

# Configuration Manual

MSc Research Project  
Cloud Computing

Sheffy Batra  
Student ID: x22121790

School of Computing  
National College of Ireland

Supervisor: Dr. Punit Gupta

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Sheffy Batra
<b>Student ID:</b>	x22121790
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2023
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Dr. Punit Gupta
<b>Submission Due Date:</b>	14/12/2023
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	699
<b>Page Count:</b>	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	
<b>Date:</b>	13th December 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Sheffy Batra  
x22121790

## 1 Introduction

This report shows the step-by-step procedure to implement the below experiment steps and the configurations that are used to run the project. The cloudsim environment and its dependencies with various parameters are documented in the below sections for successfully running this project.

## 2 Prerequisite

The project is implemented on a local laptop with a specification of a 64-bit operating system, x64-based processor, and Windows 11 with 16.0 GB RAM. The Table 1 2 consist of the packages that needs to be downloaded and are used in the project.

**Eclipse IDE** : It is used as a platform for Cloudsim.

**Cloudsim**: Simulator is used to run the algorithms and verify the results.

Packages	Version	Resource Links
Eclipse	2023-03	<a href="https://www.eclipse.org/downloads/packages/release/2023-03/r">https://www.eclipse.org/downloads/packages/release/2023-03/r</a>
Cloudsim Zip	3.0	<a href="https://github.com/Cloudslab/cloudsim/releases">https://github.com/Cloudslab/cloudsim/releases</a>
Java JDK	17.0.4	<a href="https://www.oracle.com/java/technologies/downloads/#java17">https://www.oracle.com/java/technologies/downloads/#java17</a>
Common Math library	3.2	<a href="https://mvnrepository.com/artifact/org.apache.commons/commons-math3/3.2">https://mvnrepository.com/artifact/org.apache.commons/commons-math3/3.2</a>

Table 1: Downloadable Pacakges List

## 3 Cloudsim Parameter Configurations

To simulate different numbers of VMs and hosts, the parameter is set in the Random-Constant.java file.(Figure1)

```

1 package org.cloudbus.cloudsim.examples.power.random;
2
3 /**
4  * If you are using any algorithms, policies or workload included in the power package pl
5  * the following paper:
6  *
7  * Anton Beloglazov, and Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Ada
8  * Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machi
9  * Cloud Data Centers", Concurrency and Computation: Practice and Experience, ISSN: 1532-
10  * Press, New York, USA, 2011, DOI: 10.1002/cpe.1867
11  *
12  * @author Anton Beloglazov
13  * @since Jan 5, 2012
14  */
15 public class RandomConstants {
16
17     public final static int NUMBER_OF_VMS = 50;
18
19     public final static int NUMBER_OF_HOSTS = 50;
20
21     public final static long CLOUDLET_UTILIZATION_SEED = 1;
22
23 }
24

```

Figure 1: RandomConstants.java

The proposed algorithm has various parameters mutationRate, tournamentSize, and elitism which are set in the GA.java file of the algorithm (Figure(2))

```

2 * GA.java
3
4 package org.cloudbus.cloudsim.ga_c;
5
6 import org.cloudbus.cloudsim.Cloudlet;
7
8 public class GA {
9
10     /* GA parameters */
11     private static final double mutationRate = 0.15;
12     private static final int tournamentSize = 5;
13     private static final boolean elitism = true;
14
15     // Evolves a population over one generation
16     public static Population evolvePopulation(Population pop) {
17         Population newPopulation = new Population(pop.populationSize(), false);
18
19         // Keep our best individual if elitism is enabled
20         int elitismOffset = 0;
21
22
23
24

```

Figure 2: GA.java

The simulation parameters of VM and Host Types and configuration of VM's and Host are stored in Constants.java (Figure 3)

```

30
31
32  /*
33  * VM instance types:
34  * High-Memory Extra Large Instance: 3.25 EC2 Compute Units, 8.55 GB // too much MIPS
35  * High-CPU Medium Instance: 2.5 EC2 Compute Units, 0.85 GB
36  * Extra Large Instance: 2 EC2 Compute Units, 3.75 GB
37  * Small Instance: 1 EC2 Compute Unit, 1.7 GB
38  * Micro Instance: 0.5 EC2 Compute Unit, 0.633 GB
39  * We decrease the memory size two times to enable oversubscription
40
41  */
42  public final static int[] VM_MIPS = { 2500, 2000, 1000, 1500 };
43  public final static int[] VM_PES = { 1, 1, 1, 1 };
44  public final static int[] VM_RAM = { 870, 1740, 1740, 613 };
45  public final static int VM_BW = 100000; // 100 Mbit/s
46  public final static int VM_SIZE = 2500; // 2.5 GB
47
48
49  /*
50  * Host types:
51  * HP ProLiant ML110 G4 (1 x [Xeon 3040 1860 MHz, 2 cores], 4GB)
52  * HP ProLiant ML110 G5 (1 x [Xeon 3075 2660 MHz, 2 cores], 4GB)
53  * We increase the memory size to enable over-subscription (x4)
54  */
55  public final static int HOST_TYPES = 2;
56  public final static int[] HOST_MEPS = { 1860, 2660 };
57  public final static int[] HOST_PES = { 2, 2 };
58  public final static int[] HOST_RAM = { 4096, 4096 };
59  public final static int HOST_BW = 1000000; // 1 Gbit/s
60  public final static int HOST_STORAGE = 1000000; // 1 GB
61
62
63  public final static PowerModel[] HOST_POWER = {
64      new PowerModelSpecPowerHpProLiantMl110G4Xeon3040(),
65      new PowerModelSpecPowerHpProLiantMl110G5Xeon3075()
66  };

```

Figure 3: Constants.java

By configuring all these parameters the simulation environment is created and the algorithms are tested for efficiency.

## 4 Code configuration and simulation Files

The code is developed in java and algorithms that are used is a combination of Genetic and Game theory. Figure 4 represent the pseudo code of the algorithm

---

**Algorithm 1** Game Genetic VM Allocation

---

```
0: Initialize Population:
0:   Generate an initial population of VM allocations randomly.
0: for Generation  $\leftarrow$  1 to MAX_GENERATIONS do
0:   Evaluate Fitness:
0:     Calculate the fitness values for each individual in the population based on the
0:     payoff function.
0:   Tournament Selection:
0:     Perform tournament selection to choose two parents based on their fitness values.
0:   Nash Equilibrium-inspired Crossover:
0:     Calculate neighbor payoffs.
0:     Apply Nash equilibrium-inspired crossover to create offspring.
0:     Determine crossover points based on payoffs and perform crossover with a certain
0:     probability.
0:   Mutation:
0:     Apply mutation to the offspring with a certain probability.
0:     Randomly select a VM and flip its allocation.
0:   Replace Population:
0:     Replace the old population with the new population (parents and offspring).
0:   Apply Nash Equilibrium-inspired Modifications:
0:     Apply Nash equilibrium-inspired modifications to the population.
0:     Consider the payoff of neighbors and update each VM allocation based on neigh-
0:     bor payoffs.
0:   Find Best Allocation:
0:     Identify the index of the best individual in the population based on fitness
0:     values.
0:   Print Result:
0:     Output the best VM allocation in the current generation.
0: end for
0: End Algorithm =0
```

---

Figure 4: Game-Genetic Code Files

The code is basically defined in below 4 Files (Figure 5)

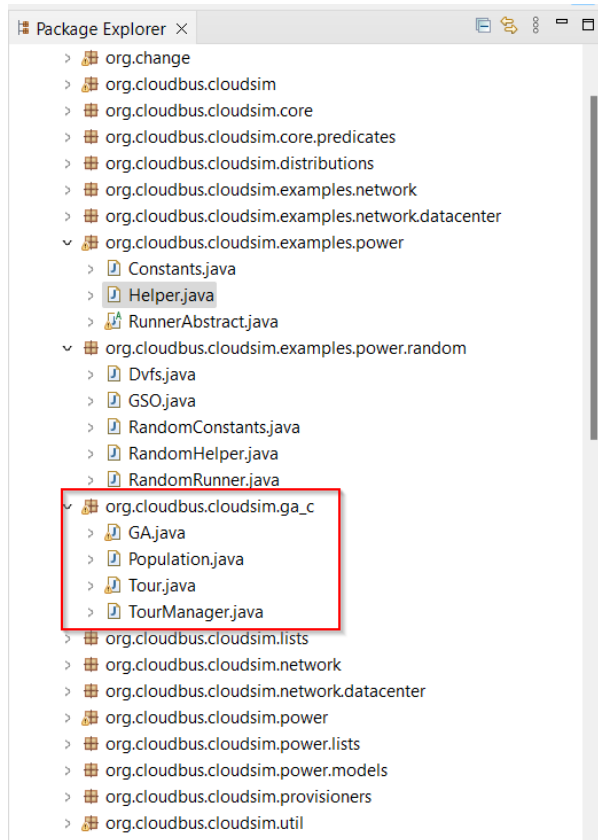


Figure 5: Game-Genetic Code Files

The Simulation is run and main function is in GSO.java file (Figure 6)

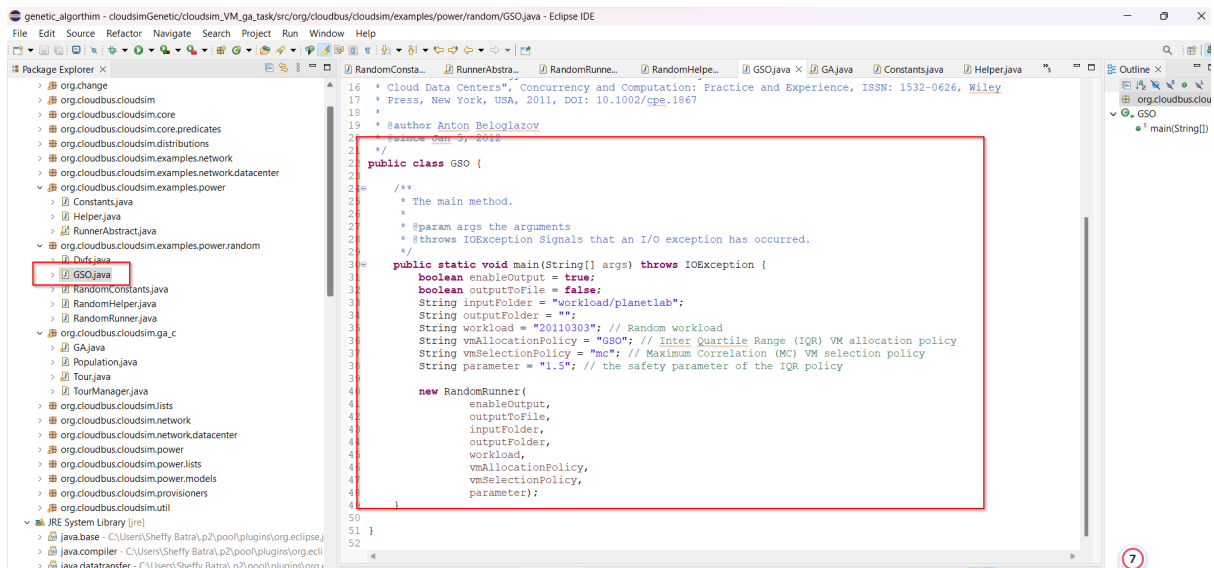


Figure 6: Main Simulation File

## 5 Test Results

The results are simulated in cloudsim and are shown below

1. Proposed Game Genetic Algorithm -GSO.java (Figure 7)
2. Genetic algorithm -Genetic.java (Figure 8)
3. IQRC algorithm- IQMC.java (Figure 9 )
4. MAD algorithm - MADMc.java (Figure 10)

The figures show the simulation for 50 VM and 50 Host. Similalry the algorithms are tested for 100, 200, 300, 400 VM's and the results and graphs are represented in the report.

The Parameters used to evaluate the algorithm are

- No. of VM Migration
- Energy Consumption
- Total Execution Time
- Standard deviation Time

The results show that the proposed Game-Genetic Algorithm is better than other algorithms.

```
Experiment name: 20110303_GSO_mmt_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 34.08 kWh
Number of VM migrations: 1759
Experiment name: 20110303_GSO_mmt_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 34.08 kWh
Number of VM migrations: 1759
SLA: 0.066918
SLA perf degradation due to migration: 0.163
SLA time per active host: 42.67%
Overall SLA violation: 12.78%
Average SLA violation: 19.56%
Number of host shutdowns: 435
Mean time before a host shutdown: 2312.49 sec
StDev time before a host shutdown: 3043.50 sec
Mean time before a VM migration: 16.86 sec
StDev time before a VM migration: 7.40 sec
Execution time - VM selection mean: 0.00007 sec
Execution time - VM selection stDev: 0.00026 sec
Execution time - host selection mean: 0.00073 sec
Execution time - host selection stDev: 0.00263 sec
Execution time - VM reallocation mean: 0.00113 sec
Execution time - VM reallocation stDev: 0.00073 sec
Execution time - total mean: 0.00397 sec
Execution time - total stDev: 0.00284 sec
```

Figure 7: Proposed Game-Genetic Algorithm



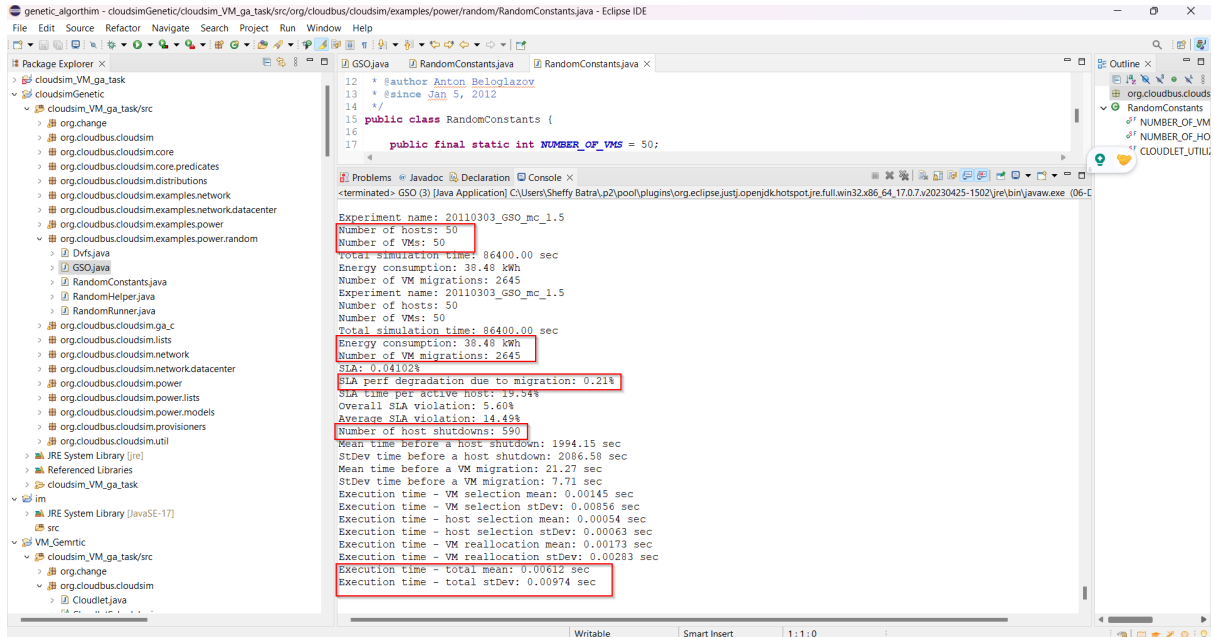


Figure 8: Genetic Algorithm

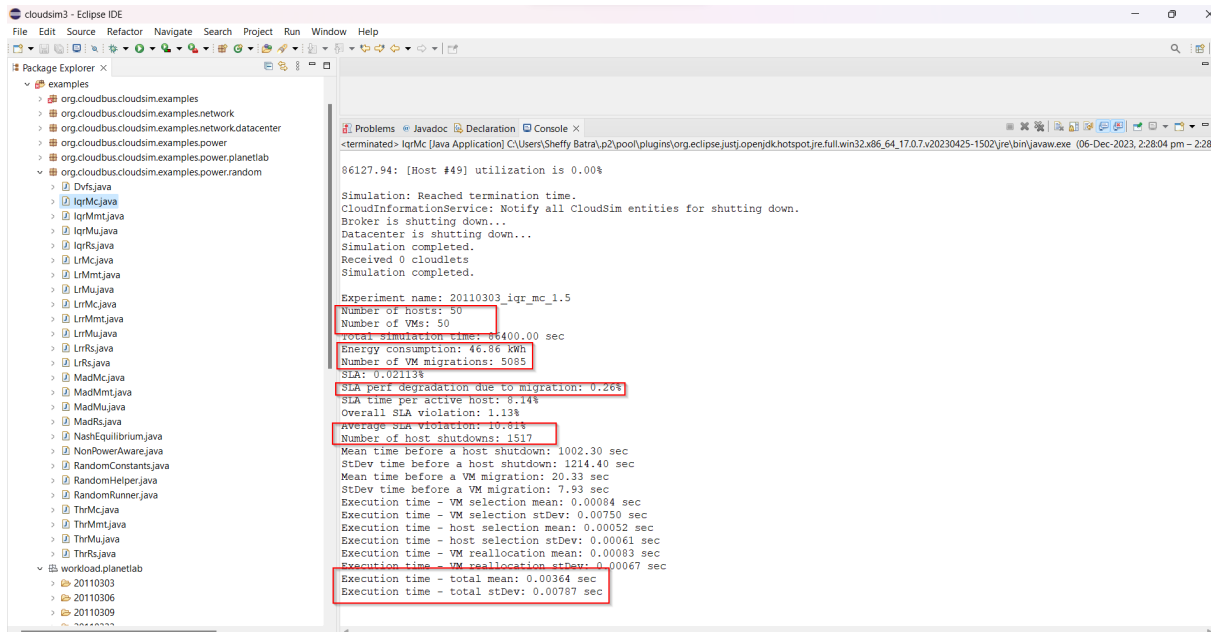


Figure 9: IQMC Algorithm

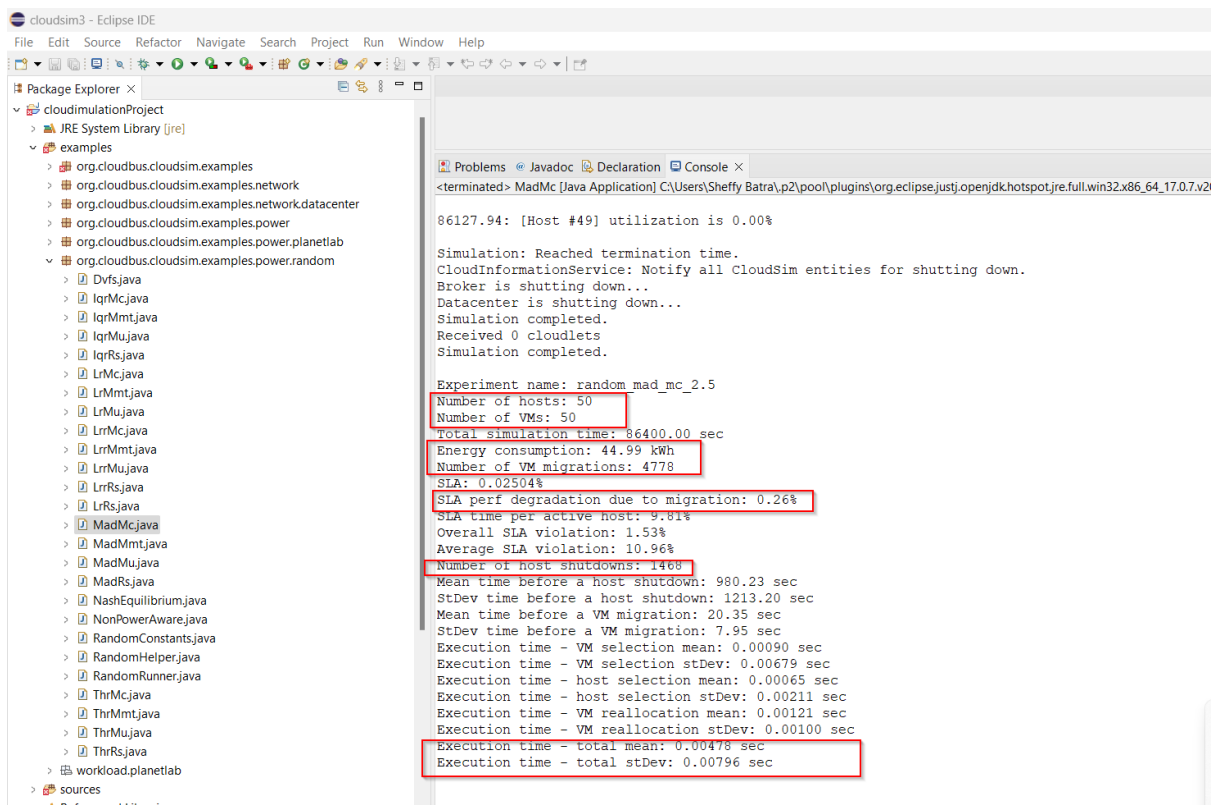


Figure 10: MAD Algorithm