 1, Min 0.05

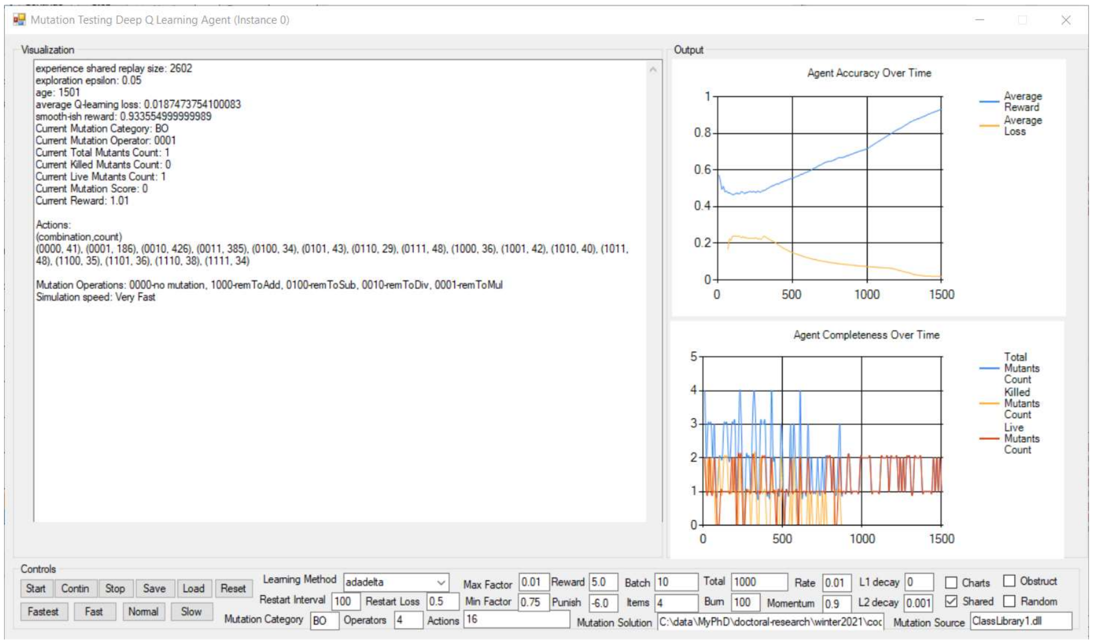
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, Obstruct, Random, Infinite

Charts: Frequency 10, [✓ Experience], [✓ Reward], [✓ Items]

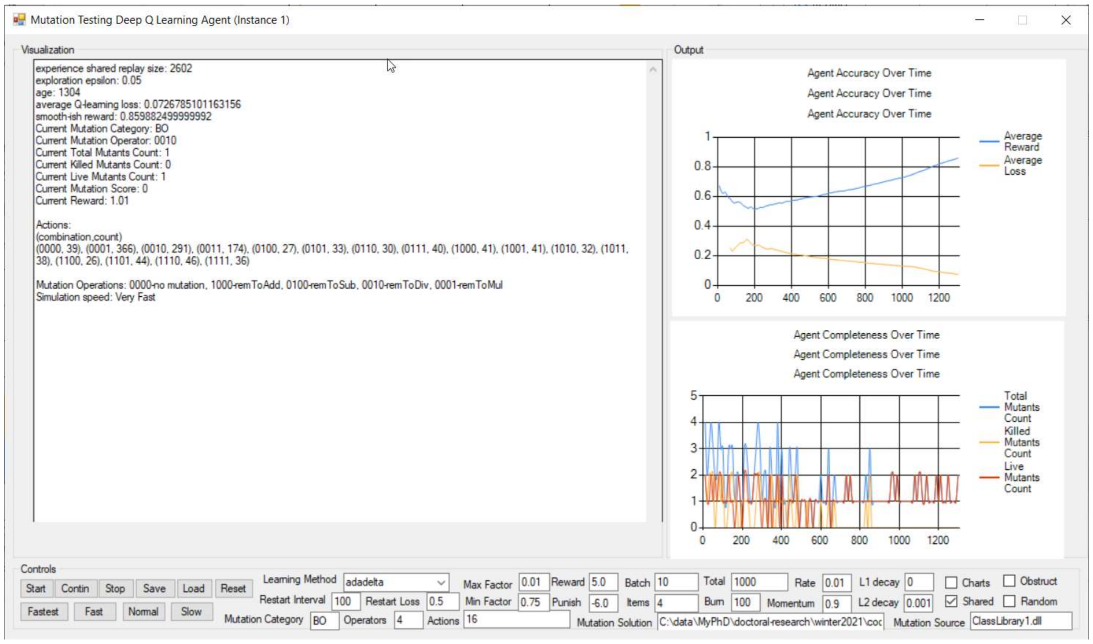
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-17_17:39:21 | END(year-month-day_hours:mins:secs):2021-06-18_03:36:57 | ELAPSED(day.hours_mins:secs):00.09_57:36



```
experience shared replay size: 2602
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0187473754100083
smooth-ish reward: 0.933554999999989
Current Mutation Category: BO
Current Mutation Operator: 0001
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 41), (0001, 186), (0010, 426), (0011, 385), (0100, 34), (0101, 43), (0110, 29), (0111, 48), (1000, 36), (1001, 42), (1010, 40), (1011,
48), (1100, 35), (1101, 36), (1110, 38), (1111, 34)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

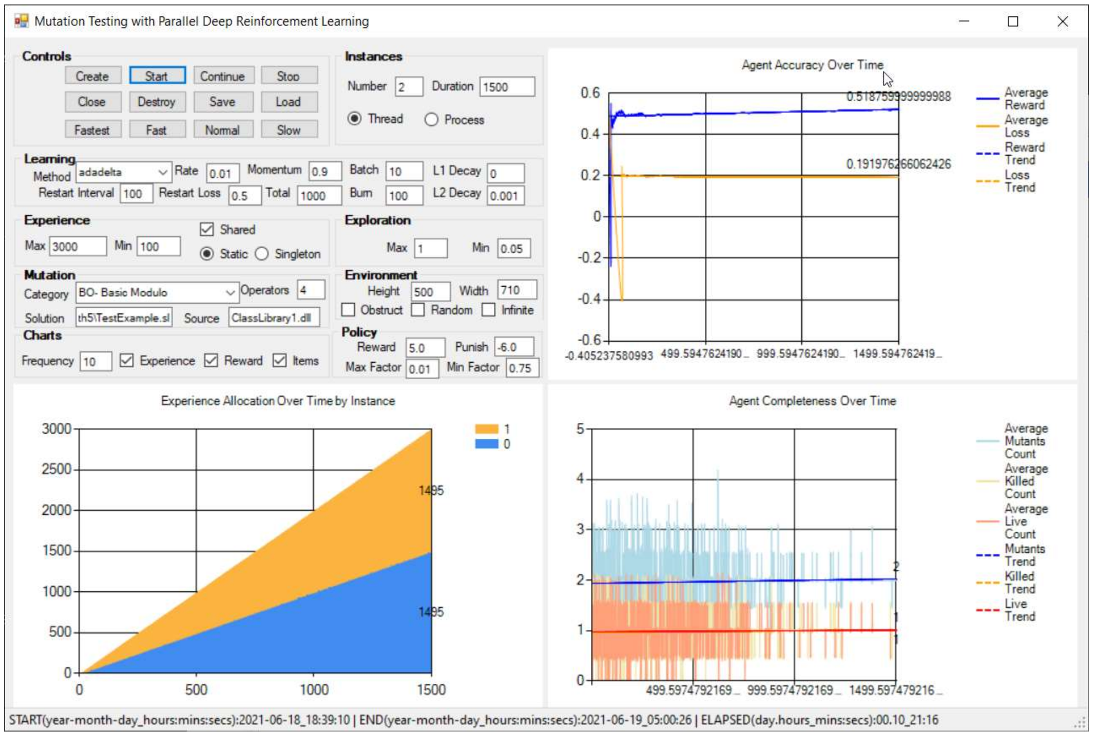
Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, Charts, Obstruct
Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, [✓ Shared], Random
Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2602
exploration epsilon: 0.05
age: 1304
average Q-learning loss: 0.0726785101163156
smooth-ish reward: 0.859882499999992
Current Mutation Category: BO
Current Mutation Operator: 0010
Current Total Mutants Count: 1
Current Killed Mutants Count: 0
Current Live Mutants Count: 1
Current Mutation Score: 0
Current Reward: 1.01

Actions:
(combination,count)
(0000, 39), (0001, 366), (0010, 291), (0011, 174), (0100, 27), (0101, 33), (0110, 30), (0111, 40), (1000, 41), (1001, 41), (1010, 32), (1011,
38), (1100, 26), (1101, 44), (1110, 46), (1111, 36)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta; Restart Interval: 100; Restart Loss: 0.5; Max Factor: 0.01; Min Factor: 0.75; Reward: 5.0; Punish: -6.0; Batch: 10; Items: 4; Total: 1000; Burn: 100; Rate: 0.01; Momentum: 0.9; L1 decay: 0; L2 decay: 0.001; [ ] Charts; [✓] Shared; [ ] Obstruct; [ ] Random; Mutation Category: BO; Operators: 4; Actions: 16; Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc; Mutation Source: ClassLibrary1.dll

5



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

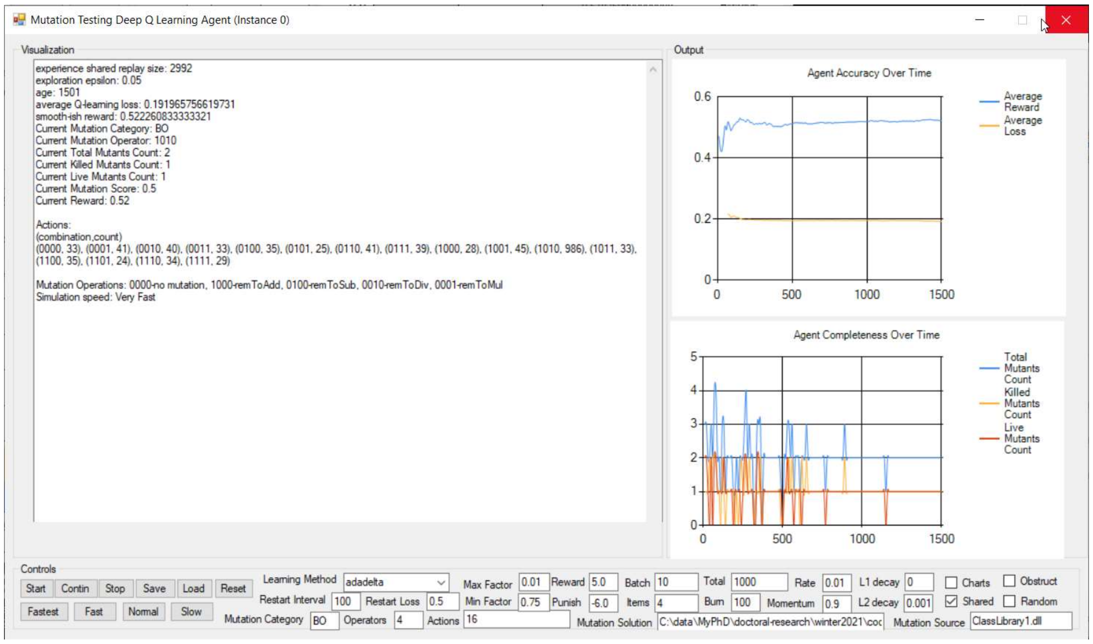
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

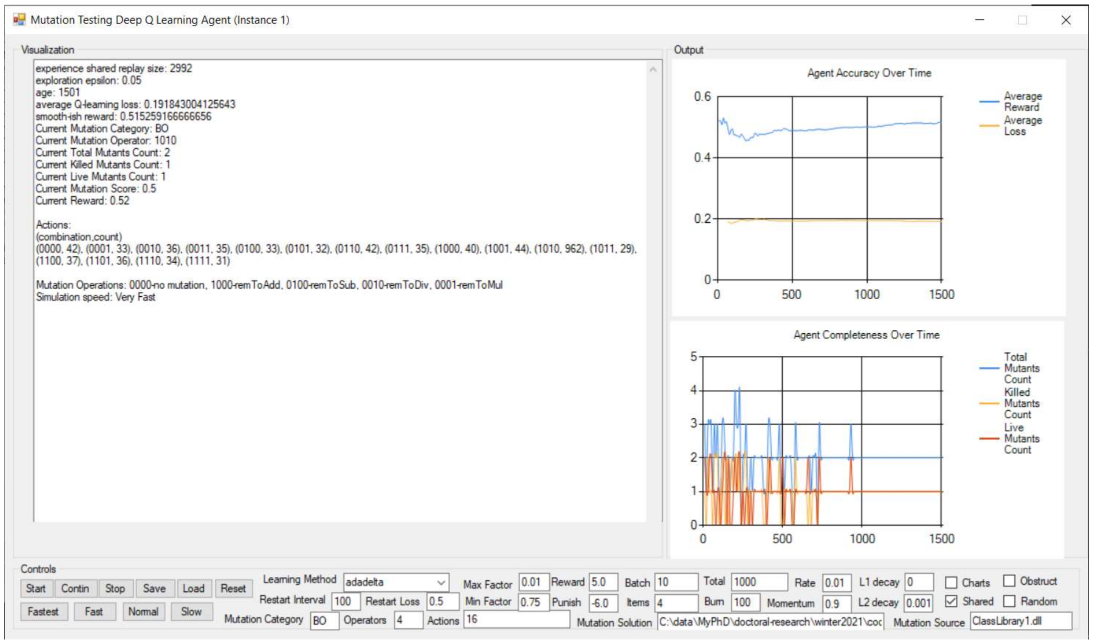
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-18_18:39:10 | END(year-month-day_hours:mins:secs):2021-06-19_05:00:26 | ELAPSED(day.hours_mins:secs):00.10_21:16



```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.191965756619731
smooth-ish reward: 0.522260833333321
Current Mutation Category: BO
Current Mutation Operator: 1010
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 33), (0001, 41), (0010, 40), (0011, 33), (0100, 35), (0101, 25), (0110, 41), (0111, 39), (1000, 28), (1001, 45), (1010, 986), (1011, 33),
(1100, 35), (1101, 24), (1110, 34), (1111, 29)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Mutation Category BO, Operators 4, Actions 16, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, ☐ Charts, ☑ Shared, ☐ Obstruct, ☐ Random, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



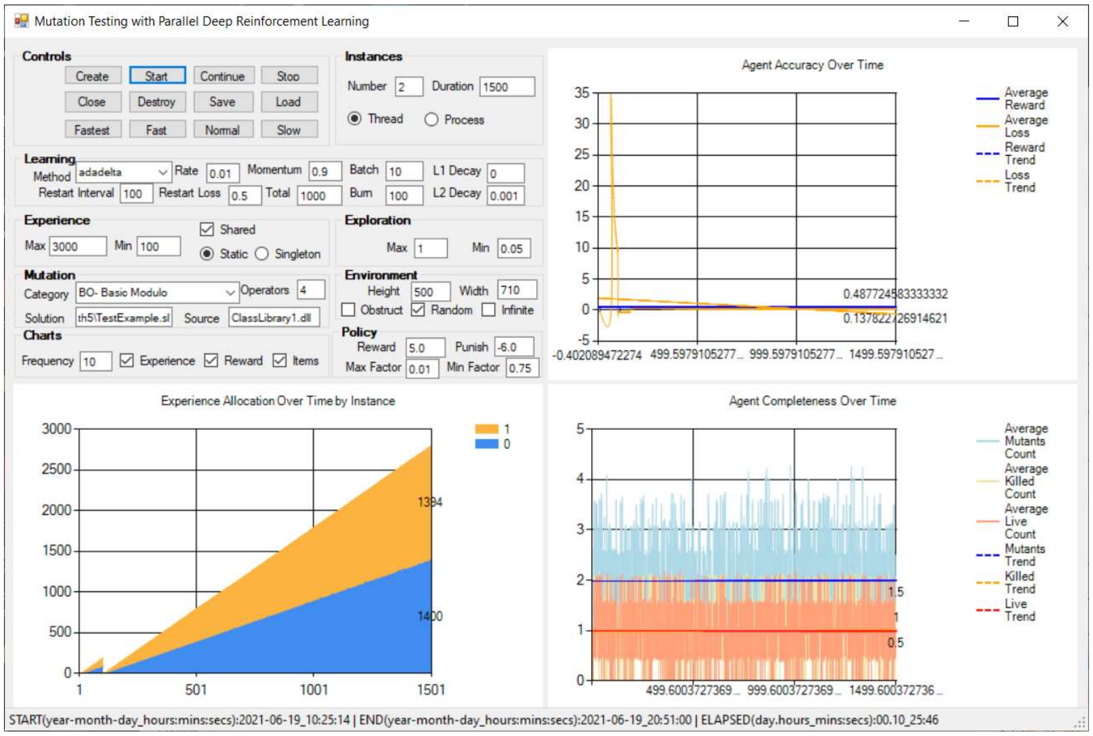
```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.191843004125643
smooth-ish reward: 0.515259166666656
Current Mutation Category: BO
Current Mutation Operator: 1010
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 42), (0001, 33), (0010, 36), (0011, 35), (0100, 33), (0101, 32), (0110, 42), (0111, 35), (1000, 40), (1001, 44), (1010, 962), (1011, 29),
(1100, 37), (1101, 36), (1110, 34), (1111, 31)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
☐ Charts ☐ Obstruct
✓ Shared ☐ Random
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

Random selection of mutation operators:

Results						
Test Metrics	**Test Average**	**Test Results**				
test runs		1	2	3	4	5
avg elapsed time (hh:mm:s	10:15:41	10:25:46	10:24:11	9:55:55	10:22:26	10:10:06
maximum action	n/a					
average mutation score	0.76					
average mutant total	1.700000000000000	1.50	2.00	1.00	2.50	1.50
average killed count	1.300000000000000	1.00	1.50	1.00	1.50	1.50
average live count	0.400000000000000	0.50	0.50	0.00	1.00	0.00
average Q-learn loss	0.126778508637186	0.14235967	0.18430662	0.08087368	0.07788896	0.14846361
smooth-ish reward	0.504345666666664	0.51459125	0.49512833	0.50377042	0.50956583	0.49867250
Test Metrics	**Test Average**	**Test Results**				
test runs - instance0		1	2	3	4	5
maximum action	n/a					
average Q-learn loss	0.128149205757168	0.08421698	0.1836353	0.07598023	0.07868408	0.21822945
smooth-ish reward	0.496130999999998	0.49842083	0.49088250	0.49715583	0.50225583	0.49194000
test runs - instance1		1	2	3	4	5
maximum action	n/a					
average Q-learn loss	0.125384138198002	0.20050770	0.18492067	0.08572057	0.07710995	0.07866180
smooth-ish reward	0.513328999999998	0.53178167	0.49937417	0.51188500	0.51739583	0.50620833
	Test Summary	1	2	3	4	5
maximum action	**n/a**					
average Q-learn loss	**0.126766671977585**	0.14236234	0.18427798	0.0808504	0.07789702	0.14844562
smooth-ish reward	**0.504729999999998**	0.51510125	0.49512833	0.50452042	0.50982583	0.49907417

1



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, [✓ Shared], [✓ Static] Singleton

Exploration: Max 1, Min 0.05

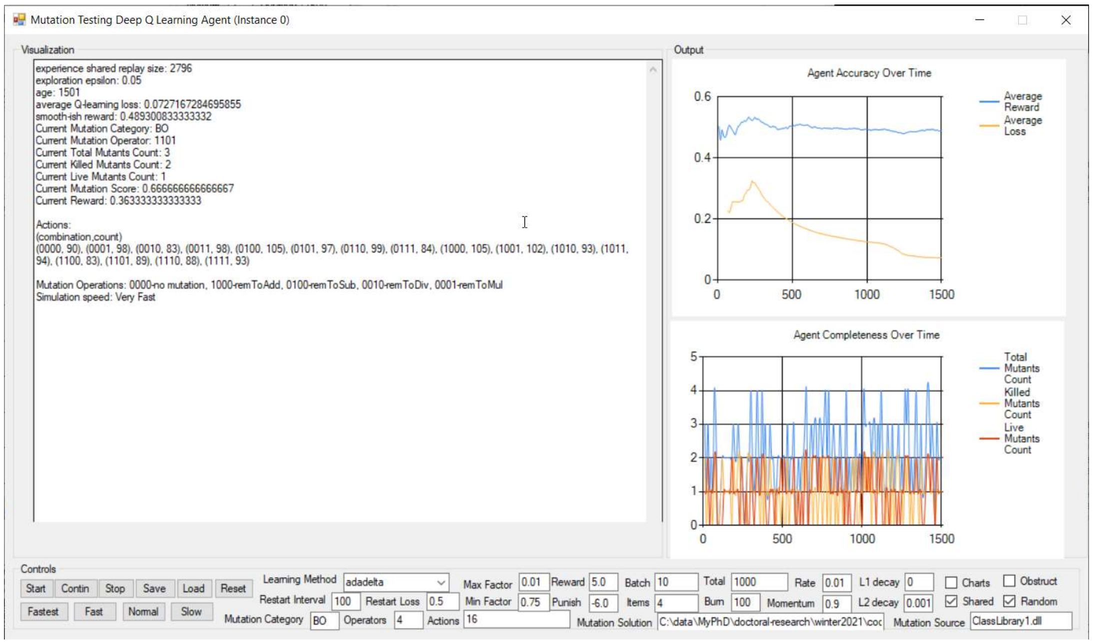
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, Obstruct [✓ Random] Infinite

Charts: Frequency 10, [✓ Experience] [✓ Reward] [✓ Items]

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-19_10:25:14 | END(year-month-day_hours:mins:secs):2021-06-19_20:51:00 | ELAPSED(day.hours_mins:secs):00.10_25:46

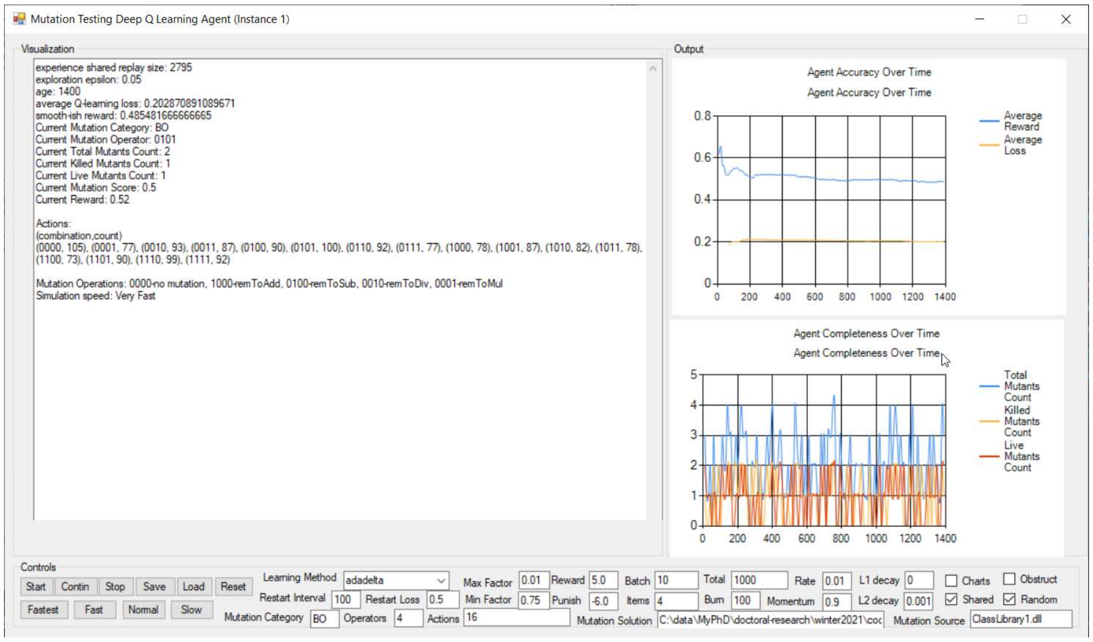


```
experience shared replay size: 2796
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0727167284695855
smooth-ish reward: 0.489300833333332
Current Mutation Category: BO
Current Mutation Operator: 1101
Current Total Mutants Count: 3
Current Killed Mutants Count: 2
Current Live Mutants Count: 1
Current Mutation Score: 0.666666666666667
Current Reward: 0.363333333333333

Actions:
(combination,count)
(0000, 90), (0001, 98), (0010, 83), (0011, 98), (0100, 105), (0101, 97), (0110, 99), (0111, 84), (1000, 105), (1001, 102), (1010, 93), (1011,
94), (1100, 83), (1101, 89), (1110, 88), (1111, 93)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, Charts, Obstruct

Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, [✓ Shared], [✓ Random]

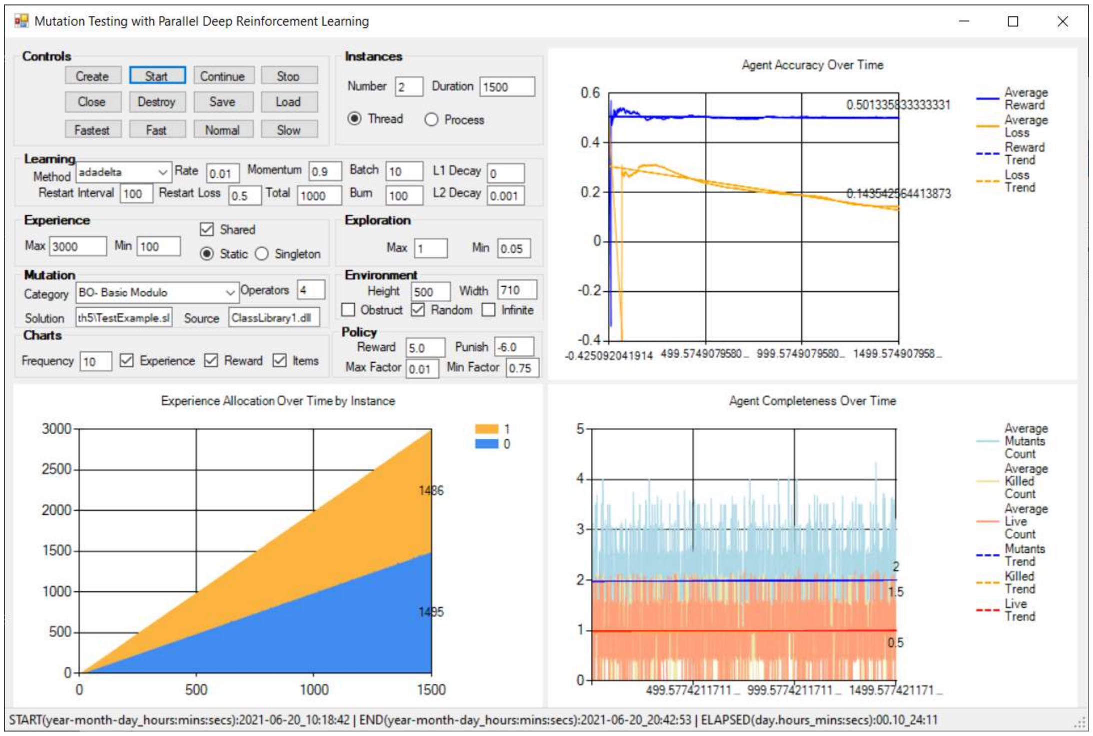
Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2795
exploration epsilon: 0.05
age: 1400
average Q-learning loss: 0.202870891089671
smooth-ish reward: 0.485481666666665
Current Mutation Category: BO
Current Mutation Operator: 0101
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 105), (0001, 77), (0010, 93), (0011, 87), (0100, 90), (0101, 100), (0110, 92), (0111, 77), (1000, 78), (1001, 87), (1010, 82), (1011, 78),
(1100, 73), (1101, 90), (1110, 99), (1111, 92)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0
☐ Charts
☐ Obstruct
Restart Interval: 100
Restart Loss: 0.5
Min Factor: 0.75
Punish: -6.0
Items: 4
Burn: 100
Momentum: 0.9
L2 decay: 0.001
☑ Shared
☑ Random
Mutation Category: BO
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

2



Mutation Testing with Parallel Deep Reinforcement Learning

Controls

Instances
Number 2 Duration 1500
[✓ Thread] Process

Learning
Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0
Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience
Max 3000 Min 100 ☑ Shared [✓ Static] Singleton

Exploration
Max 1 Min 0.05

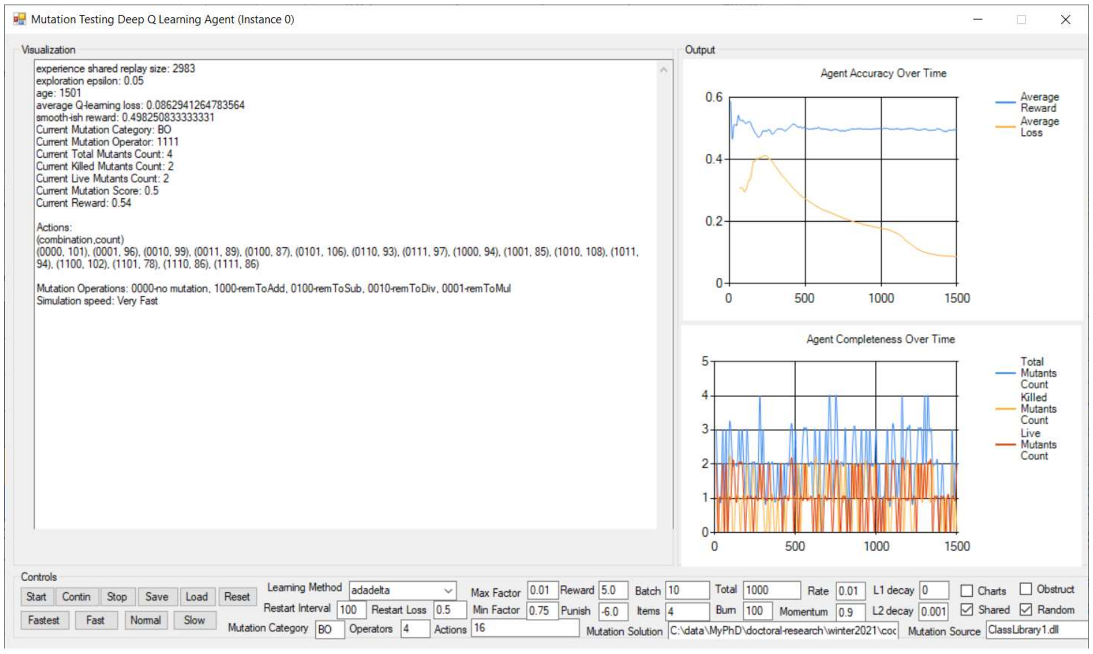
Mutation
Category BO- Basic Modulo Operators 4
Solution th5\TestExample.sl Source ClassLibrary1.dll

Environment
Height 500 Width 710
☐ Obstruct ☑ Random ☐ Infinite

Charts
Frequency 10 ☑ Experience ☑ Reward ☑ Items

Policy
Reward 5.0 Punish -6.0
Max Factor 0.01 Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-20_10:18:42 | END(year-month-day_hours:mins:secs):2021-06-20_20:42:53 | ELAPSED(day.hours_mins:secs):00.10_24:11



Mutation Testing Deep Q Learning Agent (Instance 0)

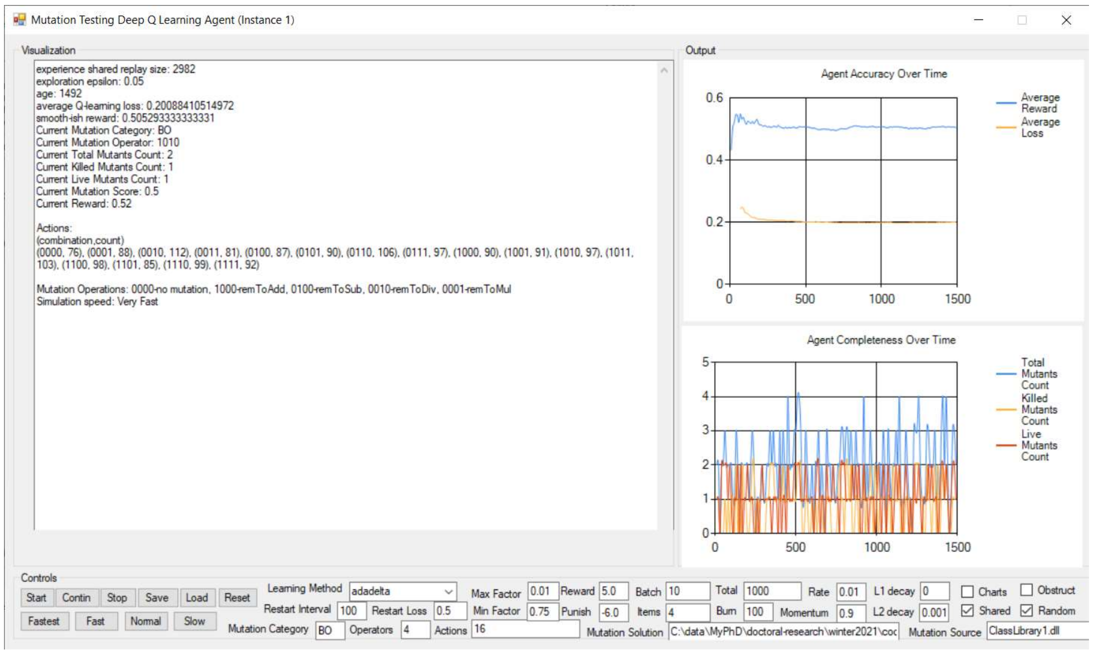
Visualization

```
experience shared replay size: 2983
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0862941264783564
smooth-ish reward: 0.498250833333331
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 101), (0001, 96), (0010, 99), (0011, 89), (0100, 87), (0101, 106), (0110, 93), (0111, 97), (1000, 94), (1001, 85), (1010, 108), (1011,
94), (1100, 102), (1101, 78), (1110, 86), (1111, 86)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Output

Controls

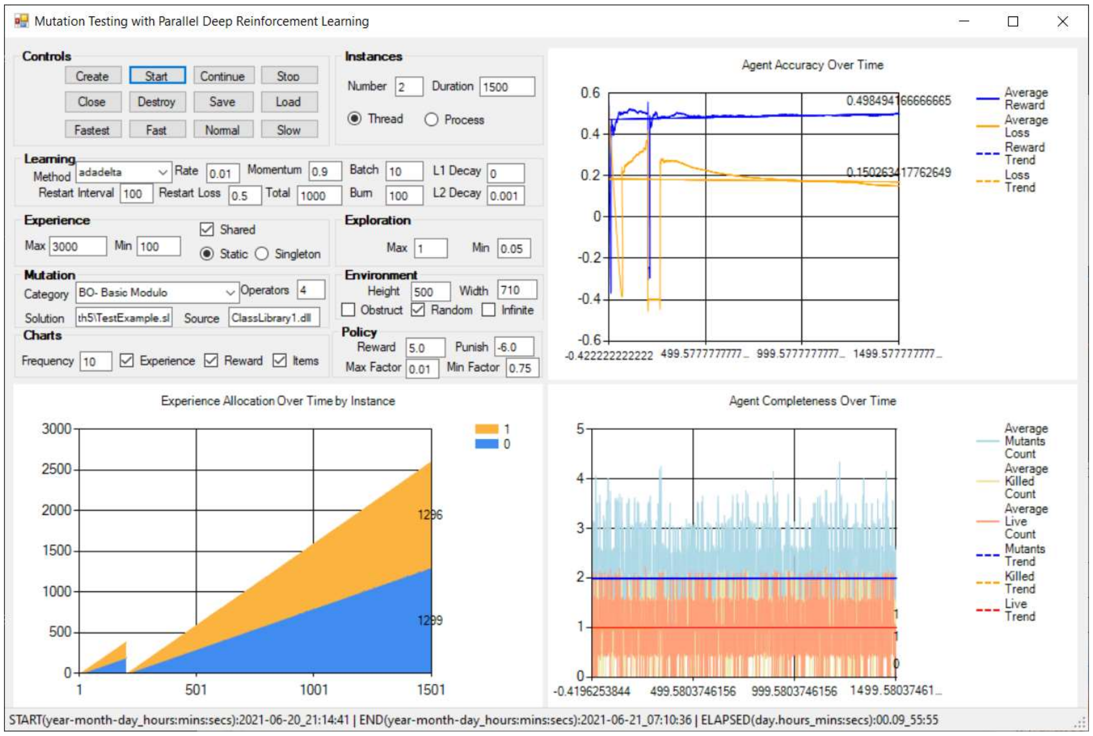
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0 ☐ Charts ☐ Obstruct
Restart Interval 100 Restart Loss 0.5 Min Factor 0.75 Punish -6.0 Items 4 Burn 100 Momentum 0.9 L2 decay 0.001 ☑ Shared ☑ Random
Mutation Category BO Operators 4 Actions 16 Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2982
exploration epsilon: 0.05
age: 1492
average Q-learning loss: 0.20088410514972
smooth-ish reward: 0.505293333333331
Current Mutation Category: BO
Current Mutation Operator: 1010
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 76), (0001, 88), (0010, 112), (0011, 81), (0100, 87), (0101, 90), (0110, 106), (0111, 97), (1000, 90), (1001, 91), (1010, 97), (1011,
103), (1100, 98), (1101, 85), (1110, 99), (1111, 92)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll
☐ Charts
☐ Obstruct
☑ Shared
☑ Random

3



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, [✓] Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

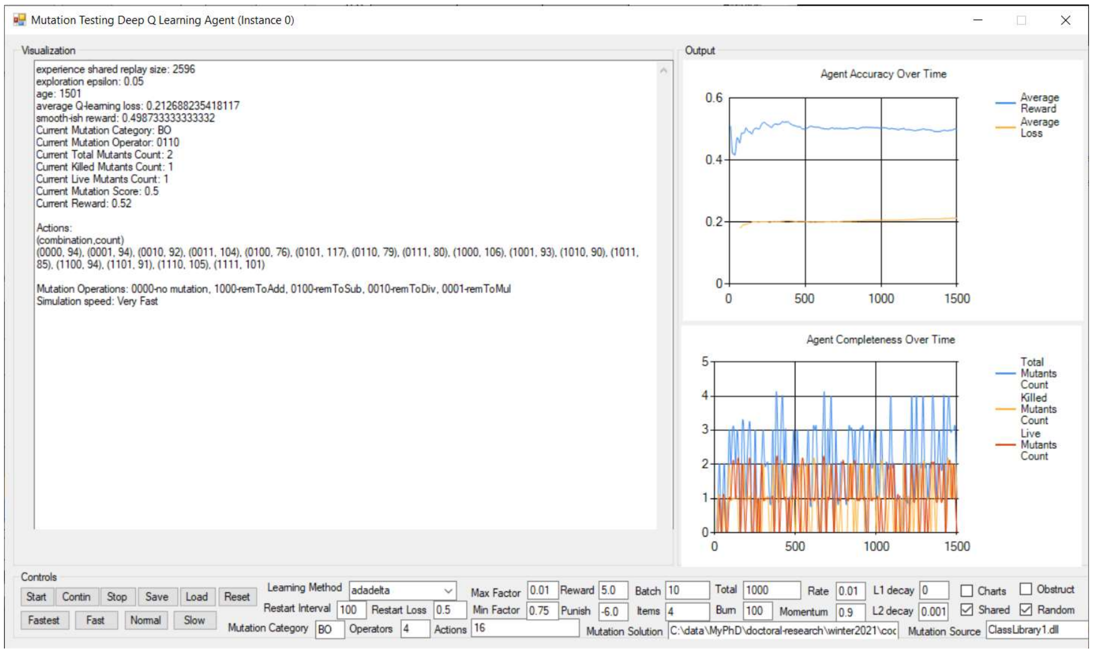
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, [ ] Obstruct, [✓] Random, [ ] Infinite

Charts: Frequency 10, [✓] Experience, [✓] Reward, [✓] Items

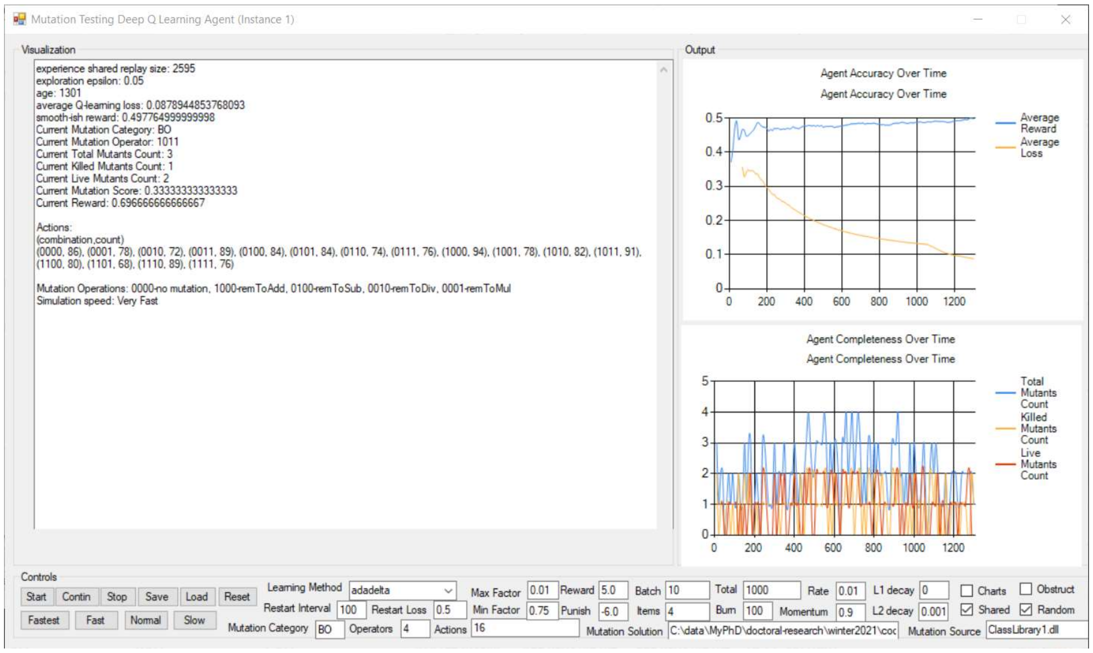
Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-20_21:14:41 | END(year-month-day_hours:mins:secs):2021-06-21_07:10:36 | ELAPSED(day.hours_mins:secs):00.09_55:55



```
experience shared replay size: 2596
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.212688235418117
smooth-ish reward: 0.498733333333332
Current Mutation Category: BO
Current Mutation Operator: 0110
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 94), (0001, 94), (0010, 92), (0011, 104), (0100, 76), (0101, 117), (0110, 79), (0111, 80), (1000, 106), (1001, 93), (1010, 90), (1011,
85), (1100, 94), (1101, 91), (1110, 105), (1111, 101)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

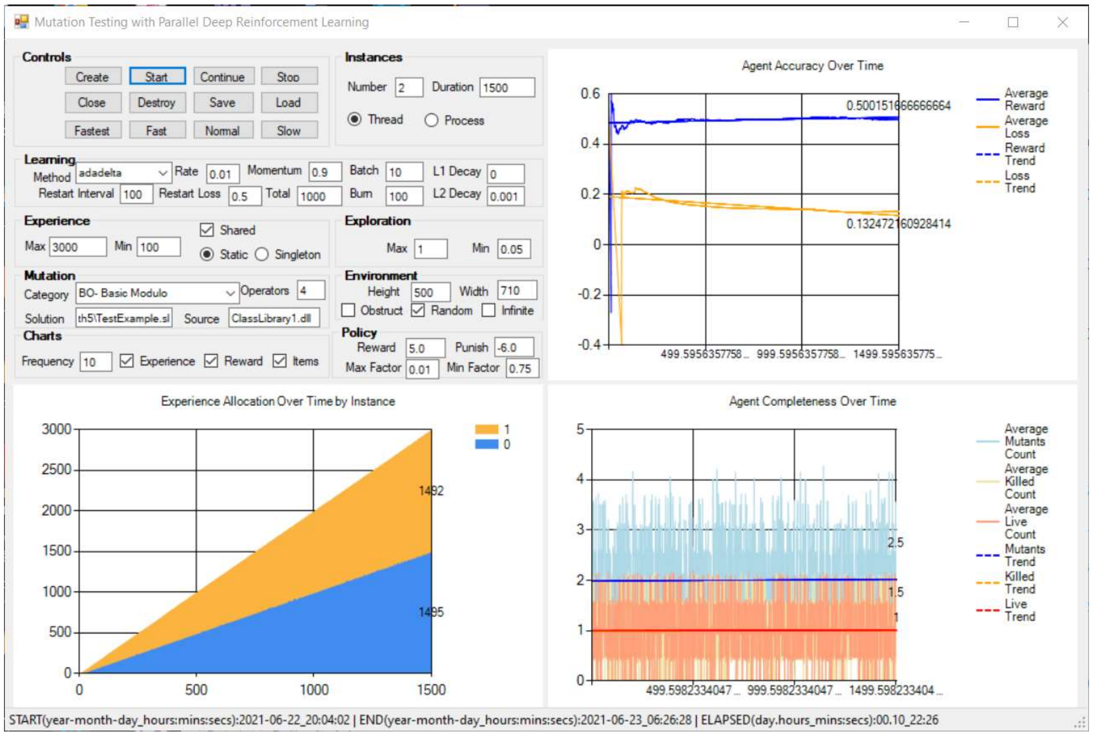
Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Mutation Category BO, Operators 4, Actions 16, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, [ ] Charts, [✓] Shared, [ ] Obstruct, [✓] Random, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2595
exploration epsilon: 0.05
age: 1301
average Q-learning loss: 0.0878944853768093
smooth-ish reward: 0.497764999999998
Current Mutation Category: BO
Current Mutation Operator: 1011
Current Total Mutants Count: 3
Current Killed Mutants Count: 1
Current Live Mutants Count: 2
Current Mutation Score: 0.333333333333333
Current Reward: 0.696666666666667

Actions:
(combination,count)
(0000, 86), (0001, 78), (0010, 72), (0011, 89), (0100, 84), (0101, 84), (0110, 74), (0111, 76), (1000, 94), (1001, 78), (1010, 82), (1011, 91),
(1100, 80), (1101, 68), (1110, 89), (1111, 76)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Charts: ☐
Shared: ☑
Obstruct: ☐
Random: ☑
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

4


Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, (● Thread ○ Process)

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, (● Static ○ Singleton)

Exploration: Max 1, Min 0.05

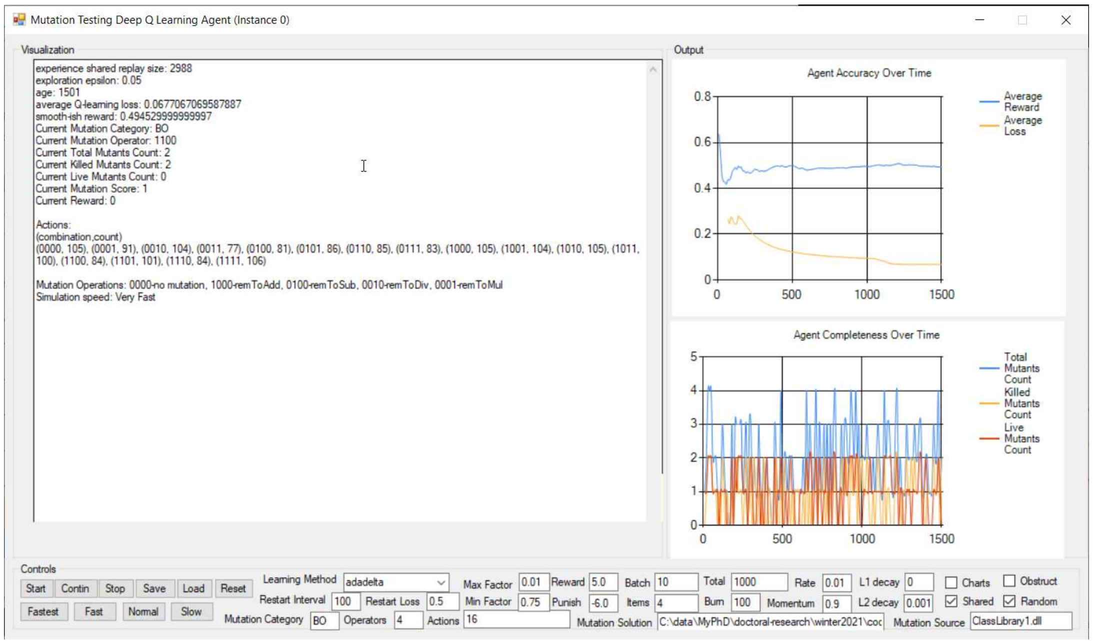
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☑ Random, ☐ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-22_20:04:02 | END(year-month-day_hours:mins:secs):2021-06-23_06:26:28 | ELAPSED(day.hours_mins:secs):00.10_22:26


Mutation Testing Deep Q Learning Agent (Instance 0)

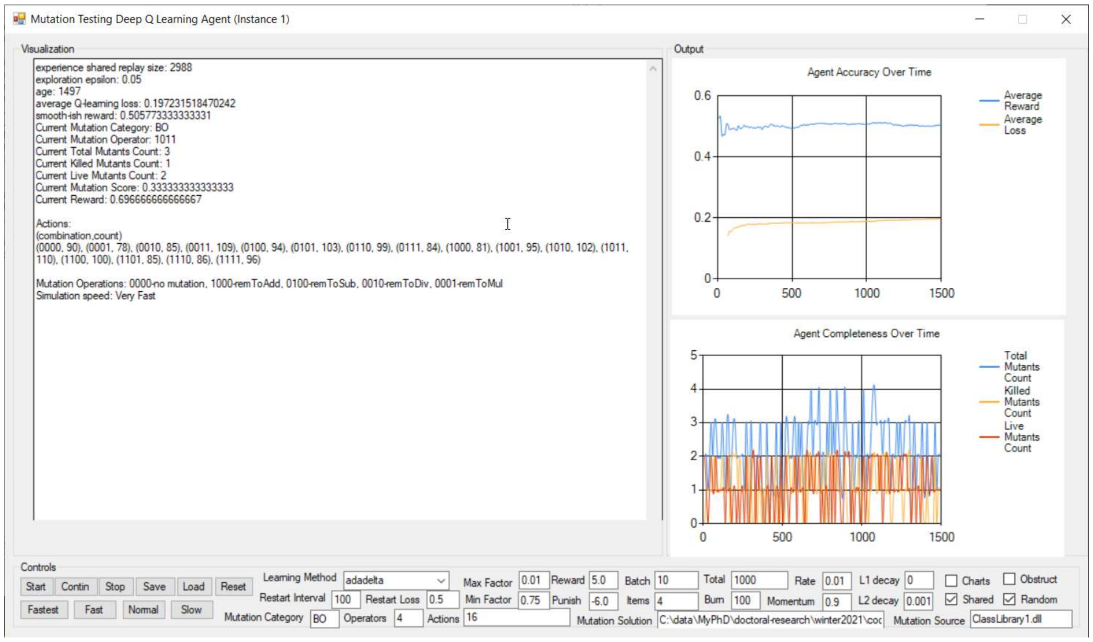
Visualization

```
experience shared replay size: 2988
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0677067069587887
smooth-ish reward: 0.494529999999997
Current Mutation Category: BO
Current Mutation Operator: 1100
Current Total Mutants Count: 2
Current Killed Mutants Count: 2
Current Live Mutants Count: 0
Current Mutation Score: 1
Current Reward: 0

Actions:
(combination,count)
(0000, 105), (0001, 91), (0010, 104), (0011, 77), (0100, 81), (0101, 86), (0110, 85), (0111, 83), (1000, 105), (1001, 104), (1010, 105), (1011,
100), (1100, 84), (1101, 101), (1110, 84), (1111, 106)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Output

Controls: Start, Contin, Stop, Save, Load, Reset, Fastest, Fast, Normal, Slow

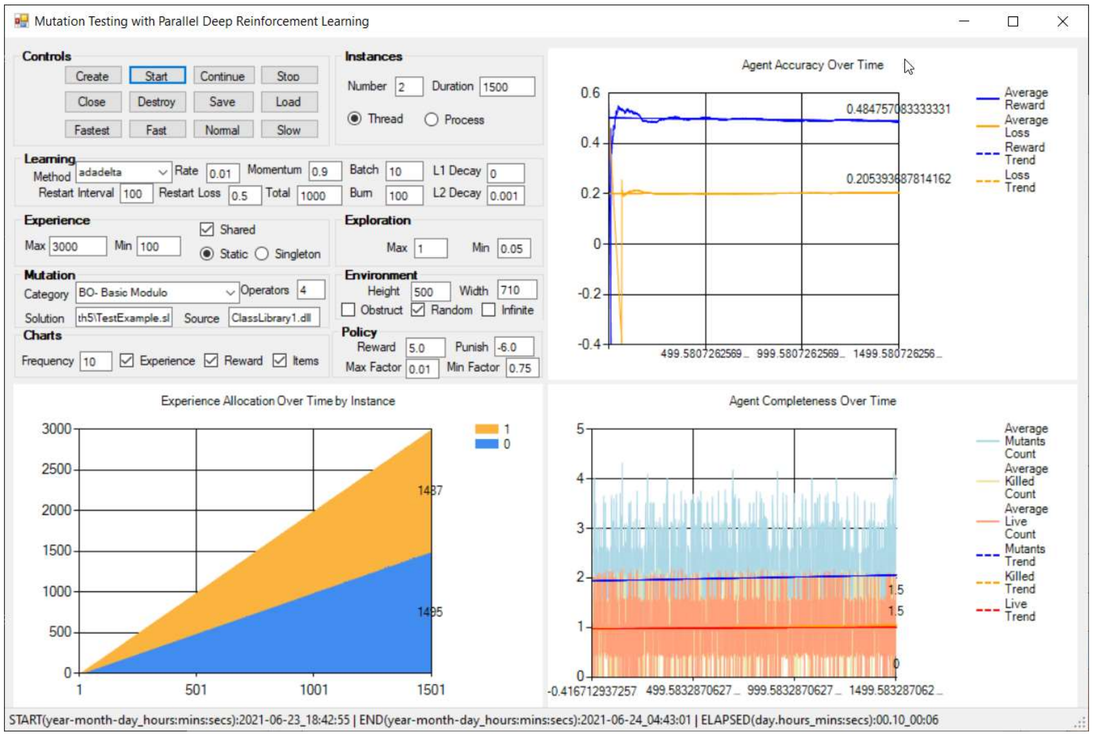
Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Mutation Category BO, Operators 4, Actions 16, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, ☐ Charts, ☑ Shared, ☐ Obstruct, ☑ Random, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2988
exploration epsilon: 0.05
age: 1497
average Q-learning loss: 0.197231518470242
smooth-ish reward: 0.505773333333331
Current Mutation Category: BO
Current Mutation Operator: 1011
Current Total Mutants Count: 3
Current Killed Mutants Count: 1
Current Live Mutants Count: 2
Current Mutation Score: 0.333333333333333
Current Reward: 0.696666666666667

Actions:
(combination,count)
(0000, 90), (0001, 78), (0010, 85), (0011, 109), (0100, 94), (0101, 103), (0110, 99), (0111, 84), (1000, 81), (1001, 95), (1010, 102), (1011,
110), (1100, 100), (1101, 85), (1110, 86), (1111, 96)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method adadelta
Restart Interval 100 Restart Loss 0.5
Mutation Category BO Operators 4 Actions 16
Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0 ☐ Charts ☐ Obstruct
Min Factor 0.75 Punish -6.0 Items 4 Burn 100 Momentum 0.9 L2 decay 0.001 ☑ Shared ☑ Random
Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc Mutation Source ClassLibrary1.dll

5


Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, (● Thread ○ Process)

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, (● Static ○ Singleton)

Exploration: Max 1, Min 0.05

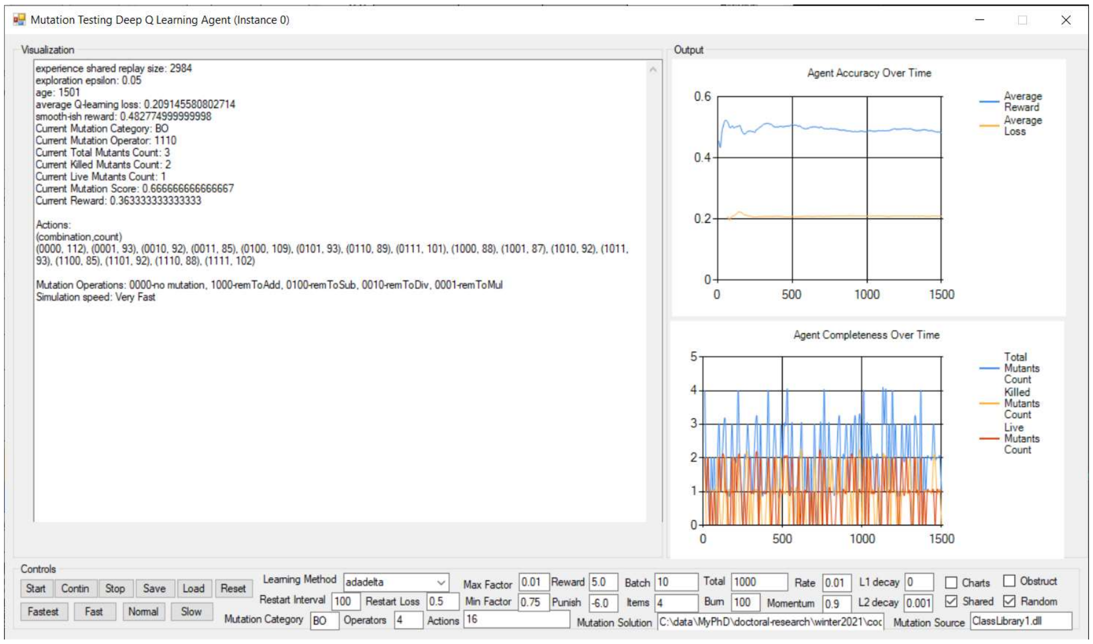
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☑ Random, ☐ Infinite

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

START(year-month-day_hours:mins:secs):2021-06-23_18:42:55 | END(year-month-day_hours:mins:secs):2021-06-24_04:43:01 | ELAPSED(day.hours_mins:secs):00.10_00:06


Mutation Testing Deep Q Learning Agent (Instance 0)

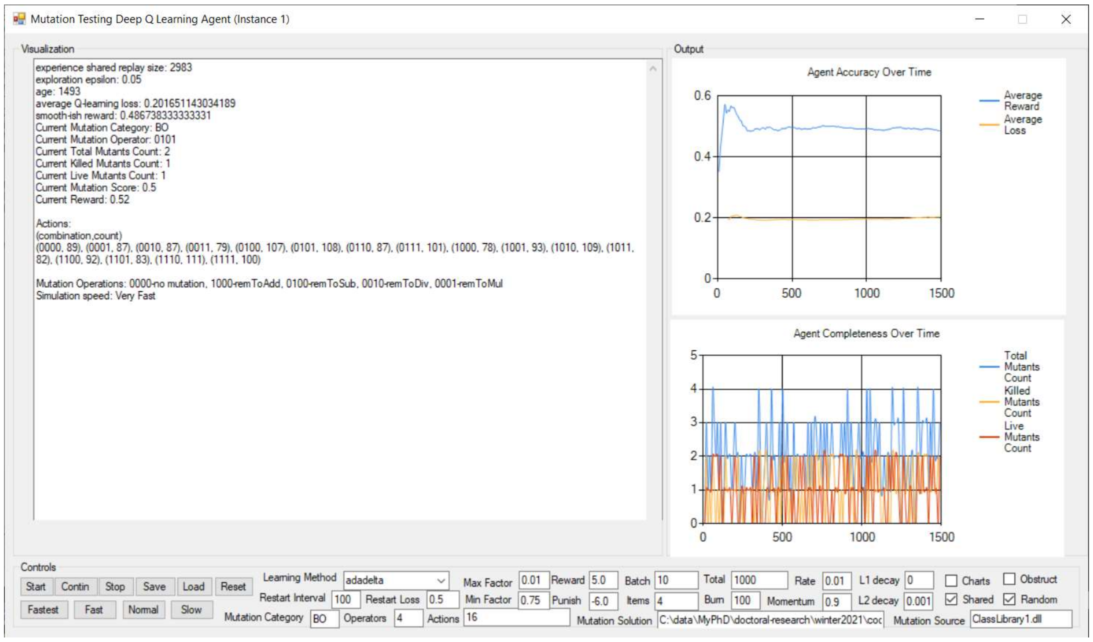
Visualization

```
experience shared replay size: 2984
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.209145580802714
smooth-ish reward: 0.482774999999998
Current Mutation Category: BO
Current Mutation Operator: 1110
Current Total Mutants Count: 3
Current Killed Mutants Count: 2
Current Live Mutants Count: 1
Current Mutation Score: 0.666666666666667
Current Reward: 0.363333333333333

Actions:
(combination,count)
(0000, 112), (0001, 93), (0010, 92), (0011, 85), (0100, 109), (0101, 93), (0110, 89), (0111, 101), (1000, 88), (1001, 87), (1010, 92), (1011,
93), (1100, 85), (1101, 92), (1110, 88), (1111, 102)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Controls: Start, Contin, Stop, Save, Load, Reset, Fastest, Fast, Normal, Slow

Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, ☐ Charts, ☐ Obstruct, ☑ Shared, ☑ Random

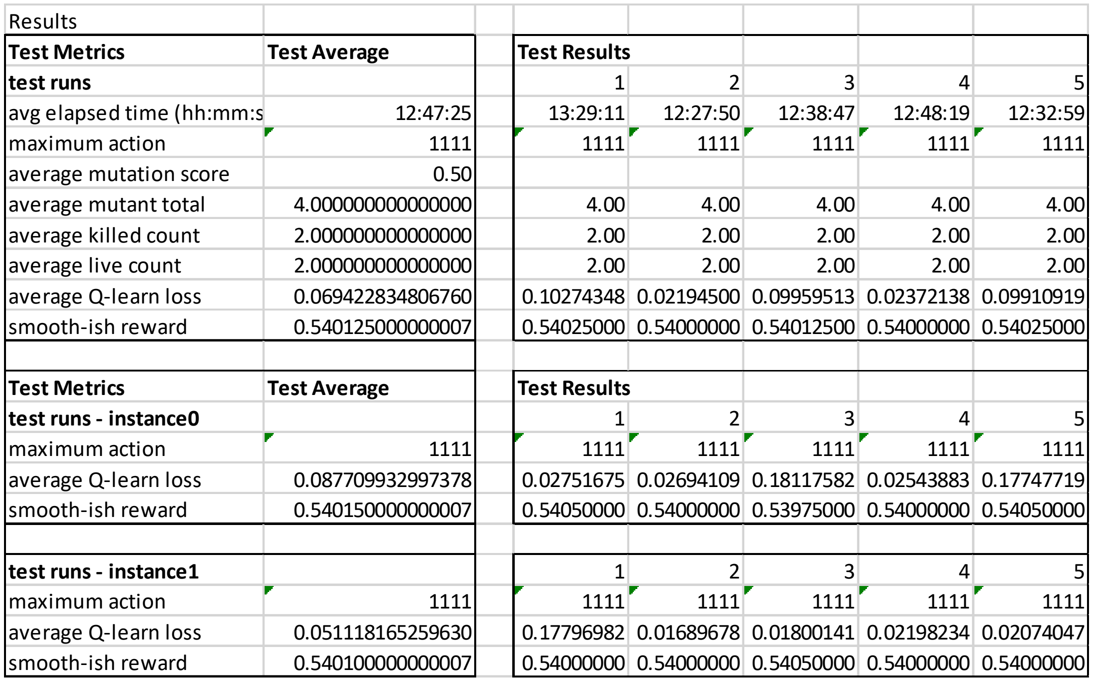
Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2983
exploration epsilon: 0.05
age: 1493
average Q-learning loss: 0.201651143034189
smooth-ish reward: 0.486738333333331
Current Mutation Category: BO
Current Mutation Operator: 0101
Current Total Mutants Count: 2
Current Killed Mutants Count: 1
Current Live Mutants Count: 1
Current Mutation Score: 0.5
Current Reward: 0.52

Actions:
(combination,count)
(0000, 89), (0001, 87), (0010, 87), (0011, 79), (0100, 107), (0101, 108), (0110, 87), (0111, 101), (1000, 78), (1001, 93), (1010, 109), (1011,
82), (1100, 92), (1101, 83), (1110, 111), (1111, 100)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

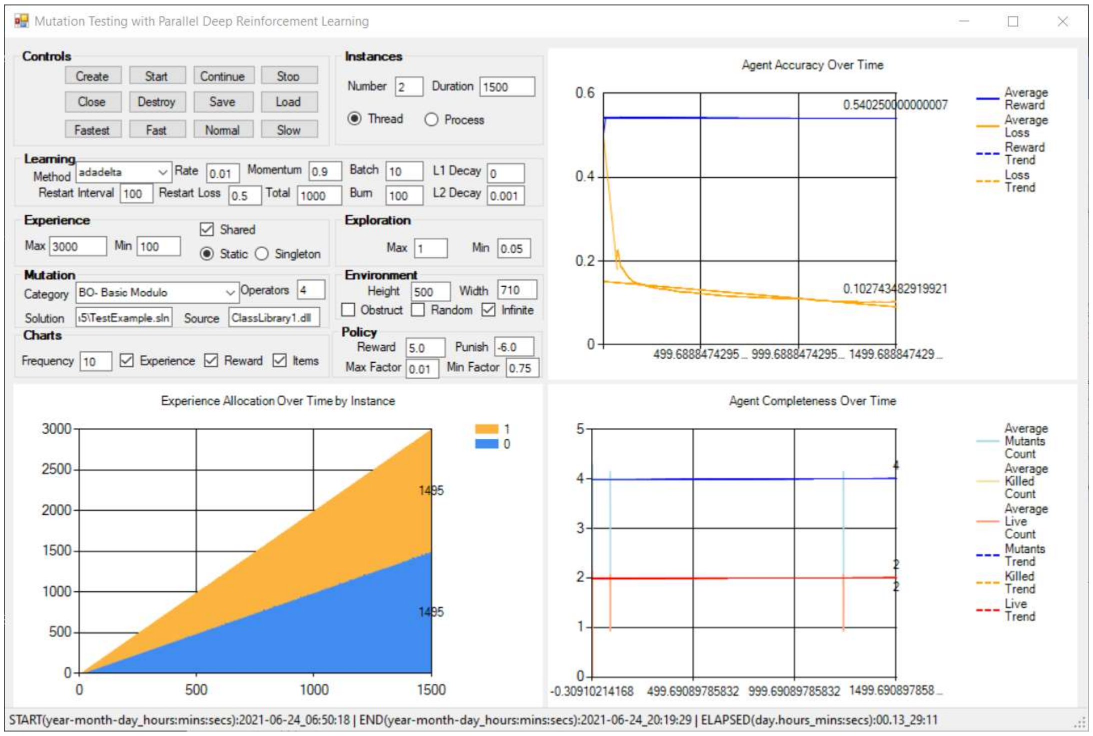
Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0
☐ Charts
☐ Obstruct
Restart Interval: 100
Restart Loss: 0.5
Min Factor: 0.75
Punish: -6.0
Items: 4
Burn: 100
Momentum: 0.9
L2 decay: 0.001
☑ Shared
☑ Random
Mutation Category: BO
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

Selection of all mutation operators:



Results						
Test Metrics	**Test Average**	**Test Results**				
test runs		1	2	3	4	5
avg elapsed time (hh:mm:s	12:47:25	13:29:11	12:27:50	12:38:47	12:48:19	12:32:59
maximum action	1111	1111	1111	1111	1111	1111
average mutation score	0.50					
average mutant total	4.000000000000000	4.00	4.00	4.00	4.00	4.00
average killed count	2.000000000000000	2.00	2.00	2.00	2.00	2.00
average live count	2.000000000000000	2.00	2.00	2.00	2.00	2.00
average Q-learn loss	0.069422834806760	0.10274348	0.02194500	0.09959513	0.02372138	0.09910919
smooth-ish reward	0.540125000000007	0.54025000	0.54000000	0.54012500	0.54000000	0.54025000
Test Metrics	**Test Average**	**Test Results**				
test runs - instance0		1	2	3	4	5
maximum action	1111	1111	1111	1111	1111	1111
average Q-learn loss	0.087709932997378	0.02751675	0.02694109	0.18117582	0.02543883	0.17747719
smooth-ish reward	0.540150000000007	0.54050000	0.54000000	0.53975000	0.54000000	0.54050000
test runs - instance1		1	2	3	4	5
maximum action	1111	1111	1111	1111	1111	1111
average Q-learn loss	0.051118165259630	0.17796982	0.01689678	0.01800141	0.02198234	0.02074047
smooth-ish reward	0.540100000000007	0.54000000	0.54000000	0.54050000	0.54000000	0.54000000

1



Mutation Testing with Parallel Deep Reinforcement Learning

Controls

Instances
Number 2 Duration 1500
(● Thread) ○ Process

Learning
Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0
Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience
Max 3000 Min 100 [✓] Shared
(● Static) ○ Singleton

Exploration
Max 1 Min 0.05

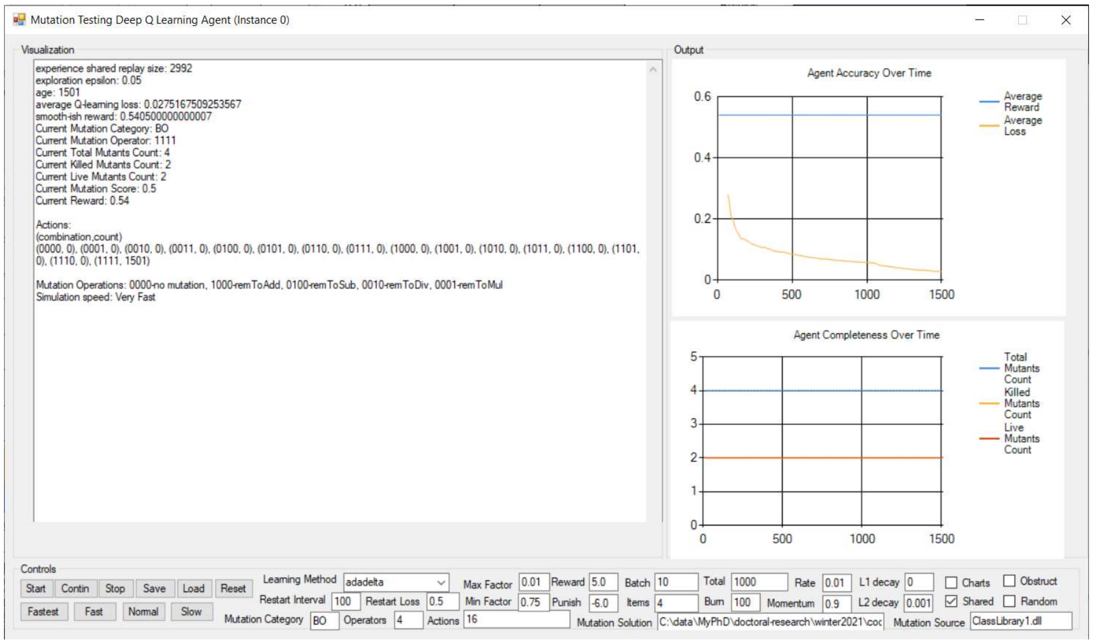
Mutation
Category BO- Basic Modulo Operators 4
Solution ;5\TestExample.sln Source ClassLibrary1.dll

Environment
Height 500 Width 710
☐ Obstruct ☐ Random [✓] Infinite

Charts
Frequency 10 [✓] Experience [✓] Reward [✓] Items

Policy
Reward 5.0 Punish -6.0
Max Factor 0.01 Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-24_06:50:18 | END(year-month-day_hours:mins:secs):2021-06-24_20:19:29 | ELAPSED(day.hours_mins:secs):00.13_29:11



Mutation Testing Deep Q Learning Agent (Instance 0)

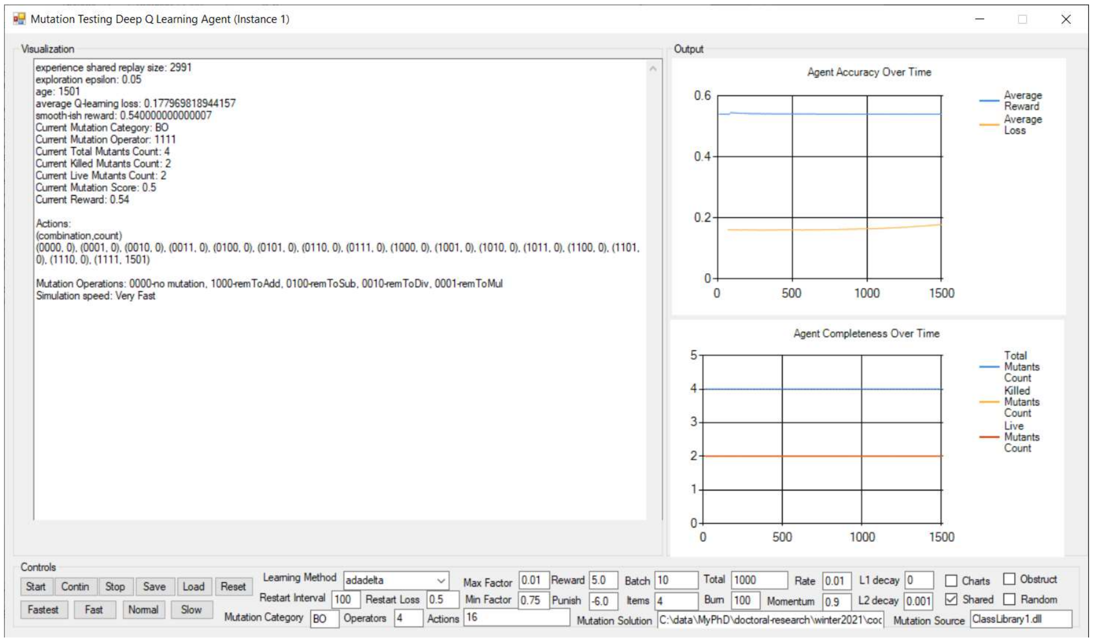
Visualization

```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0275167509253567
smooth-ish reward: 0.540500000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Output

Controls

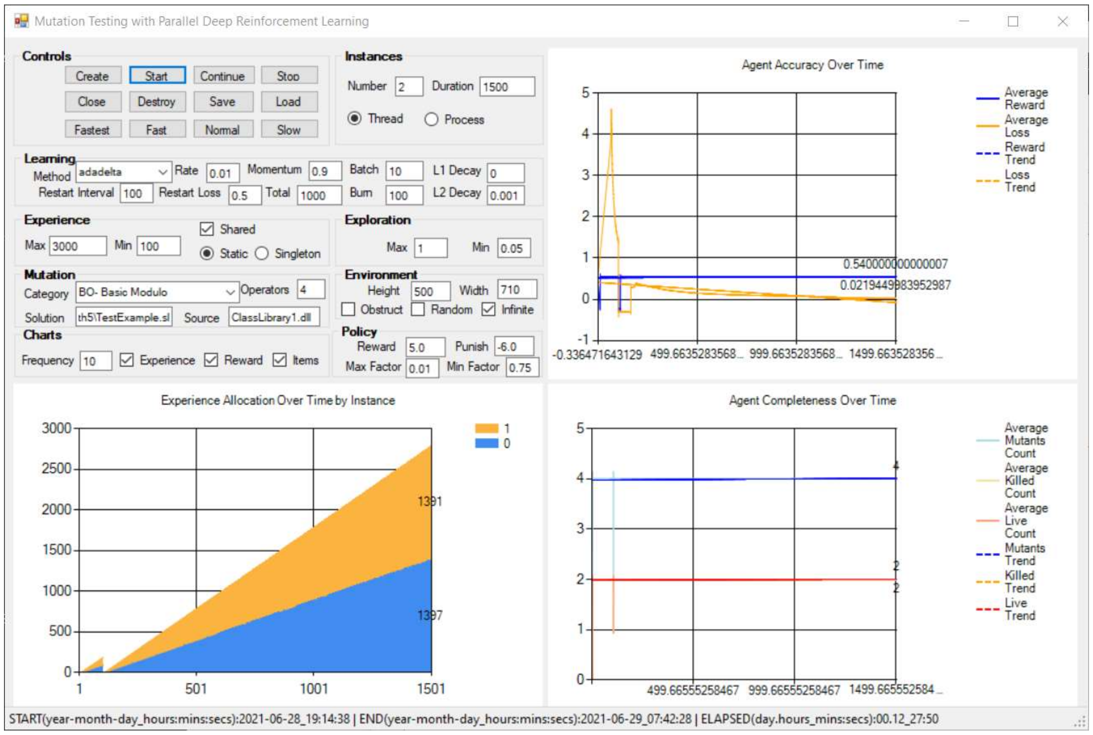
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0 ☐ Charts ☐ Obstruct
Restart Interval 100 Restart Loss 0.5 Min Factor 0.75 Punish -6.0 Items 4 Burn 100 Momentum 0.9 L2 decay 0.001 [✓] Shared ☐ Random
Mutation Category BO Operators 4 Actions 16 Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.177969818944157
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
☐ Charts
☑ Shared
☐ Obstruct
☐ Random
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

2



Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

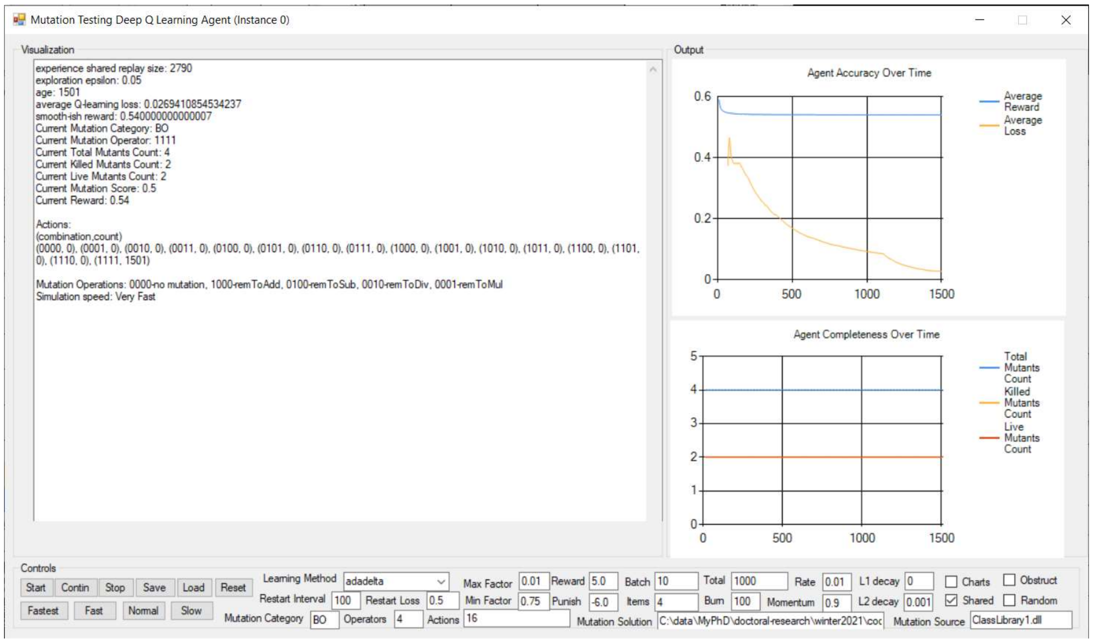
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☑ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

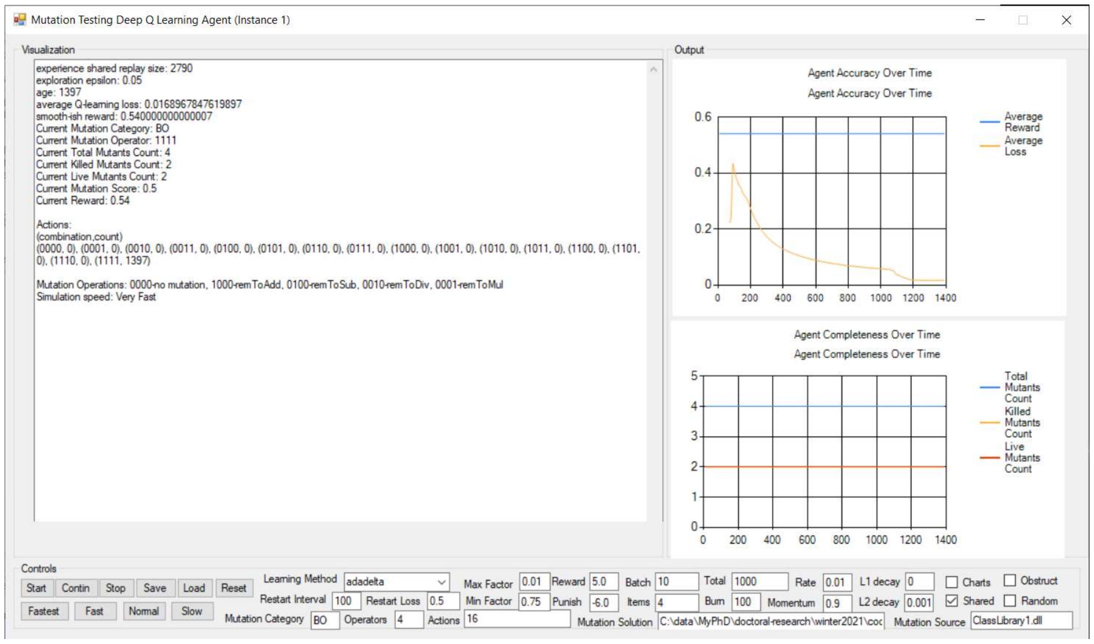
```
START(year-month-day_hours:mins:secs):2021-06-28_19:14:38 | END(year-month-day_hours:mins:secs):2021-06-29_07:42:28 | ELAPSED(day.hours_mins:secs):00.12_27:50
```



```
experience shared replay size: 2790
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0269410854534237
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, ☐ Charts, ☑ Shared, ☐ Obstruct, ☐ Random

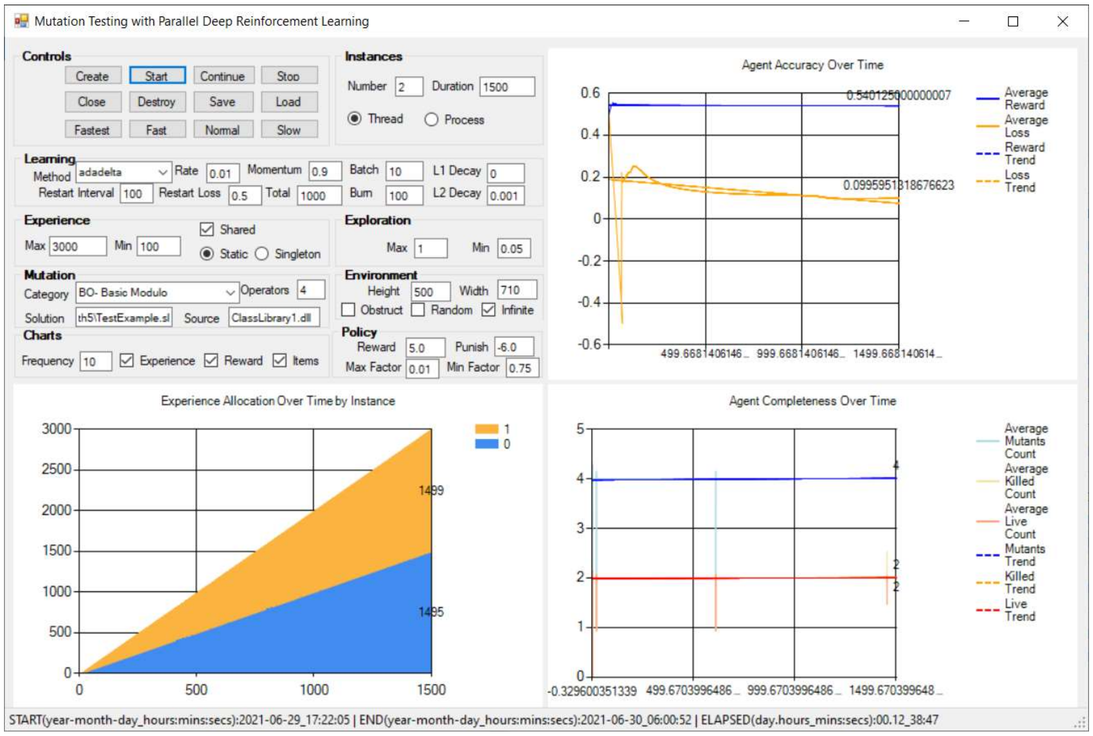
Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2790
exploration epsilon: 0.05
age: 1397
average Q-learning loss: 0.0168967847619897
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1397)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
☐ Charts
☑ Shared
☐ Obstruct
☐ Random
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

3



Controls

Instances
Number 2 Duration 1500
[✓ Thread] Process

Learning
Method adadelta Rate 0.01 Momentum 0.9 Batch 10 L1 Decay 0
Restart Interval 100 Restart Loss 0.5 Total 1000 Burn 100 L2 Decay 0.001

Experience
Max 3000 Min 100 [✓ Shared]
[✓ Static] Singleton

Exploration
Max 1 Min 0.05

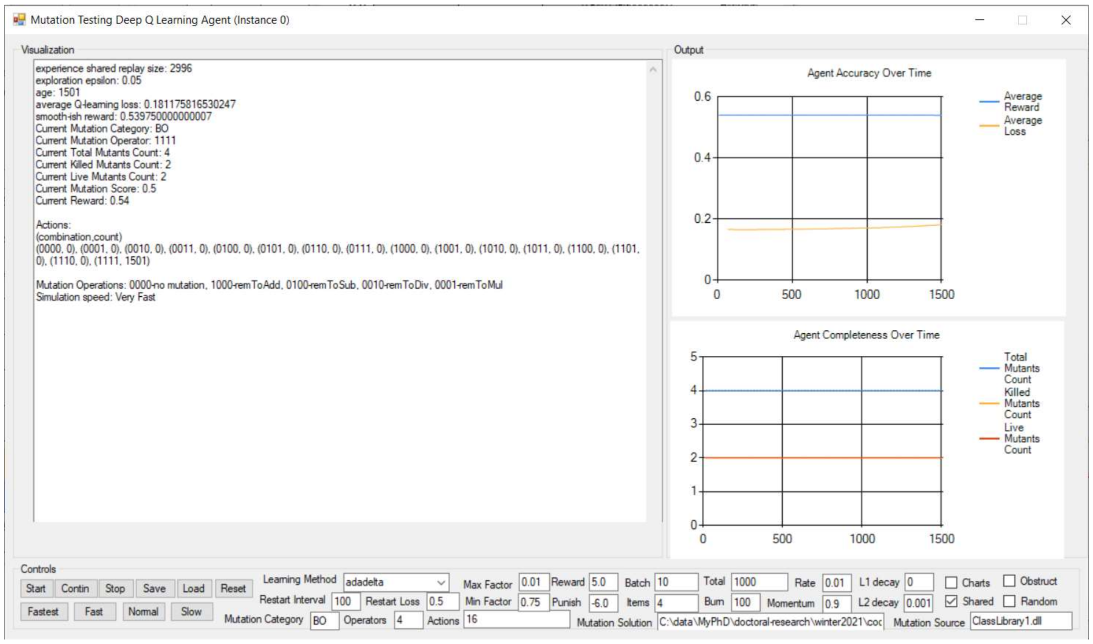
Mutation
Category BO- Basic Modulo Operators 4
Solution th5\TestExample.sl Source ClassLibrary1.dll

Environment
Height 500 Width 710
Obstruct Random [✓ Infinite]

Charts
Frequency 10 [✓ Experience] [✓ Reward] [✓ Items]

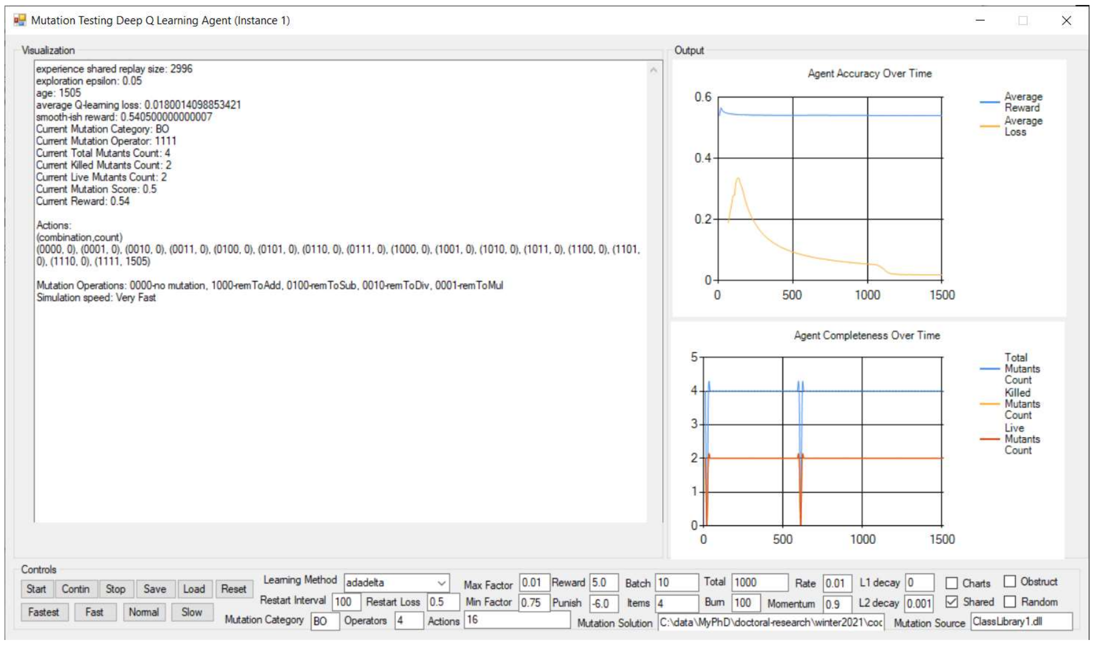
Policy
Reward 5.0 Punish -6.0
Max Factor 0.01 Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-29_17:22:05 | END(year-month-day_hours:mins:secs):2021-06-30_06:00:52 | ELAPSED(day.hours_mins:secs):00.12_38:47



```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.181175816530247
smooth-ish reward: 0.539750000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

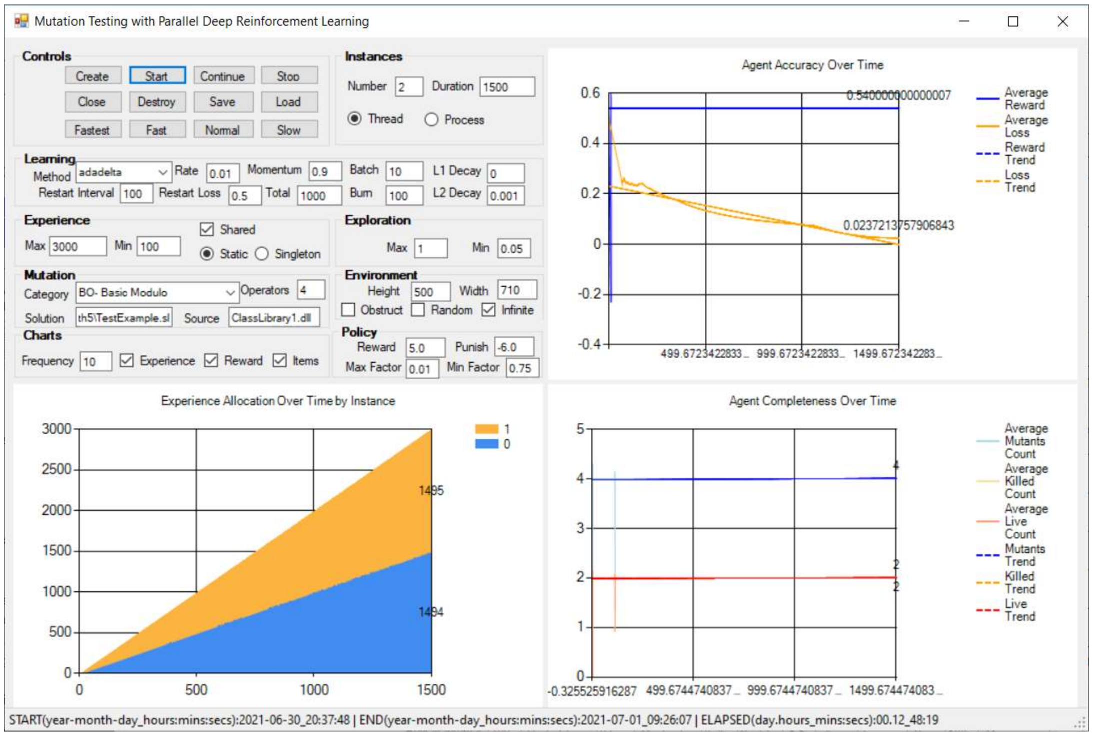
Learning Method adadelta Max Factor 0.01 Reward 5.0 Batch 10 Total 1000 Rate 0.01 L1 decay 0 Charts Obstruct
Restart Interval 100 Restart Loss 0.5 Min Factor 0.75 Punish -6.0 Items 4 Burn 100 Momentum 0.9 L2 decay 0.001 [✓ Shared] Random
Mutation Category BO Operators 4 Actions 16 Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2996
exploration epsilon: 0.05
age: 1505
average Q-learning loss: 0.0180014098853421
smooth-ish reward: 0.540500000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1505)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01 | Reward: 5.0 | Batch: 10 | Total: 1000 | Rate: 0.01 | L1 decay: 0
Restart Interval: 100 | Restart Loss: 0.5 | Min Factor: 0.75 | Punish: -6.0 | Items: 4 | Burn: 100 | Momentum: 0.9 | L2 decay: 0.001
Mutation Category: BO | Operators: 4 | Actions: 16 | Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc | Mutation Source: ClassLibrary1.dll
☐ Charts ☐ Obstruct ☑ Shared ☐ Random

4



Mutation Testing with Parallel Deep Reinforcement Learning

Controls

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

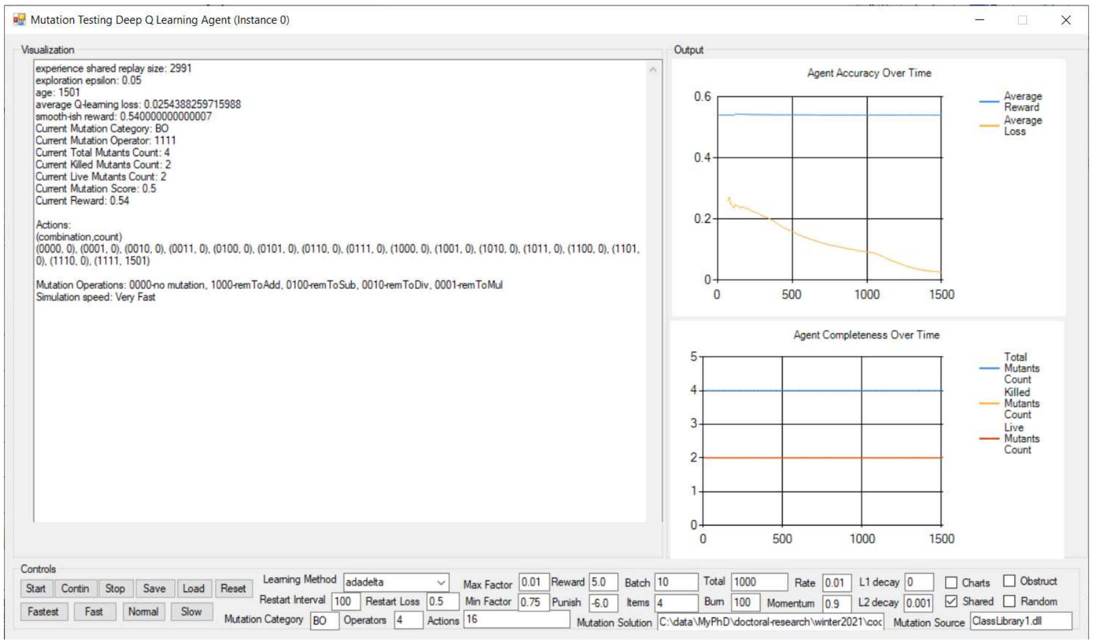
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☑ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-06-30_20:37:48 | END(year-month-day_hours:mins:secs):2021-07-01_09:26:07 | ELAPSED(day.hours_mins:secs):00.12_48:19



Mutation Testing Deep Q Learning Agent (Instance 0)

Visualization

```
experience shared replay size: 2991
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.0254388259715988
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

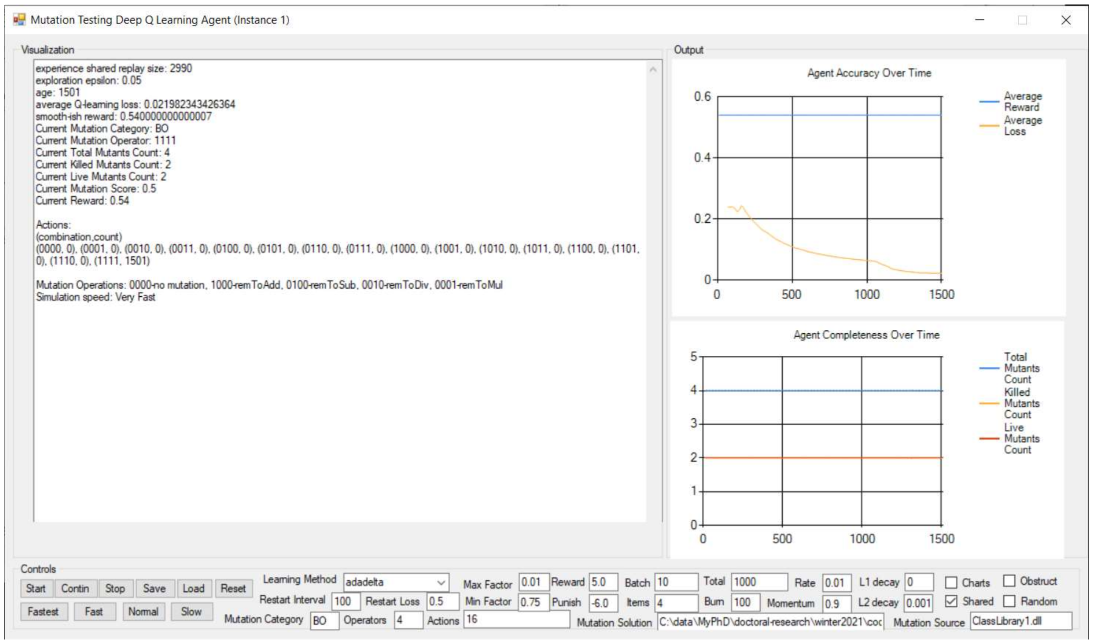
Output

Controls

Learning Method adadelta, Max Factor 0.01, Reward 5.0, Batch 10, Total 1000, Rate 0.01, L1 decay 0, ☐ Charts, ☐ Obstruct

Restart Interval 100, Restart Loss 0.5, Min Factor 0.75, Punish -6.0, Items 4, Burn 100, Momentum 0.9, L2 decay 0.001, ☑ Shared, ☐ Random

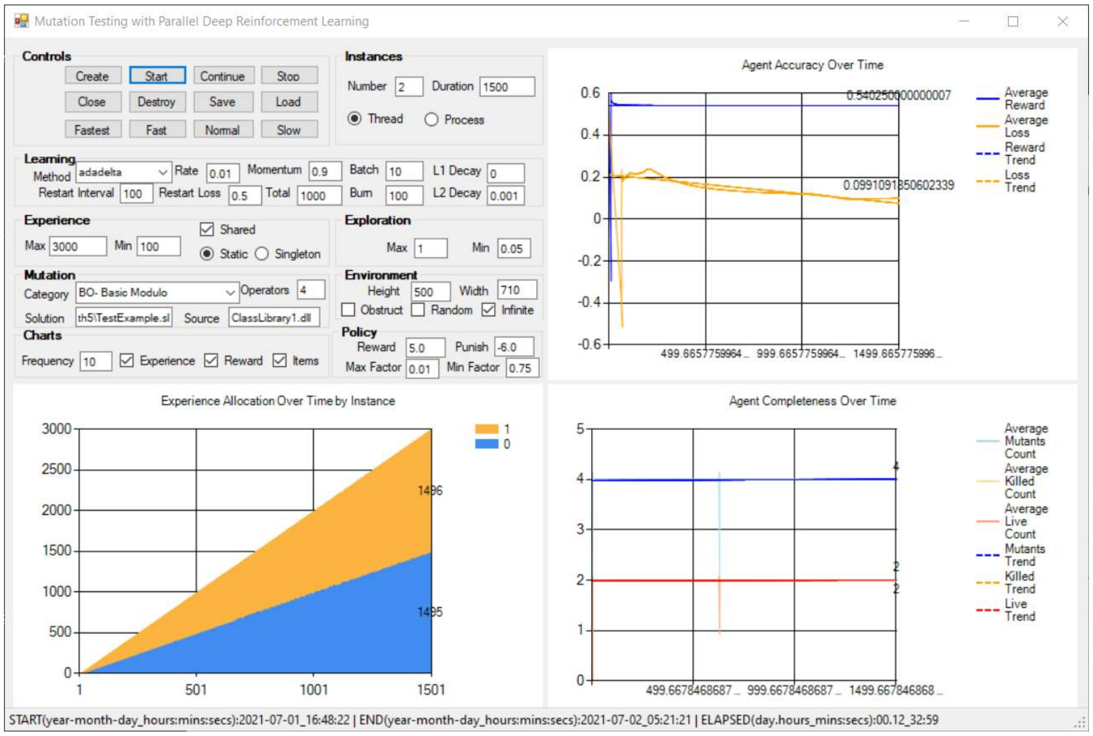
Mutation Category BO, Operators 4, Actions 16, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2990
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.021982343426364
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Max Factor: 0.01
Reward: 5.0
Batch: 10
Total: 1000
Rate: 0.01
L1 decay: 0
Restart Interval: 100
Restart Loss: 0.5
Min Factor: 0.75
Punish: -6.0
Items: 4
Burn: 100
Momentum: 0.9
L2 decay: 0.001
☐ Charts
☐ Obstruct
☑ Shared
☐ Random
Mutation Category: BO
Operators: 4
Actions: 16
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll

5



Mutation Testing with Parallel Deep Reinforcement Learning

Controls: Create, Start, Continue, Stop, Close, Destroy, Save, Load, Fastest, Fast, Normal, Slow

Instances: Number 2, Duration 1500, [✓ Thread] Process

Learning: Method adadelta, Rate 0.01, Momentum 0.9, Batch 10, L1 Decay 0, Restart Interval 100, Restart Loss 0.5, Total 1000, Burn 100, L2 Decay 0.001

Experience: Max 3000, Min 100, ☑ Shared, [✓ Static] Singleton

Exploration: Max 1, Min 0.05

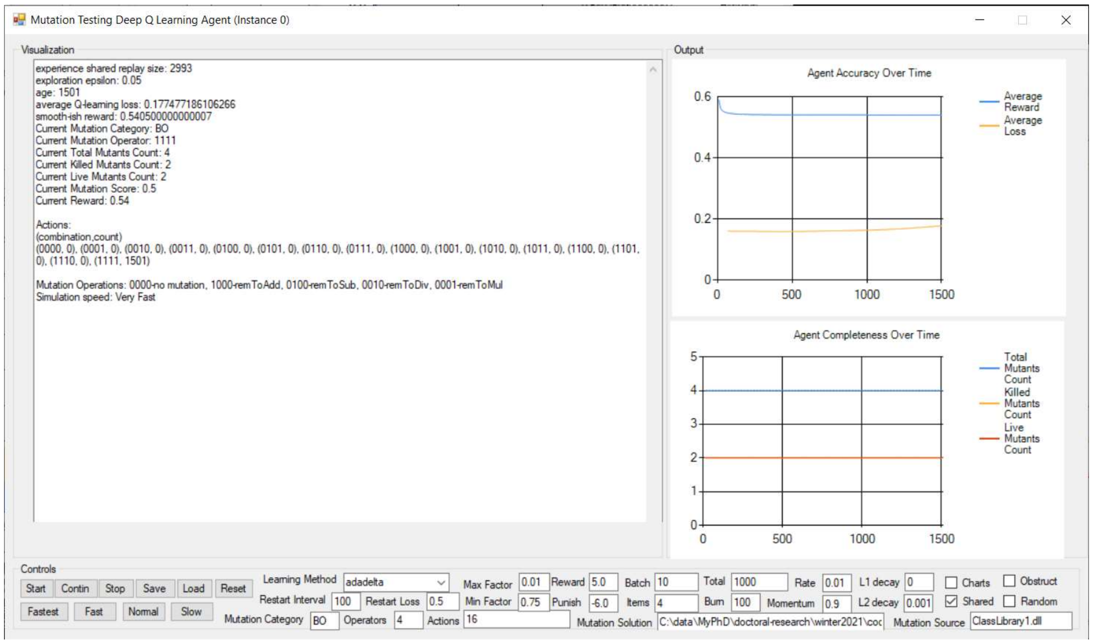
Mutation: Category BO- Basic Modulo, Operators 4, Solution th5\TestExample.sl, Source ClassLibrary1.dll

Environment: Height 500, Width 710, ☐ Obstruct, ☐ Random, ☑ Infinite

Charts: Frequency 10, ☑ Experience, ☑ Reward, ☑ Items

Policy: Reward 5.0, Punish -6.0, Max Factor 0.01, Min Factor 0.75

START(year-month-day_hours:mins:secs):2021-07-01_16:48:22 | END(year-month-day_hours:mins:secs):2021-07-02_05:21:21 | ELAPSED(day.hours_mins:secs):00.12_32:59



Mutation Testing Deep Q Learning Agent (Instance 0)

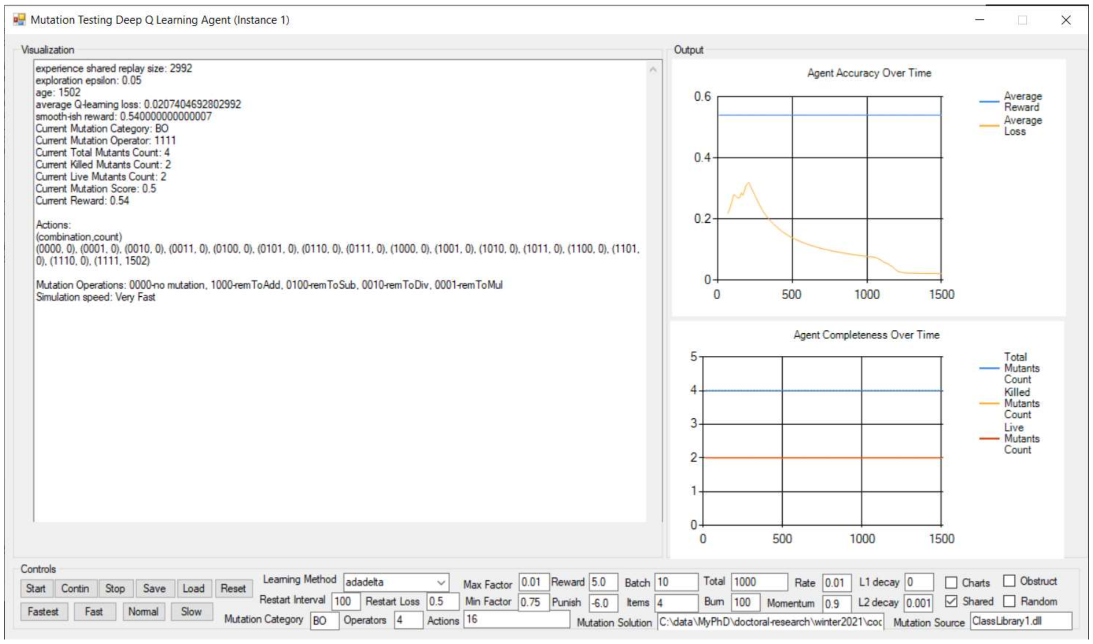
Visualization

```
experience shared replay size: 2993
exploration epsilon: 0.05
age: 1501
average Q-learning loss: 0.177477186106266
smooth-ish reward: 0.540500000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1501)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Output

Controls: Start, Contin, Stop, Save, Load, Reset, Fastest, Fast, Normal, Slow

Learning Method adadelta, Restart Interval 100, Restart Loss 0.5, Mutation Category BO, Operators 4, Actions 16, Max Factor 0.01, Min Factor 0.75, Reward 5.0, Punish -6.0, Batch 10, Items 4, Total 1000, Burn 100, Rate 0.01, Momentum 0.9, L1 decay 0, L2 decay 0.001, ☐ Charts, ☑ Shared, ☐ Obstruct, ☐ Random, Mutation Solution C:\data\MyPhD\doctoral-research\winter2021\coc, Mutation Source ClassLibrary1.dll



```
experience shared replay size: 2992
exploration epsilon: 0.05
age: 1502
average Q-learning loss: 0.0207404692802992
smooth-ish reward: 0.540000000000007
Current Mutation Category: BO
Current Mutation Operator: 1111
Current Total Mutants Count: 4
Current Killed Mutants Count: 2
Current Live Mutants Count: 2
Current Mutation Score: 0.5
Current Reward: 0.54

Actions:
(combination,count)
(0000, 0), (0001, 0), (0010, 0), (0011, 0), (0100, 0), (0101, 0), (0110, 0), (0111, 0), (1000, 0), (1001, 0), (1010, 0), (1011, 0), (1100, 0), (1101,
0), (1110, 0), (1111, 1502)

Mutation Operations: 0000-no mutation, 1000-remToAdd, 0100-remToSub, 0010-remToDiv, 0001-remToMul
Simulation speed: Very Fast
```

Learning Method: adadelta
Restart Interval: 100
Restart Loss: 0.5
Mutation Category: BO
Operators: 4
Actions: 16
Max Factor: 0.01
Min Factor: 0.75
Reward: 5.0
Punish: -6.0
Batch: 10
Items: 4
Total: 1000
Burn: 100
Rate: 0.01
Momentum: 0.9
L1 decay: 0
L2 decay: 0.001
Mutation Solution: C:\data\MyPhD\doctoral-research\winter2021\coc
Mutation Source: ClassLibrary1.dll
☐ Charts ☐ Obstruct
☑ Shared ☐ Random

References

Beck, K. (2003). *Test-driven development: by example*. Addison-Wesley Professional.

Boehm, H. J. (2005). Threads cannot be implemented as a library. *ACM Sigplan Notices*, *40*(6), 261-268.

Booch, G. (1994*). Object-oriented analysis and design with applications*. Redwood City, Calif: Benjamin/Cummings Pub. Co...

Briand, L. C. (2008, August). Novel applications of machine learning in software testing. In *Quality Software, 2008. QSIC'08. The Eighth International Conference on* (pp. 3-10). IEEE.

Briand, L. C., Labiche, Y., & Bawar, Z. (2008, August). Using machine learning to refine black-box test specifications and test suites. In *Quality Software, 2008. QSIC'08. The Eighth International Conference on* (pp. 135-144). IEEE.

Campos, E. C., & de Almeida Maia, M. (2017, November). Common bug-fix patterns: a large-scale observational study. In *Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on* (pp. 404-413). IEEE.

Chekam, T. T., Papadakis, M., Le Traon, Y., & Harman, M. (2017, May). An empirical study on mutation, statement and branch coverage fault revelation that avoids the unreliable clean program assumption. In *Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on* (pp. 597-608). IEEE.

Demeyer, S., Verhaeghe, B., Etien, A., Anquetil, N., & Ducasse, S. (2018, March). Evaluating the efficiency of continuous testing during test-driven development. In *Validation, Analysis and Evolution of Software Tests (VST), 2018 IEEE Workshop on* (pp. 21-25). IEEE.

DeMillo, R. A., Lipton, R. J., & Sayward, F. G. (1979). *Papers on Program Testing* (No. GIT-ICS-79/04). GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUTER SCIENCE.

Derezinska, A. (2006, October). Quality assessment of mutation operators dedicated for C# programs. In *2006 Sixth International Conference on Quality Software (QSIC'06)* (pp. 227-234). IEEE.

Derezińska, A., & Szustek, A. (2007). CREAM-a System for Object-oriented Mutation of C# Programs. In *Annals Gdansk University of Technology Faculty of ETI* (Vol. 13, No. 5, pp. 389-406). Information Technology.

Derezinska, A., & Szustek, A. (2008, June). Tool-supported advanced mutation approach for verification of C# programs. In *2008 Third International Conference on Dependability of Computer Systems DepCoS-RELCOMEX* (pp. 261-268). IEEE.

Derezińska, A., & Rudnik, M. (2012, May). Quality evaluation of object-oriented and standard mutation operators applied to C# programs. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation* (pp. 42-57). Springer, Berlin, Heidelberg.

Derezińska, A., & Trzpil, P. (2015). Mutation Testing Process Combined with Test-Driven Development in. NET Environment. In *Theory and Engineering of Complex Systems and Dependability* (pp. 131-140). Springer, Cham.

Etiemble, D. (2018). 45-year CPU evolution: one law and two equations. arXiv preprint arXiv:1803.00254.

Ghiduk, A. S., Girgis, M. R., & Shehata, M. H. (2018). Reducing the Cost of Higher-Order Mutation Testing. *Arabian Journal for Science and Engineering*, 1-14.

Grounds, M., & Kudenko, D. (2005). Parallel reinforcement learning with linear function approximation. In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning* (pp. 60-74). Springer, Berlin, Heidelberg.

Guillaume, S. J. (2015). Mutant Selection Using Machine Learning Techniques. *Machine Learning: Theory and Applications*, 24.

Honfi, D., & Micskei, Z. (2019). Classifying generated white-box tests: an exploratory study. Software Quality Journal, 27(3), 1339-1380.

Huang, Q., Xia, X., & Lo, D. (2017, September). Supervised vs unsupervised models: A holistic look at effort-aware just-in-time defect prediction. In *Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on* (pp. 159-170). IEEE.

Huene, P. C., Cunningham, J. A., & Vidolov, B. V. (2011). *U.S. Patent No. 8,079,018*. Washington, DC: U.S. Patent and Trademark Office.

Huffman, C. (2014). *Windows Performance Analysis Field Guide*. Elsevier.

Hutchins, M., Foster, H., Goradia, T., & Ostrand, T. (1994, May). Experiments on the effectiveness of dataflow-and control-flow-based test adequacy criteria. In Proceedings of 16th International conference on Software engineering (pp. 191-200). IEEE.

Jalbert, K., & Bradbury, J. S. (2012, June). Predicting mutation score using source code and test suite metrics. In *Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering* (pp. 42-46). IEEE Press.

Jia, Y., & Harman, M. (2011). An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, *37*(5), 649-678.

Kirdey, S., Cureton, K., Rick, S., & Ramanathan, S. (2019). Lerner — using RL agents for test case scheduling [Web log post]. Retrieved March 5, 2020, from https://netflixtechblog.com

Kurtz Jr, R. G. (2018). *Improving Mutation Testing with Dominator Mutants* (Doctoral dissertation, George Mason University).

Lenz, A. R., Pozo, A., & Vergilio, S. R. (2013). Linking software testing results with a machine learning approach. *Engineering Applications of Artificial Intelligence*, *26*(5-6), 1631-1640.

Liaw, A., & Wiener, M. (2002). Classification and regression by RandomForest. *R news*, *2*(3), 18-22.

Lipton, R. J. (1971). Fault diagnosis of computer programs.

Lu, H., Setiono, R., & Liu, H. (1996). Effective data mining using neural networks. IEEE transactions on knowledge and data engineering, 8(6), 957-961.

Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2007, May). 'Good' organisational reasons for 'Bad' software testing: An ethnographic study of testing in a small software company. In *Proceedings of the 29th international conference on Software Engineering* (pp. 602-611). IEEE Computer Society.

Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE transactions on software engineering*, *33*(1), 2-13.

Microsoft Corporation (2013), C# Language Specification Version 5.0. Available from: http://www.microsoft.com/en-us/download/details.aspx?id=7029

Microsoft Corporation (2019), Process Explorer v16.31. Available from: https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

Microsoft Corporation (2020), Partition III: CIL Instruction Set - Microsoft Download Center. Available from: https://download.microsoft.com/download/7/3/3/733ad403-90b2-4064-a81e-01035a7fe13c/ms%20partition%20iii.pdf

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K., & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.

Namin, A. S., Andrews, J., & Murdoch, D. (2008, May). Sufficient mutation operators for measuring test effectiveness. In *2008 ACM/IEEE 30th International Conference on Software Engineering* (pp. 351-360). IEEE.

Ostrand, T. J., & Balcer, M. J. (1988). The category-partition method for specifying and generating functional tests. *Communications of the ACM*, *31*(6), 676-686.

Patterson, D. (2010). The trouble with multi-core. IEEE Spectrum, 47(7), 28-32.

Qu, X., Cohen, M. B., & Woolf, K. M. (2007, October). Combinatorial interaction regression testing: A study of test case generation and prioritization. In 2007 IEEE International Conference on Software Maintenance (pp. 255-264). IEEE.

Quinlan, J. R. (1993). *C 4.5: programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Recht, B., Re, C., Wright, S., & Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems* (pp. 693-701).

Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (2001). Prioritizing test cases for regression testing. *IEEE Transactions on software engineering*, *27*(10), 929-948.

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909-3943.

Spieker, H., Gotlieb, A., Marijan, D., & Mossige, M. (2018). Reinforcement learning for automatic test case prioritization and selection in continuous integration. *arXiv preprint arXiv:1811.04122*.

Spinellis, D., Gousios, G., Karakoidas, V., Louridas, P., Adams, P. J., Samoladas, I., & Stamelos, I. (2009). Evaluating the quality of open source software. *Electronic Notes in Theoretical Computer Science*, *233*, 5-28.

Strug, J., & Strug, B. (2012, November). Machine learning approach in mutation testing. In *IFIP International Conference on Testing Software and Systems* (pp. 200-214). Springer, Berlin, Heidelberg.

Strug, J., & Strug, B. (2017, September). Using classification for cost reduction of applying mutation testing. In *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on* (pp. 99-108). IEEE.

Strug, J., & Strug, B. (2018, June). Cost Reduction in Mutation Testing with Bytecode-Level Mutants Classification. In *International Conference on Artificial Intelligence and Soft Computing* (pp. 714-723). Springer, Cham.

Strug, J., & Strug, B. (2018, September). Evaluation of the prediction-based approach to cost reduction in mutation testing. In *International Conference on Information Systems Architecture and Technology* (pp. 340-350). Springer, Cham.

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT press.

Tillmann, N., & De Halleux, J. (2008, April). Pex–white box test generation for. net. In International conference on tests and proofs (pp. 134-153). Springer, Berlin, Heidelberg.

Vincenzi, A. M. R., Simao, A. S., Delamaro, M. E., & Maldonado, J. C. (2006). Muta-Pro: Towards the definition of a mutation testing process. *Journal of the Brazilian Computer Society*, *12*(2), 49-61.

Whittaker, J. A. (2000). What is software testing? And why is it so hard?. *IEEE software*, *17*(1), 70-79.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, J., Zhang, L., Harman, M., Hao, D., Jia, Y., & Zhang, L. (2018). Predictive mutation testing. *IEEE Transactions on Software Engineering*.