RESOLVER

BOOLEAN SATISFIABILITY SOLVER BASED ON A GENETIC ALGORITHM

Test Report

Team Imperium

Dewald de Jager
Ernst Eksteen
Vignesh Iyer
Regan Koopmans
Craig van Heerden

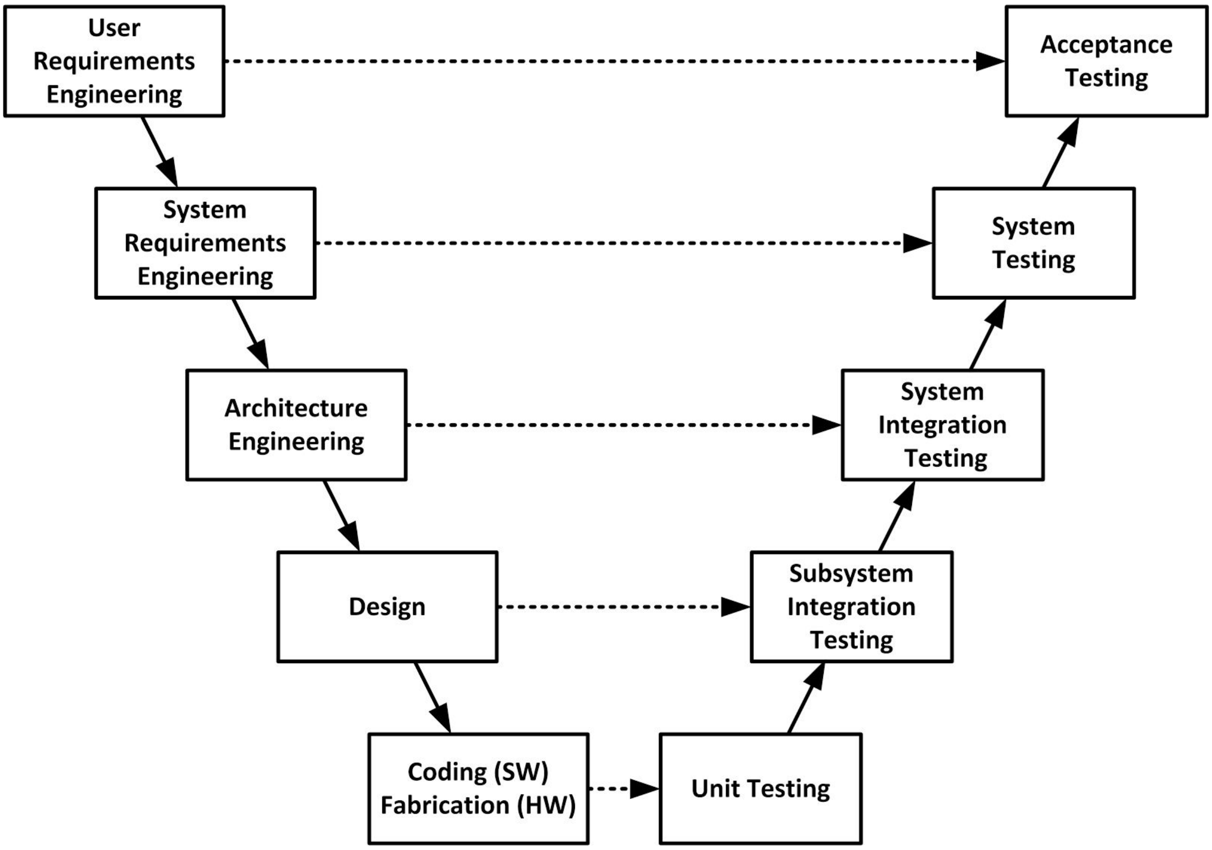
October 19, 2017

Software Testing Methodology

Software testing is a practical engineering activity that is essential to producing high-quality software. We use the software testing methodology presented in Introduction to Software Testing by Ammann and Offutt for designing test inputs, producing test case values, running test scripts and analyzing and reporting the results. The methodology places a large emphasis on treating testing activities as a lifecycle-wide activity and encourages designing tests concurrently with each of the development activities regardless of whether the software artifact is in an executable state. This allows us to identify defects in our design decisions early on in the project reducing the overall time and cost.

Test Level Descriptions

Our testing process is divided into different testing levels, where each testing level corresponds to a software development activity. This is widely known in literature as “The V Model”. This model illustrates how the testing levels relate to each of the lifecycle activities.


User Requirements Engineering
Acceptance Testing
System Requirements Engineering
System Testing
Architecture Engineering
System Integration Testing
Design
Subsystem Integration Testing
Coding (SW) Fabrication (HW)
Unit Testing

An overview of each of the testing levels is given below:

Unit Testing

Unit testing verifies that the smallest entities (units) can function correctly when isolated from the rest of the units. A unit is the smallest entity which can independently exist and is typically one or more contiguous program statements with a name that other parts of the system use to call it. In the case of Resolver, units are implemented as functions and methods. The unit test plans are developed primarily during the detailed design phase. These plans are executed to eliminate bugs at code level or unit level. This level is the lowest level of testing and is done in a very isolated manner. We make use of techniques such as input space partitioning to ensure that generalisations of the different types of cases are found, including corner cases. We also use automated testing tools to assist in running the unit tests as part of regression testing.

Integration Testing

Integration Test verify that units created and tested independently can coexist and that the interfaces between modules have consistent assumptions and communicate correctly. Since Resolver has relatively few modules, we have decided to merge the integration and module testing phases. Module testing assesses each module in isolation and verifies that the units that each module is composed of interact correctly and use the shared data structures correctly. Both of these testing activities are typically the sole responsibility of the system development team.

System Testing

System testing verifies that functional and quality requirements have been met and that the system as a whole meets its specifications. System tests plans are primarily developed during the architectural design phase. It is assumed that the different parts of the system work individually and the whole application is tested for its functionality, interdependency and communication. Load and performance testing as well as stress testing are done at this stage. This testing phase is usually done by a team other than the developers but due to the nature of this project we have assigned the roles among ourselves and dedicated time for this purpose.

User Acceptance Testing

The goal of user acceptance testing is to ensure that the software does what the user wants. User acceptance test plans are therefore primarily developed during the Requirements Engineering phase and is the testing must involve users, the product owner and/or individuals with strong domain knowledge. User acceptance testing is performed in a user environment that resembles the production environment, using realistic data. Our data for this testing purposes is acquired from the data sets of global SAT competitions and encodings from the System Specifications and Formal Methods research group at the University of Pretoria.

Test Field Descriptions

Each test case in our software testing report document is described by these fields:

1. **Test Case ID**: A unique identifier for the test case
2. **Purpose**: A short textual description of the purpose of the test case
3. **Expectation**: A short textual description of the anticipated observation for the specified input

4. **Pass Criterion**: The condition that needs to be satisfied in order for the test case to be considered as “passed”
5. **Input**: The values that were used as input for this test case
6. **Observation**: A short textual description of what was observed when the unit being tested was executed - This should be completely objective and no judgement should be made according to the pass criterion
7. **Judgement**: Whether the test case was passed or not (Yes / No)

Testing Tools

Testing is a very important part of the software development lifecycle and as such requires a lot of time and effort. To assist in maintaining high quality standards and implementing good software testing practices, we make use of a few vital tools.

The majority of the project is developed in Python and as such the unit tests have been implemented in Python. We use a **unit testing framework** called *unittest* that is provided by the Python standard library to run unit tests. This framework allows us to automate tests, share setup and shutdown code among tests, aggregate related tests into collections and maintain independence between the code of the tests and the reporting framework.

In addition to ensuring all our code abides by our **coding standards** (The PEP8 standard), the JetBrains’ *PyCharm* integrated development environment provides tools for running **code coverage** tests and reporting statistics. We have set goals of achieving 100% node coverage (Equivalent to 100% statement coverage) and reaching 90% edge coverage. Path coverage is practically impossible because of the large number of loops in Resolver resulting in a countless number of paths. For this reason, loops are manually tested using the methods suggested by Beizer in *Software Testing Techniques*.

PyCharm also includes a **profiler** that gives us insightful statistics on run-time performance which aided us in identifying bottlenecks, making optimisations and gathering statistics for stress tests. The profiler also populates a graphical **call graph** allowing us to see how many function calls were made, how long they took and the relationships between the functions.

The Travis **continuous integration tool** allows us to merge small code changes into the codebase frequently rather than merging in a single, large change at the end of the lifecycle. We hope to achieve higher-quality code by developing and testing in smaller increments. Travis allows real-time monitoring of tests as they execute, is highly configurable, can run tests concurrently and integrates nicely with our communication platforms such as Slack so that we receive updates in real-time.

Unit Tests

Many of the tests below refer to example problems by name, for example trivial.cnf. These are benchmark problems that were used throughout development and are made publicly available on our GitHub repository so that the results can be reproduced. For brevity, many parameters are omitted from the input fields; These were usually left to their defaults or set to valid values. All inputs that may affect the results of the test have been included.

Test Case ID: UT001

Name	test_sat
Purpose	Determines whether sat function successfully identifies a satisfied clause.
Expectation	The clause should be satisfiable
Pass Criterion	Sat returns true

Input	Individual = 000100000
	Clause = [9, -5]

Observation	Sat returns true
Judgement	Pass

Test Case ID: UT002

Name	test_sat
Purpose	Determines whether sat function successfully identifies a satisfied clause.
Expectation	The clause should not be satisfiable
Pass Criterion	Sat returns false

Input	Individual = 000100000
	Clause = [1, 3, 6]

Observation	Sat returns false
Judgement	Pass

Test Case ID: UT003

Name	test_sat
Purpose	Determines whether sat function successfully identifies a satisfied clause.
Expectation	The clause should not be satisfiable
Pass Criterion	Sat returns false

Input	Individual = 111111111
	Clause = [6, -4]

Observation	Sat returns false
Judgement	Pass

Test Case ID: UT004

Name	test_evaluate
Purpose	To test that the fitness function, evaluate, evaluates the fitness of an individual correctly
Expectation	The fitness value should be 1
Pass Criterion	Evaluate returns 1

Input	Formula = trivial.cnf
	Individual = 111111111

Observation	Evaluate returns 1
Judgement	Pass

Test Case ID: UT005

Name	test_evaluate
Purpose	To test that the fitness function, evaluate, evaluates the fitness of an individual correctly
Expectation	The fitness value should be 2
Pass Criterion	Evaluate returns 1

Input	Formula = trivial.cnf
	Individual = 111111110

Observation	Evaluate returns 2
Judgement	Pass

Test Case ID: UT006

Name	test_improvement

Purpose	To test that the improvement function correctly calculates the difference in fitness of an individual if one of its bits are flipped
Expectation	The difference in fitness value should be 1
Pass Criterion	Evaluate returns 1

Input	Formula = trivial.cnf
	Individual = 000100000
	Bit index = 1

Observation	Evaluate returns 1
Judgement	Pass

Test Case ID: UT007

Name	test_improvement
Purpose	To test that the improvement function correctly calculates the difference in fitness of an individual if one of its bits are flipped
Expectation	The difference in fitness value should be 1
Pass Criterion	Evaluate returns 1

Input	Formula = trivial.cnf
	Individual = 000100000
	Bit index = 6

Observation	Evaluate returns 1
Judgement	Pass

Test Case ID: UT008

Name	test_improvement
Purpose	To test that the improvement function correctly calculates the difference in fitness of an individual if one of its bits are flipped
Expectation	The difference in fitness value should be 1
Pass Criterion	Evaluate returns -1

Input	Formula = trivial.cnf
	Individual = 000101000
	Bit index = 6

Observation	Evaluate returns -1
Judgement	Pass

Test Case ID: UT009

Name	test_corrective_clause
Purpose	To test that the corrective clause crossover operator correctly produces a new candidate solution from two parent candidate solutions
Expectation	The crossover should produce a child that has all the bits in the child that can be calculated deterministically set to the correct values
Pass Criterion	Child is 00X11X000 where X can be either a 0 or 1
Input	Formula = trivial.cnf First Parent = 000111000 Second Parent = 001110000
Observation	Child is 000111000
Judgement	Pass

Test Case ID: UT010

Name	test_corrective_clause_with_truth_maintenance
Purpose	To test that the corrective clause with truth maintenance crossover operator correctly produces a new candidate solution from two parent candidate solutions
Expectation	The crossover should produce a child that has all the bits in the child that can be calculated deterministically set to the correct values
Pass Criterion	Child is 00X11X000 where X can be either a 0 or 1
Input	Formula = trivial.cnf First Parent = 000111000 Second Parent = 001110000
Observation	Child is 001111000
Judgement	Pass

Test Case ID: UT011

Name	test_fluerent_and_ferland
Purpose	To test that the Fleurent & Ferland crossover operator correctly produces a new candidate solution from two parent candidate solutions
Expectation	The crossover should produce a child that has all the bits in the child that can be calculated deterministically set to the correct values
Pass Criterion	Child is 0011X0X11 where X can be either a 0 or 1
Input	Formula = trivial.cnf First Parent = 001101011 Second Parent = 001110111
Observation	Child is 001110011
Judgement	Pass

Test Case ID: UT012

Name	test_standard_tabu_choose
Purpose	Tests whether the best possible move (index of bit to flip) for an assignment is returned (Only the improvement gain criterion is used for selection)
Expectation	Should return a list of indices that result in the maximum improvement when flipped
Pass Criterion	The indices returned are 1, 2, 4 and 6

Input	Formula = trivial.cnf
	Individual = 000100000

Observation	The indices returned are 1, 2, 4 and 6
Judgement	Pass

Test Case ID: UT013

Name	test_standard_tabu_choose
Purpose	Tests whether the best possible move (index of bit to flip) for an assignment is returned (Only the improvement gain criterion is used for selection)

Expectation	Should return a list of indices that result in the maximum improvement when flipped
Pass Criterion	The indices returned are 1, 2 and 3

Input	Formula = trivial2.cnf
	Individual = 000

Observation	The indices returned are 1, 2 and 3
Judgement	Pass

Test Case ID: UT014

Name	test_standard_tabu
Purpose	A check to determine whether a potential individual generated by crossover is improved to produce the best individual due to bit flips (tabu search intensification)
Expectation	The tabu search should produce a solution that is either the same as or an improved (Fitter) version of the child that was given as input.
Pass Criterion	The solution returned is the correct length

Input	Formula = trivial.cnf
	Individual = 000001111

Observation	Child is 000001111
Judgement	Pass

Test Case ID: UT015

Name	test_standard_tabu
Purpose	A check to determine whether a potential individual generated by crossover is improved to produce the best individual due to bit flips (tabu search intensification)
Expectation	The tabu search should produce a solution that is either the same as or an improved (Fitter) version of the child that was given as input.
Pass Criterion	The solution returned is the correct length

Input	Formula = trivial.cnf
	Individual = 111111111
	Max Flip = 0

Observation	Child is 111111111
Judgement	Pass

Test Case ID: UT016

Name	test_standard_tabu
Purpose	A check to determine whether a potential individual generated by crossover is improved to produce the best individual due to bit flips (tabu search intensification)
Expectation	The tabu search should produce a solution that is either the same as or an improved (Fitter) version of the child that was given as input.
Pass Criterion	The solution returned is the correct length

Input	Formula = trivial.cnf
	Individual = 111110111
	Max Flip = 2

Observation	Child is 111011111
Judgement	Pass

Test Case ID: UT017

Name	test_degree
Purpose	Tests to determine whether the "trueness" calculation of a clause with respect to a specifc individual represents the "count of the number of true/false atoms depending on whether the atoms were negated or not".
Expectation	The number of literals of a clause that are true for the given assignment should be returned (A non-negative integer)
Pass Criterion	The degree is 1

Input	Individual = 100100000
	Clause = [9, -5]

Observation	The degree is 1
Judgement	Pass

Test Case ID: UT018

Name	test_degree
Purpose	Tests to determine whether the "trueness" calculation of a clause with respect to a specifc individual represents the "count of the number of true/false atoms depending on whether the atoms were negated or not".
Expectation	The number of literals of a clause that are true for the given assignment should be returned (A non-negative integer)
Pass Criterion	The degree is 1

Input	Individual = 100100000
	Clause = [1, 3, 6]

Observation	The degree is 1
Judgement	Pass

Test Case ID: UT019

Name	test_degree
Purpose	Tests to determine whether the "trueness" calculation of a clause with respect to a specific individual represents the "count of the number of true/false atoms depending on whether the atoms were negated or not".
Expectation	The number of literals of a clause that are true for the given assignment should be returned (A non-negative integer)
Pass Criterion	The degree is 3

Input	Individual = 000000110
	Clause = [7, 8, -3]

Observation	The degree is 3
Judgement	Pass

Test Case ID: UT020

Name	test_weight
Purpose	Tests whether the weight value of an individual with respect to an index is equivalent to the ratio of the sum of degrees of clauses to the cardinality of clauses, where the index appears in the clause.

Expectation	The weight value returned is a floating point value representing the ratio of the sum of degrees of clauses to the cardinality of clauses containing the specified literal
Pass Criterion	The weight is 1.5

Input	Formula = trivial.cnf
	Individual = 000000000
	Index = 4

Observation	The weight is 1.5
Judgement	Pass

Test Case ID: UT021

Name	test_weight
Purpose	Tests whether the weight value of an individual with respect to an index is equivalent to the ratio of the sum of degrees of clauses to the cardinality of clauses, where the index appears in the clause.
Expectation	The weight value returned is a floating point value representing the ratio of the sum of degrees of clauses to the cardinality of clauses containing the specified literal
Pass Criterion	The weight is 1

Input	Formula = trivial.cnf
	Individual = 111111111
	Index = 4

Observation	The weight is 1
Judgement	Pass

Test Case ID: UT022

Name	test_weight
Purpose	Tests whether the weight value of an individual with respect to an index is equivalent to the ratio of the sum of degrees of clauses to the cardinality of clauses, where the index appears in the clause.
Expectation	The weight value returned is a floating point value representing the ratio of the sum of degrees of clauses to the cardinality of clauses containing the specified literal
Pass Criterion	The weight is 2

Input	Formula = trivial.cnf
	Individual = 111010101
	Index = 4

Observation	The weight is 2
Judgement	Pass

Test Case ID: UT023

Name	test_rvcf
Purpose	Tests whether the best possible move (index of bit to flip) for an assignment is returned (The improvement gain criterion is used in combination with the weight criterion in order to break ties in the number of candidate variables)
Expectation	A list of indices with the highest improvement should be returned
Pass Criterion	The only index returned is 6

Input	Formula = trivial.cnf
	Individual = 000000000

Observation	The only index returned is 6
Judgement	Pass

Test Case ID: UT024

Name	test_rvcf
Purpose	Tests whether the best possible move (index of bit to flip) for an assignment is returned (The improvement gain criterion is used in combination with the weight criterion in order to break ties in the number of candidate variables)
Expectation	A list of indices with the highest improvement should be returned
Pass Criterion	The indices returned are 1, 2 and 3

Input	Formula = trivial2.cnf
	Individual = 000

Observation	The indices returned are 1, 2 and 3
Judgement	Pass

Test Case ID: UT025

Name	test_get
Purpose	Tests whether the get method of an individual correctly retrieves an atom value from a bit-string
Expectation	The value of the bit at the specified position of the individual should be returned
Pass Criterion	The bit value returned is 1

Input	Individual = 011010010
	Index = 2

Observation	The bit value returned is 1
Judgement	Pass

Test Case ID: UT026

Name	test_get
Purpose	Tests whether the get method of an individual correctly retrieves an atom value from a bit-string
Expectation	The value of the bit at the specified position of the individual should be returned
Pass Criterion	The bit value returned is 0

Input	Individual = 011010010
	Index = 9

Observation	The bit value returned is 0
Judgement	Pass

Test Case ID: UT027

Name	test_get
Purpose	Tests whether the get method of an individual correctly retrieves an atom value from a bit-string
Expectation	The value of the bit at the specified position of the individual should be returned

Pass Criterion	The bit value returned is 1

Input	Individual = 1
	Index = 1

Observation	The bit value returned is 1
Judgement	Pass

Test Case ID: UT028

Name	test_get
Purpose	Tests whether the get method of an individual correctly retrieves an atom value from a bit-string
Expectation	The index is detected to be out of range so None is returned
Pass Criterion	The bit value returned is None

Input	Individual = 011010010
	Index = 10

Observation	The bit value returned is None
Judgement	Pass

Test Case ID: UT029

Name	test_set
Purpose	Tests whether the set method of an individual correctly sets an atom to the specified value
Expectation	The value of the bit at the specified position of the individual should be set to the specified value
Pass Criterion	The individual is changed to 001010010

Input	Individual = 011010010
	Index = 2
	Value = 0

Observation	The individual is changed to 001010010
Judgement	Pass

Test Case ID: UT030

Name	test_set
Purpose	Tests whether the set method of an individual correctly sets an atom to the specified value
Expectation	The value of the bit at the specified position of the individual should be set to the specified value
Pass Criterion	The individual is changed to 011010010

Input	Individual = 011010010
	Index = 2
	Value = 1

Observation	The individual is changed to 011010010
Judgement	Pass

Test Case ID: UT031

Name	test_set
Purpose	Tests whether the set method of an individual correctly sets an atom to the specified value
Expectation	The individual is left unchanged as the index is out of bounds
Pass Criterion	The individual is not modified

Input	Individual = 011010010
	Index = 10
	Value = 1

Observation	The individual is not modified
Judgement	Pass

Test Case ID: UT032

Name	test_set
Purpose	Tests whether the set method of an individual correctly sets an atom to the specified value
Expectation	The individual is left unchanged as the value is invalid

Pass Criterion	The individual is not modified

Input	Individual = 011010010
	Index = 2
	Value = ‘x’

Observation	The individual is not modified
Judgement	Pass

Test Case ID: UT033

Name	test_flip
Purpose	Tests whether the flip method of an individual correctly negates an atom at a specified index
Expectation	The value of the atom at the specified index set to 1 if it is 0 and set to 0 if it is 1
Pass Criterion	The individual changed to 010010010

Input	Individual = 011010010
	Index = 3

Observation	The individual changed to 010010010
Judgement	Pass

Test Case ID: UT034

Name	test_flip
Purpose	Tests whether the flip method of an individual correctly negates an atom at a specified index
Expectation	The value of the atom at the specified index set to 1 if it is 0 and set to 0 if it is 1
Pass Criterion	The individual changed to 010010010

Input	Individual = 011010010
	Index = 3

Observation	The individual changed to 010010010
Judgement	Pass

Test Case ID: UT035

Name	test_flip
Purpose	Tests whether the flip method of an individual correctly negates an atom at a specified index
Expectation	The value of the atom at the specified index set to 1 if it is 0 and set to 0 if it is 1
Pass Criterion	The individual changed to 111010010

Input	Individual = 011010010
	Index = 1

Observation	The individual changed to 111010010
Judgement	Pass

Test Case ID: UT036

Name	test_flip
Purpose	Tests whether the flip method of an individual correctly negates an atom at a specified index
Expectation	The individual remains unchanged as the index is out of bounds
Pass Criterion	The individual remains unmodified

Input	Individual = 011010010
	Index = 10

Observation	The individual remains unmodified
Judgement	Pass

Test Case ID: UT037

Name	test_flip
Purpose	Tests whether the flip method of an individual correctly negates an atom at a specified index
Expectation	The individual remains unchanged as the index is out of bounds
Pass Criterion	The individual remains unmodified

Input	Individual = 011010010
	Index = 0

Observation	The individual remains unmodified
Judgement	Pass

Test Case ID: UT038

Name	test_flip
Purpose	Tests whether the flip method of an individual correctly negates an atom at a specified index
Expectation	The individual remains unchanged as the index is out of bounds
Pass Criterion	The individual remains unmodified

Input	Individual = 011010010
	Index = -1

Observation	The individual remains unmodified
Judgement	Pass

Test Case ID: UT039

Name	test_create
Purpose	Tests whether the factory creates lists of individuals that are of the correct dimensions and implementation method
Expectation	The population is initialised with the right amount of individuals and the individuals are of the correct size
Pass Criterion	The population contains 50 individuals that are all 10 bits long
Input	Population Size = 50 Length = 10
Observation	The population contains 50 individuals that are all 10 bits long
Judgement	Pass

Integration Tests

Test Case ID: IT001

Name	test_push_to_all
Purpose	Creates two test clients and calls the server's push_to_all() function. The test then checks that the message received by the clients is the same message that was sent by the server.
Expectation	The message sent by the server is received correctly by both clients
Pass Criterion	Both clients receive the message correctly
Input	Message = “Test Message#”
Observation	Both clients receive “Test Message#” correctly
Judgement	Pass

Test Case ID: IT002

Name	test_push_to_one
Purpose	Creates two test clients and sends both unique messages. The test then checks that each client received the correct message.
Expectation	Each client receives the correct message and the messages are received intact
Pass Criterion	Client one receives the first message correctly and client two receives the second message
Input	Message 1 = “Test Message 1#” Message 2 = “Test Message 2#”
Observation	Client one receives “Test Message 1#” correctly and client two receives “Test Message 2#” correctly
Judgement	Pass

Test Case ID: IT003

Name	test_address_in_use_exception
Purpose	Tests if the correct exception is thrown if the requested socket is already

	in use.
Expectation	An exception should be thrown and the server should terminate
Pass Criterion	A Socket Exception is thrown and the server terminates gracefully with a helpful error message
Input	N/A
Observation	The server throws the exception but does not terminate.
Judgement	Fail

Test Case ID: IT004

Name	test_process_message_from_client
Purpose	Tests if the message received from the client is correct and processed correctly.
Expectation	The server receives the message from the client correctly
Pass Criterion	The server receives the message from the client correctly
Input	Message = “Test Message#”
Observation	The server receives “Test Message#”
Judgement	Pass

Test Case ID: IT005

Name	test_get_port
Purpose	Test that the correct port of the server is returned.
Expectation	The correct port number that was requested by the user was used and is returned
Pass Criterion	The port number is 55555
Input	Command line argument “--port 55555” is used
Observation	55555 is returned
Judgement	Pass

Test Case ID: IT006

Name	test_close
Purpose	Tests that the server's socket is properly closed and that all clients threads are joined.
Expectation	The server terminates gracefully and the socket is released so that it can be used by other applications
Pass Criterion	The return value is true
Input	N/A
Observation	Close returns true
Judgement	Pass

System Tests

Test Case ID: ST001

Purpose	To test that the system does not find a solution for an unsatisfiable formula
Expectation	The system should attempt to solve the formula but does not find a solution within the maximum number of generations
Pass Criterion	No solution is found and the system terminates when the maximum number of generations is reached
Input	trivial3.cnf
Observation	The system went through all 100 generations and did not find a solution.
Judgement	Pass

Test Case ID: ST002

Purpose	To test that the system behaves correctly when trying to solve a satisfiable formula
Expectation	The system should start solving the formula and stop when a solution is found or the maximum number of generations is reached. The progress should be shown in real-time via the visualizations

Pass Criterion	The system finds a valid solution or reaches the maximum number of generations and terminates gracefully
Input	Problem = trivial.cnf Maximum Generations = 5
Observation	The solution is found in 1 generation
Judgement	Pass

Test Case ID: ST003

Purpose	To test that the system behaves correctly when trying to solve a satisfiable formula
Expectation	The system should start solving the formula and stop when a solution is found or the maximum number of generations is reached. The progress should be shown in real-time via the visualizations
Pass Criterion	The system finds a valid solution or reaches the maximum number of generations and terminates gracefully
Input	Problem = trivial5.cnf Maximum Generations = 5
Observation	The maximum number of generations elapses, the user is notified that a solution was not found and the solving process stops
Judgement	Pass

Test Case ID: ST004

Purpose	To test that the system behaves correctly when trying to solve a satisfiable formula
Expectation	The system should start solving the formula and stop when a solution is found or the maximum number of generations is reached. The progress should be shown in real-time via the visualizations
Pass Criterion	The system finds a valid solution or reaches the maximum number of generations and terminates gracefully
Input	Problem = trivial5.cnf Maximum Generations = 500
Observation	The solution is found in 36 generations
Judgement	Pass

User Acceptance Tests

Test Case ID: AT001

Purpose	To test that the system proceeds when a valid DIMACS file is used as input
Expectation	We assume the parameters are all valid. The system should proceed to attempting to solve the formula and no error message is displayed
Pass Criterion	The system attempts to solve the formula
Input	trivial.cnf
Observation	The system successfully finds a solution for the problem
Judgement	Pass

Test Case ID: AT002

Purpose	To test that the system does not proceed when an invalid DIMACS formula is used as input
Expectation	We assume the parameters are all valid. The system should display an error message stating that the input is invalid as it does not conform to the DIMACS format
Pass Criterion	An error message is displayed
Input	p cnf 9 5 9 -5 0 1 3 6 0 2 -4 x 0 7 8 -3 0 -6 -4 0
Observation	Gave an error message and the elapsed time kept running
Judgement	Pass, but the error message needs to be improved

Test Case ID: AT003

Purpose	To test that the system does not find a solution for an unsatisfiable formula
Expectation	The system should attempt to solve the formula but does not find a

	solution within the maximum number of generations
Pass Criterion	No solution is found and the system terminates when the maximum number of generations is reached
Input	p cnf 3 6 1 0 -1 0 2 0 -2 0 3 0 -3 0
Observation	The system went through all 100 generations and did not find a solution.
Judgement	Pass

Test Case ID: AT004

Purpose	To test that an invalid population size is not accepted by the system
Expectation	The system should display an error message stating that the population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Population Size = 0 trivial.cnf
Observation	An error message displayed but it was not the error we were expecting
Judgement	Fail

Test Case ID: AT005

Purpose	To test that an invalid population size is not accepted by the system
Expectation	The system should display an error message stating that the population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Population Size = -1
Observation	An error message displayed but it was not the error we were expecting

Judgement	Fail

Test Case ID: AT006

Purpose	To test that an invalid sub-population size is not accepted by the system
Expectation	The system should display an error message stating that the sub-population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Sub-Population Size = 0 Population Size = 50
Observation	An error message stating that the sub-population size must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT007

Purpose	To test that an invalid sub-population size is not accepted by the system
Expectation	The system should display an error message stating that the sub-population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Sub-Population Size = -1 Population Size = 50
Observation	An error message stating that the sub-population size must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT008

Purpose	To test that an invalid sub-population size is not accepted by the system
Expectation	The system should display an error message stating that the sub-population size is invalid. The system should not start attempting to

	solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Sub-Population Size = 51 Population Size = 50
Observation	An error message displayed but it was not the error we were expecting
Judgement	Fail

Test Case ID: AT009

Purpose	To test that an invalid tabu list length is not accepted by the system
Expectation	The system should display an error message stating that the tabu list length is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Tabu List Length = 0
Observation	An error message stating that the tabu list length must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT010

Purpose	To test that an invalid tabu list length is not accepted by the system
Expectation	The system should display an error message stating that the tabu list length is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Tabu List Length = -1
Observation	An error message stating that the tabu list length must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT011

Purpose	To test that an invalid maximum flips parameter is not accepted by the system
Expectation	The system should display an error message stating that the maximum flips parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Maximum Flips = 0
Observation	An error message stating that the maximum flips parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT012

Purpose	To test that an invalid maximum flips parameter is not accepted by the system
Expectation	The system should display an error message stating that the maximum flips parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Maximum Flips = -1
Observation	An error message stating that the maximum flips parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT013

Purpose	To test that an invalid max false parameter is not accepted by the system
Expectation	The system should display an error message stating that the max false parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error

	message
Input	Max False = 0
Observation	An error message stating that the max false parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT014

Purpose	To test that an invalid max false parameter is not accepted by the system
Expectation	The system should display an error message stating that the max false parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Max False = -1
Observation	An error message stating that the max false parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT015

Purpose	To test that an invalid recursion count parameter is not accepted by the system
Expectation	The system should display an error message stating that the recursion count parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Recursion Count = 0
Observation	An error message stating that the recursion count parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT016

Purpose	To test that an invalid recursion count parameter is not accepted by the system
Expectation	The system should display an error message stating that the recursion count parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Recursion Count = -1
Observation	An error message stating that the recursion count parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT017

Purpose	To test that an invalid flip constraint parameter is not accepted by the system
Expectation	The system should display an error message stating that the flip constraint parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Flip Constraint = 0
Observation	An error message stating that the flip constraint parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT018

Purpose	To test that an invalid flip constraint parameter is not accepted by the system
Expectation	The system should display an error message stating that the flip constraint parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error

	message
Input	Flip Constraint = -1
Observation	An error message stating that the flip constraint parameter must be greater than 0 is shown
Judgement	Pass

Test Case ID: AT019

Purpose	To test that the system does not find a solution for an unsatisfiable formula
Expectation	The system should attempt to solve the formula but does not find a solution within the maximum number of generations
Pass Criterion	No solution is found and the system terminates when the maximum number of generations is reached
Input	Max Generations = 20000 p cnf 3 6 1 0 -1 0 2 0 -2 0 3 0 -3 0
Observation	The system went through 4415 generations and then the interface froze.
Judgement	Fail

Test Case ID: AT020

Purpose	To test that an invalid population size is not accepted by the system
Expectation	The system should display an error message stating that the population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Population Size = 7.3
Observation	An error message displayed but it was not the error we were expecting
Judgement	Fail

Test Case ID: AT021

Purpose	To test that an invalid sub-population size is not accepted by the system
Expectation	The system should display an error message stating that the sub-population size is the wrong type or invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Sub-Population Size = 7.3 Population Size = 50
Observation	The system successfully solves the problem
Judgement	Fail

Test Case ID: AT022

Purpose	To test that an invalid tabu list length is not accepted by the system
Expectation	The system should display an error message stating that the tabu list length is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Tabu List Length = 7.3
Observation	An error message stating that the tabu list length specified is not a valid integer literal is shown
Judgement	Pass

Test Case ID: AT023

Purpose	To test that an invalid maximum flips parameter is not accepted by the system
Expectation	The system should display an error message stating that the maximum flips parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message

Input	Maximum Flips = 7.3
Observation	The system successfully solves the problem
Judgement	Fail

Test Case ID: AT024

Purpose	To test that an invalid max false parameter is not accepted by the system
Expectation	The system should display an error message stating that the max false parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Max False = 7.3
Observation	
Judgement	

Test Case ID: AT025

Purpose	To test that an invalid recursion count parameter is not accepted by the system
Expectation	The system should display an error message stating that the recursion count parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Recursion Count = 7.3
Observation	
Judgement	

Test Case ID: AT026

Purpose	To test that an invalid flip constraint parameter is not accepted by the system

Expectation	The system should display an error message stating that the flip constraint parameter is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Flip Constraint = 7.3
Observation	
Judgement	

Test Case ID: AT027

Purpose	To test that the system is able to accept the input formula in the form of an external file
Expectation	The system should successfully read the formula from the file and attempt to solve it with no error messages
Pass Criterion	The formula is read from the external file and the system proceeds to the next step
Input	trivial.cnf
Observation	The system reads the input file and successfully finds a solution for the formula
Judgement	Pass

Test Case ID: AT028

Purpose	To test that the system is able to accept the input formula in via direct entry through the interface
Expectation	The system should successfully read the formula from the interface and attempt to solve it with no error messages
Pass Criterion	The formula is read in from the interface and the system proceeds to the next step
Input	p cnf 9 5 9 -5 0 1 3 6 0 2 -4 6 0 7 8 -3 0 -6 -4 0

Observation	The system reads the input and successfully finds a solution for the formula
Judgement	Pass

Test Case ID: AT029

Purpose	To test that the system allows for a purely text-based control and display mechanism
Expectation	The text-based interface should correctly communicate with the back-end and transmit all necessary information so that a problem can be solved
Pass Criterion	A formula can be specified and sent to the solver from the text-based interface. The result should also be shown.
Input	trivial.cnf
Observation	The text-based interface connects to the back-end and submits the formula for solving. The formula is successfully solved.
Judgement	Pass

Test Case ID: AT030

Purpose	To test that the system allows for a graphical desktop interface
Expectation	The graphical desktop interface should correctly communicate with the back-end and transmit all necessary information so that a problem can be solved
Pass Criterion	A formula can be specified and sent to the solver from the graphical desktop interface. The result should also be shown.
Input	trivial.cnf
Observation	The graphical desktop interface connects to the back-end and submits the formula for solving. The formula is successfully solved.
Judgement	Pass

Test Case ID: AT031

Purpose	To test that the system allows for a web browser-based interface
Expectation	The web browser-based interface should correctly communicate with the

	back-end and transmit all necessary information so that a problem can be solved
Pass Criterion	A formula can be specified and sent to the solver from the web browser-based interface. The result should also be shown.
Input	trivial.cnf
Observation	The web browser-based interface connects to the back-end and submits the formula for solving. The formula is successfully solved.
Judgement	Pass

Test Case ID: AT032

Purpose	To test that the system reports a complete solution if one is found
Expectation	The system should terminate iteration if a solution is found
Pass Criterion	The system has finds a solution and does not go through all the generations
Input	trivial.cnf
Observation	The system finds a solution in one generation and immediately reports it
Judgement	Pass

Test Case ID: AT033

Purpose	To test that the system reports the best partial solution found in all generations if a solution could not be found
Expectation	The system should reach the maximum number of generations without finding a solution and then report the solution with the lowest fitness
Pass Criterion	A partial solution is reported if the system reaches the maximum number of generations
Input	hgen2-a.cnf
Observation	The solving process stops and a solution is reported and its fitness is shown to be greater than zero
Judgement	Pass

Test Case ID: AT034

Purpose	To test that the system allows for spontaneous termination of the solving process
Expectation	If the Stop button is pressed while the algorithm is running, it should stop immediately and report the best partial solution found prior to termination
Pass Criterion	The solving process terminates when the Stop button is pressed
Input	hgen2-a.cnf
Observation	The solving process stopped and the best partial solution found was reported
Judgement	Pass

Test Case ID: AT035

Purpose	To test that an invalid population size is not accepted by the system
Expectation	The system should display an error message stating that the population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Population Size = 1 trivial.cnf
Observation	
Judgement	

Test Case ID: AT036

Purpose	To test that an invalid sub-population size is not accepted by the system
Expectation	The system should display an error message stating that the sub-population size is invalid. The system should not start attempting to solve the formula.
Pass Criterion	The system does not attempt to solve the formula and displays an error message
Input	Sub-Population Size = 1 trivial.cnf

Observation	
Judgement	

Traceability Matrix

	G1	G1.1	G1.2	G1.3	G2	G2.1	G2.2	G2.3	G3
AT001	**X**	**X**	**X**	**X**					
AT002	**X**	**X**	**X**	**X**					
AT003									
AT004									
AT005									
AT006									
AT007									
AT008									
AT009									
AT010									
AT011									
AT012									
AT013									
AT014									
AT015									
AT016									
AT017									
AT018									
AT019									
AT020									
AT021									
AT022									

AT023									
AT024									
AT025									
AT026									
AT027	**X**	**X**							
AT028	**X**		**X**						
AT029					**X**	**X**			**X**
AT030					**X**		**X**		
AT031					**X**			**X**	
AT032									
AT033									
AT034									
AT035									
AT036									

	G4	G5	G5.1	G5.2	G6	G7	G7.1	G7.2	G7.3
AT001									
AT002									
AT003									
AT004			**X**						
AT005			**X**						
AT006			**X**						
AT007			**X**						
AT008			**X**						
AT009			**X**						
AT010			**X**						
AT011			**X**						

AT012			X						
AT013			X						
AT014			X						
AT015			X						
AT016			X						
AT017			X						
AT018			X						
AT019									
AT020			X						
AT021			X						
AT022			X						
AT023			X						
AT024			X						
AT025			X						
AT026			X						
AT027									
AT028									
AT029									
AT030									
AT031									
AT032									
AT033									
AT034									
AT035			X						
AT036			X						

	G8	G9	G10.1	G10.2	G11	G12	G13	G14	G14.1

AT001	**X**								
AT002									
AT003	**X**								
AT004									
AT005									
AT006									
AT007									
AT008									
AT009									
AT010									
AT011									
AT012									
AT013									
AT014									
AT015									
AT016									
AT017									
AT018									
AT019									
AT020									
AT021									
AT022									
AT023									
AT024									
AT025									
AT026									
AT027									
AT028									

AT029									
AT030									
AT031									
AT032								**X**	**X**
AT033								**X**	
AT034									
AT035									
AT036									

	G14.2	G14.3	G15	G15.1	G15.2	G15.3	G16
AT001							
AT002							
AT003							
AT004							
AT005							
AT006							
AT007							
AT008							
AT009							
AT010							
AT011							
AT012							
AT013							
AT014							
AT015							
AT016							
AT017							

AT018							
AT019							
AT020							
AT021							
AT022							
AT023							
AT024							
AT025							
AT026							
AT027							
AT028							
AT029							
AT030							
AT031							
AT032							
AT033	**X**	**X**					
AT034							**X**
AT035							
AT036							

Remarks

In this section, we discuss:

- Any open issues
- Critical remarks, such as the issue of non-determinism in some test cases
- Shortcomings of our test cases
- Why we have X amount of test cases. Why not more? Why did we stop testing?

Our project has drawn heavily on the GASAT research paper by *Lardeaux, et al*. Although this paper is of high quality, it is not without error or ambiguity. There are a number of places where we have used our own discretion in particular algorithms within the software.

Due to the non-deterministic stochastic nature of the genetic algorithm, many possible tests are made void or otherwise invalid. Testing solving on an example SAT problem, for instance, cannot be a repeatable test. The outcome of solving differs greatly on parameters and the random initial state of the population. To this end, we can only prove that the mechanisms of our genetic algorithm are correct, and not that they repeatedly find a solution to a given problem.

Our unit-tests necessarily test only small fraction of the possible domain of inputs to our software. Although these tests greatly increase our confidence in the correctness of our implementation, they do not illustrate the absence of bugs.

In the current state of the software project we have 85 tests. This number of tests has arisen organically.