A Comparative Analysis of Execution Times of Two Python Algorithms for a Virtual Robot

Mark Henry Dsouza
Masters Student - Advanced Robotics
University of Genoa Italy

***Abstract*—This study compares two implementations of a Python Robotics Simulator based on their execution times. The following report details the test performed to evaluate the hypothesis, as well as the experimental setup, results and computations performed.**

I. Introduction

The Python Robotics Simulator is a simple, portable robot simulator [1] developed by Student Robotics, in which a robot is required to locate and collect a series of distributed boxes to a single location. The primary components of the program can essentially be broken down into the robot, the boxes as well as an arena, within the bounds of which the first two elements interact. The original setup of the simulator is simple in nature: the boxes are always spawned at fixed locations, in a circle around the centre of the arena, while the initial location of the robot is always at the top left. The robot is capable of executing simple motions (forward/backward and turning), and is also able to detect the boxes as well as key information about them such as their relative linear and angular displacement, the box ID and the box type. Using this simulator, two variations of the simulation were created by Mark Henry Dsouza and Michal Krepa, students of the University of Genoa, on which the comparison was performed. To avoid unnecessary repetitions, the two implementations shall be referred to as the 'Ma' and 'Mi' programs. "Fig. 1" depicts the original setup of the simulator.

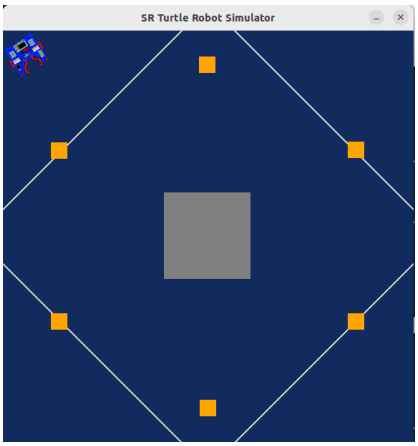

Fig. 1: Original setup of the simulator

II. Hypothesis

A. Null Hypothesis (H_0)

The means of the execution times for the 'Ma' and 'Mi' programs are the same, when the boxes are randomly generated on a ring around the centre. ($u_{Ma} = u_{Mi}$, with u_{Ma} and u_{Mi} being the mean execution times of 'Ma' and 'Mi' respectively.)

B. Alternative Hypothesis (H_a)

The mean of the execution time for the 'Ma' program is less than that of the 'Mi' program, when the boxes are randomly generated on a ring around the centre ($u_{Ma} < u_{Mi}$).

The level of significance is taken to be as 0.05. ($\alpha = 0.05$).

III. Experimental Setup

As mentioned before, boxes are spawned offset from the centre in a ring at fixed angles. For the experiment, the boxes were made to randomly generate in a ring around the centre by altering the *'two-colours_assignment_arena.py'* file, which involved giving random values to the angles. While random generation for the spawn radius was considered, it was ultimately rejected because the two programs were optimised to collect boxes arranged in a ring. "Fig. 2" depicts examples of these randomly generated boxes. Execution times were logged into two text files, *'MarkProgramTime.txt'* and *'MichalProgramTime.txt'*, found in the *'Output_Files'* folder, with the Python scripts amended to log the execution times at the end of each program.

In total, 71 observations were taken into consideration for the analysis. To avoid manually running the tests, a Bash script titled *'runtest.sh'* was created to successively run each test iteration. Any execution that took longer than 150 seconds was considered a failure and logged accordingly. Before the experiment, test runs of both programs were conducted to compare their semantic performances, revealing significant differences in the dispersion of the boxes in their final clusters. The 'Ma' program was extremely conservative, keeping the boxes as close together as possible, while the 'Mi' program was more liberal, allowing for more space between the boxes. To maintain consistency, the placement threshold for the 'Ma' program was increased so that the end dispersion of both programs was comparable.

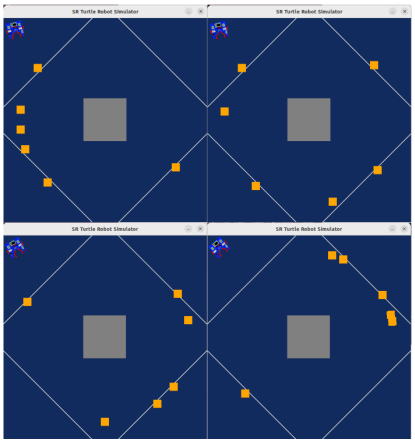

Fig. 2: Examples of the randomly generated boxes

Seventy-five runs of the 'Ma' program were performed, with 4 failures due to execution time hitting the threshold, typically when the robot got stuck in a corner trying to reach a box. Conversely, 128 runs of the 'Mi' program were executed, with 49 failures recorded. The first 71 successful observations were considered for analysis. Most failures of the 'Mi' program were due to the robot being unable to locate a box, thus reaching the timeout value, while some failures were due to the robot getting stuck, similar to the 'Ma' program.

IV. Results

The 71 observations for both the 'Ma' and 'Mi' programs were graphed into histograms, depicted in "Fig. 3".

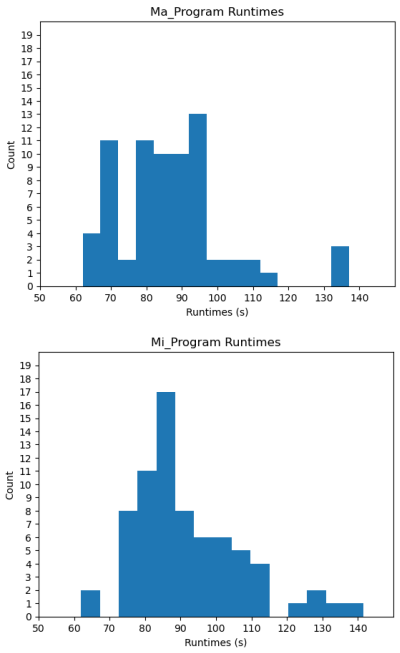

Fig. 3: Histograms of the samples

Before computing the sample means and standard deviations, any outliers present in the samples were extracted. A criterion was adopted to identify outliers, defined by observable breaks in the histogram readings. Based on this criterion, readings above 130 seconds were discarded from the 'Ma' program, while for the 'Mi' program, values below 70 seconds and above 120 seconds were discarded. After extraction, the 'Ma' program had 68 readings, and the 'Mi' program had 65. The 'Ma' program recorded a mean of 86.73 seconds and a standard deviation of 15.63, while the 'Mi' program recorded a mean of 92.96 seconds and a standard deviation of 16.09. The post-experiment analysis of the observations was carried out in Jupyter Notebook. The file titled *'PostExperiment.ipynb'* can be found in the *'Output_Files'* folder. "Fig. 4" depicts the outputs displayed as in Jupyter notebook.

```
MA SAMPLE:
Number of observations=  68
Ma_mean=  86.73705882352942
Ma_std=  15.630208031698965

MI SAMPLE:
Number of observations=  65
Mi_mean=  92.9673846153846
Mi_std=  16.091769924828327
```

Fig. 4: Observed outputs

V. Statistical Analysis

A two sample lower-tailed T-test was performed to evaluate the hypothesis. The test-statistic (t-value) was computed according to the following formula:

$$t = \frac{u_{Ma} - u_{Mi}}{\sqrt{s_p^2(\frac{1}{N_{Ma}} + \frac{1}{N_{Mi}})}} \tag{1}$$

where u_{Ma} and u_{Mi} are the sample means of the 'Ma' and 'Mi' programs respectively, and N_{Ma} and N_{Mi} their sample sizes. The pooled variance, s_p^2, is formulated as follows:

$$s_p^2 = \frac{((N_{Ma} - 1) * s_{Ma}^2) + ((N_{Mi} - 1) * s_{Mi}^2)}{N_{Ma} + N_{Mi} - 2} \tag{2}$$

with s_{Ma} and s_{Mi} being the standard deviations of the observations.

Based on the above formulas, a t-value of -2.264 was obtained.

VI. Conclusion

The critical value for a one-tailed test with a significance level of 0.05 and 100 degrees of freedom is 1.660. As the t-value is less than the critical value, the null hypothesis is rejected. Thus, it can be concluded that the mean execution time of the 'Ma' program is less than that of the 'Mi' program.

References

[1] https://github.com/CarmineD8/python_simulator