University of Puerto Rico
Mayagüez Campus
Mayagüez, Puerto Rico

Department of Electrical and Computer Engineering

Agents and Task Management

by:
Manuel Alejandro Umaña Rodriguez
Dahyna Gabrielle Martínez Pérez
Jan Luis Pérez de Jesús
Christopher Hans Mayens Matías

For: Profesor José Fernando Vega Riveros
Course: ICOM 5015, Section 001D
Date: March 7,2025

Abstract:

How is the performance of a simple reflex agent affected by its environment? Throughout the following report the objective is to establish how the initial environment affects the efficiency of a simple reflex agent. To determine this the vacuum agent problem was implemented in 2 different ways. With them being a 1 dimensional implementation where the agent has only 3 actions available which are left, right and suck in a 2 block environment where its states are clean or dirty. The other implementation was creating a 2 dimensional 5x5 interactable grid where it can be dynamically changed based on user input, where the agent in addition to the actions already described it also had available the up, down, and idle actions all chosen in a random basis within the agent. In addition the second implementation is further divided into 2 subsets with one set having memory of the previous 4 cleaned locations in the environment, and the other having no memory of what locations have been cleaned. The goal for this is having the agent have a random pattern that measures how much of the environment affects the agent when there is no precreated pattern for a given environment. The results for the 1 dimensional experiment determined that the initial condition on where the agent starts has a great impact on the performance of the agent, with the performance of 100% when the agent starts on the dirty block with no other dirty location and an average efficiency of 60% for most other initial conditions. The results for the second experiment do not provide such clear answers. Due to the fact that both approaches randomized the agent's actions, it resulted in the memoryless agent performing better in environments with more open spaces and fewer restrictions. Meanwhile, the agent with memory favored environments with more rigid structure where the movement options were more limited. As the reflex agent lacks full information about its surroundings, the different implementations gave inconclusive results as to which implementation is better. Depending on the environment being evaluated an agent without memory may outperform an agent with memory of its previous position or vice versa as in some cases the agent with memory got stuck in a part of the environment that the agent with no memory got out of.

Table of Contents:

Group Collaboration:

The tasks amongst each team member were divided by each member's experience utilizing python, as follows:

- Christopher Mayens and Manuel Umaña were in charge of exercise 2.14 within the python code and its segments within the report. In addition to the creation of the presentation for the video with the help of Jan Pérez and Dahyna Martínez.

-Jan Pérez and Dahyna Martínez were in charge of the development of the exercise 2.11 within the python code and its segments within the report. In addition to the creation of the presentation for the video with the help of Christopher Mayens and Manuel Umaña.

-All members of the group were included and participated in the video. Video editing was made by Jan Pérez.

I. Introduction:

How effective are agents at solving problems when tested in different environments? As part of the class discussions, two different versions of the vacuum world environment exercises were analyzed. These had different agent variations such as a reflex agent with no memory of the previous states in the environment, as well as a reflex agent that keeps track of its states. A major issue was predicting the efficiency and performance costs of these agents during group discussions. For the first experiment, since the environment was only limited to two locations, the performance and efficiencies were calculated by setting cost points to each state and calculating the amount of moves required to reach a goal state. This was not possible to determine for the second experiment due to the complexity of the environment. Therefore, different tests were made to be able to gain empirical results of said parameters. The experiments used a 2-dimensional vacuum model, displaying performance and efficiency for comparison as

different variations were made to the environment. Some changes made included changing the initial location of the agent, establishing which room was clean or dirty, having the agent be senseless in a random environment, analyzing the effects of no-operation, creating an agent with memory, establishing where a wall was placed and so forth. Throughout this report, the methodology used to measure agent performance will be studied in two specific exercises: Exercise 2.11 and Exercise 2.14. he base implementation for both exercises was available in the Aimacode GitHub repository, with some modifications made for the specific exercises.

II. Related Work:

For the environment of exercise 2.11, it was determined before conducting the simulations to be partially observable, discrete, deterministic, static, single agent, and the agent represents a low complexity reflex agent based on the classic vacuum world environment. Specific measurements were considered to correctly study the behavior of said environment. Now considering the environment of exercise 2.14, it was determined to be partially observable, discrete, non-deterministic, static, single agent, and the agent in one test had no memory [1] while a second implementation had the same agent with a type of memory implementation so it may not repeat locations. Understanding the information obtained is crucial to comprehend basic environments and how AI works internally. The ability of AI to analyze and optimize complex systems is an invaluable tool for managing environmental metrics in the manufacturing industry [2].

The design criteria of the environment for exercise 2.11 was a GUI with 2D, 1 by 2 grid that only had two locations: left and right. Each room had two modifiable statuses: clean (C) or dirty (D). A vacuum agent was randomly placed on one of the two locations. A movement, performance, and efficiency counter were placed on top of the screen. The test was executed until a goal state was reached.

The design criteria of the environment for exercise 2.14 was a GUI with 2D, 7 by 7 grid where all the borders of the grid were classified as walls (W). Each room had two modifiable statuses: clean or dirty. A vacuum agent was placed on location (3,3). A movement, performance, and efficiency counter were placed on top of the screen. For this environment, the user could also add or remove a wall wherever within the grid except the borders of the grid to prevent the agent from accidentally exiting the environment. The limit for a test was 500 moves since there were possibilities of the agent never cleaning the entire room in smaller samples.

For this report, environmental metrics were designed as previously stated. Performance was measured utilizing a flag that increased using multiples of 10 for exercise 2.11 and multiples of 100 for exercise 2.14 whenever the agent made the correct steps to reach the end state and decreased by 1 for both cases when the agent strayed away from it. A similar performance technique is used in current-day AI models to help them learn from patterns and correctly reach a

particular goal [3]. Performance dictates how well the agent performs in the environment. Other than performance, efficiency was measured and studied. Efficiency percentage was defined as the amount of completed tasks to reach the end state divided by the number of steps to get to the end state multiplied by 100. This measure gave an accurate representation of efficiency and helped understand what the agent was doing outside of what visually could be seen from the simulation.

For the vacuum world environment the formulas used were:

$$Performance = (Amount\ of\ dirt\ sucked * 10) - Amount\ of\ steps\ taken$$

$$Efficiency = \frac{Number\ of\ dirt\ sucked}{Number\ of\ steps\ to\ complete\ task} \times 100\%$$

III. Performance Approach and Results:

A. Performance Measurements of Basic Vacuum Cleaner World

The performance of an agent is a crucial step to determine if it is capable of solving a problem in the most optimum way. For this reason it is necessary to test the efficiency/performance of the agent by giving it certain problems to solve within a predetermined environment. For exercise 2.11 the established environment was partially observable, single agent, static, discrete, deterministic, and sequential. These will be explained in more details as the experiment progresses.

For the exercise, a 1 by 2 vacuum-cleaner environment was analyzed. The premade environment consisted of a vacuum cleaner with two locations that needed to be cleaned when dirty. The vacuum was limited to three functions: Left, Right, and Suck. It depended only on the agent's sensors, one to detect its current location and another to determine whether the room was dirty or clean.

The implementation of the simulation base was already available in the provided repository Aimacode, which is the code for the book "Artificial Intelligence: A Modern Approach" [4]. In the python examples we explored the GUI section and obtained vacuum_agent.py which had the base code. The implementation of the environment was modular, meaning that specific variables could be changed easily by altering the code which was documented by a designated team member. Documenting the code also helped define the design limits and allowed them to be extended if necessary.

The execution started with the vacuum being placed randomly on one of the two possible locations (A, B). The user was then allowed to choose which location was clean or dirty with the possibilities being {(C, C), (D, C), (C, D), (D, D)}. When the agent was allowed to act, it checked if the current room was dirty, and if it was, it cleaned the room. If the room was already cleaned it moved to the next room and repeated the same procedure until both rooms were cleaned. The implementation found was nearly complete but lacked a proper way to measure performance. There was a feature in the code that subtracted 1 whenever the vacuum moved and added 10 when it sucked dirt from a box. This was a simple way to measure and understand the performance of the environment, but it lacked a display for the user to observe. Adding a display not only aided in the comprehension of the code, but also served as a way to study the performance utilizing data generated by the code. With this setup, it became clear that every action that did not contribute to reaching the goal state subtracted from the total performance, resulting in a less optimal solution.

To make the performance measurement more explicit, an efficiency percentage display was created, which divided the amount of dirt sucked by the number of steps taken. While a simple 1 by 2 vacuum environment can be solved in exactly 4 steps, the efficiency percentage was added to allow comparison between different starting states of the environment.

The environment could only start in one of the states shown in Figure 1. Performance and efficiency were measured when no more dirt remained to be sucked, as shown in Table 1. Only states 1 through 6 were considered, as states 7 and 8 were the goal states with a performance and efficiency of 100%. Environments where there is no dirt to be sucked. At the end of the experiment, the most optimal starting states, shown in Graphs 1 and 2, were states 3 and 6, which required only one action and had 100% efficiency. The worst states to start with were states 1 and 2, which required the agent to perform two sucking actions and move between locations. However, there is a serious issue regarding memory. The agent can identify when a room is clean and move on to the next room, but if both rooms are cleaned it will remain applying the action Left and Right if the user prompts the agent to continue. If this repetition is kept going the agent's performance would reach negative values thus leading to an inefficient agent.

B. Performance Measurements of an XY Vacuum Cleaner World

For Exercise 2.14, the focus was on testing different types of agents in a more complex vacuum world, where the environment is partially observable, and the agent doesn't have prior knowledge of geography or dirt locations. The agent was studied in a 7x7 grid, with the ability to modify, at any moment, the environment by adding walls or dirt, or removing them. The vacuum functions for this exercise included moving Right, Left, Up, or Down, as well as the ability to Suck and do nothing (No-op). There are two buttons, one that makes the vacuum move in a random direction, and the other that resets both the grid and the vacuum's position to their original state. The display includes a performance, efficiency and move counter.The base code for this implementation, as well as for Exercise 2.11, was found in the provided repository Aimacode. Key additions to the implementation included the ability to make the program run without human interaction until goal has been reached or time out occurs.

In the vacuum environment grid, the tiles with the letter “W” indicate walls, the tiles with the letter “D” indicate dirt, and the empty tiles indicate empty areas. Due to the high variation in the model, certain design choices were made in the creation of different environments, with each environment being run up until 500 moves or the dirt has been cleaned. If the agent got stuck in a specific place within the environment, it was still allowed to continue until it reached the 500-move limit. If no progress was made by the time it reached 500 moves, the test was judged as a failure. This gives the agent enough time to be able to measure its performance without waiting an undefined amount of time and functions as a time out method. Due to the nature of the environment all possible scenarios could not be conducted. This leads to certain environments being created and measured 5 distinct times for each variation. One variation where the agent has no memory and another set where the agent has memory for 4 experiments. For the sake of measurements, an environment with only walls and an environment with no dirt was avoided, as one scenario would result in an inefficient agent that can't perform any actions, and the other in a perfectly optimized agent with no dirt to clean. The created environments for the different tests are shown in Figures 2,3,4 and 5. The biggest contributor in the waste of moves for all scenarios is the addition of the idle action (no-op). In various scenarios and tests, the idle action was one of the main contributors for wasted movements across all experiments, as it was not uncommon for the agent to waste 10 concurrent moves because all those randomly chosen actions were the idle action. During the different experiments the scenario was tested using 2 distinct approaches. The agent was tested in two variations: one with no memory of previous states, and the other with limited memory, storing only the last 4 squares it had cleaned.

Using the scenario depicted in Figure 2 as a basis, the agent with no memory completed the task with varying degrees of success, as shown in Table 2. It was expected

that the agent would be able to complete this scenario without problems as it had no obstacles blocking its progress, as shown in Graph 3 and 4. When adding a state into this scenario there were cases where the agent did not manage to complete the scenario and resulted in a time out, as shown in Table 3. TThis occurred because the agent could not find a path to the remaining dirty blocks that it had not already passed, resulting in blocked paths, as shown in Graph 5 and Graph 6. Overall, there was no clear advantage to having memory of previous states, as both approaches had comparable results in some cases and favorable results in different scenarios. The stateless approach performed better when there were no constraints along its path, while the approach with memory of the previous states fared better in more constrained environments such as the ones shown in Figure 3 and 4. The measurements of these results can be found in Table 4 with Graphs 7 and 8, and in Table 5 with Graphs 9 and 10 for Figure 3. For Figure 4 the results are found in Table 6 with Graphs 11 and 12; and Table 7 with Graphs 13 and 14.One notable setback was in the scenario depicted in Figure 5, where in the tests only the stateless approach managed to complete the objective of cleaning the dirt and it only completed it in 1 attempt; the rest ended in failure, missing from 1 to 3 dirt tiles depending on the attempt, as shown in Table 8 with Graphs 15 and 16. The biggest contributing factor to the timeout failure was the dirt in the corners with only one access point to it. As the agent struggled to find the entrance with the randomized actions or wasted a lot of moves clashing with the walls or because of the idle command. Additionally, when presented with the same environment, the agent with state memory performed worse, as shown in Table 9 with Graphs 17 and 18. The combination of random action choices and the agent's strategy of avoiding already cleaned blocks caused it to get stuck in the corners, as it had no way to return. To be able to return, it would have had to backtrack through already cleaned blocks.

C. Figures and Tables

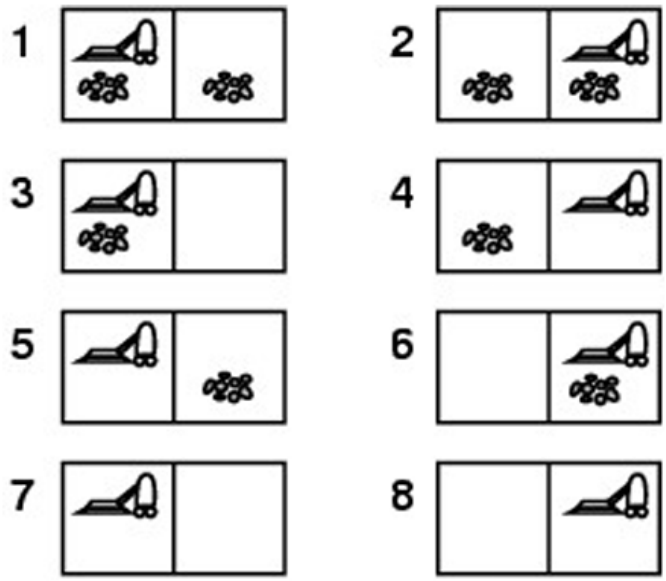
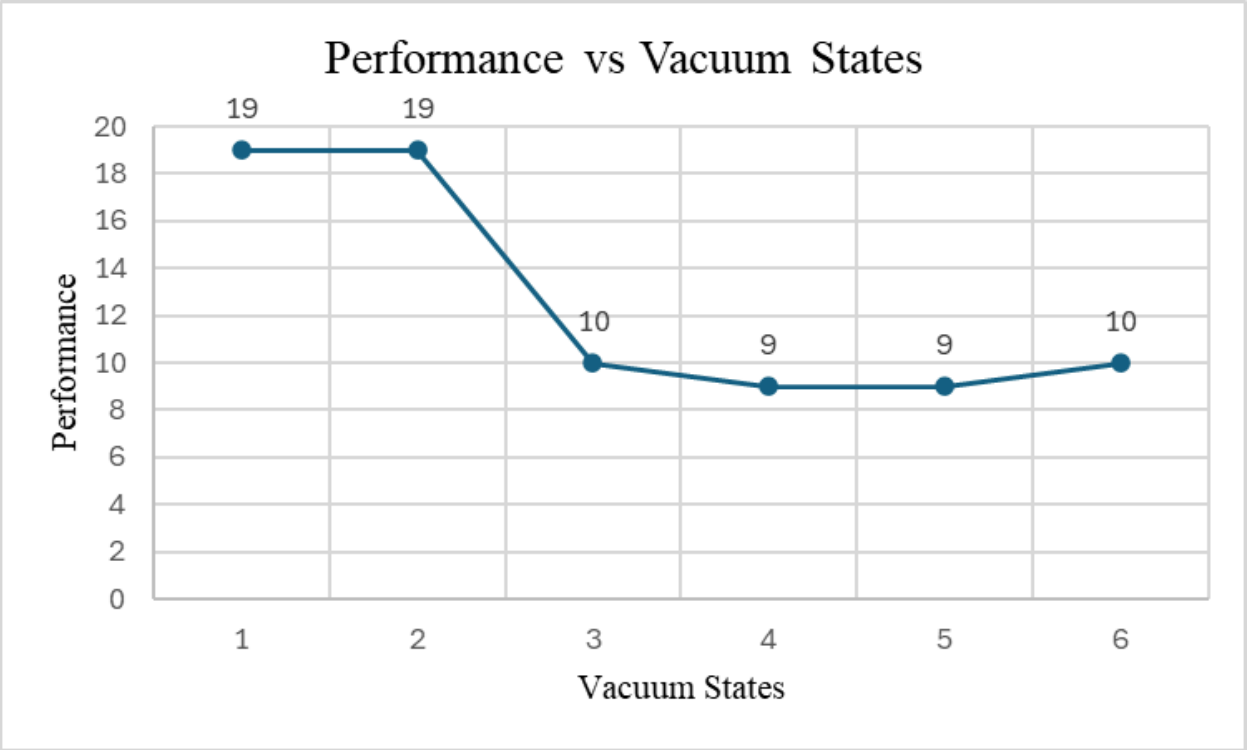


Figure 1. Possible states for vacuum cleaner environment.

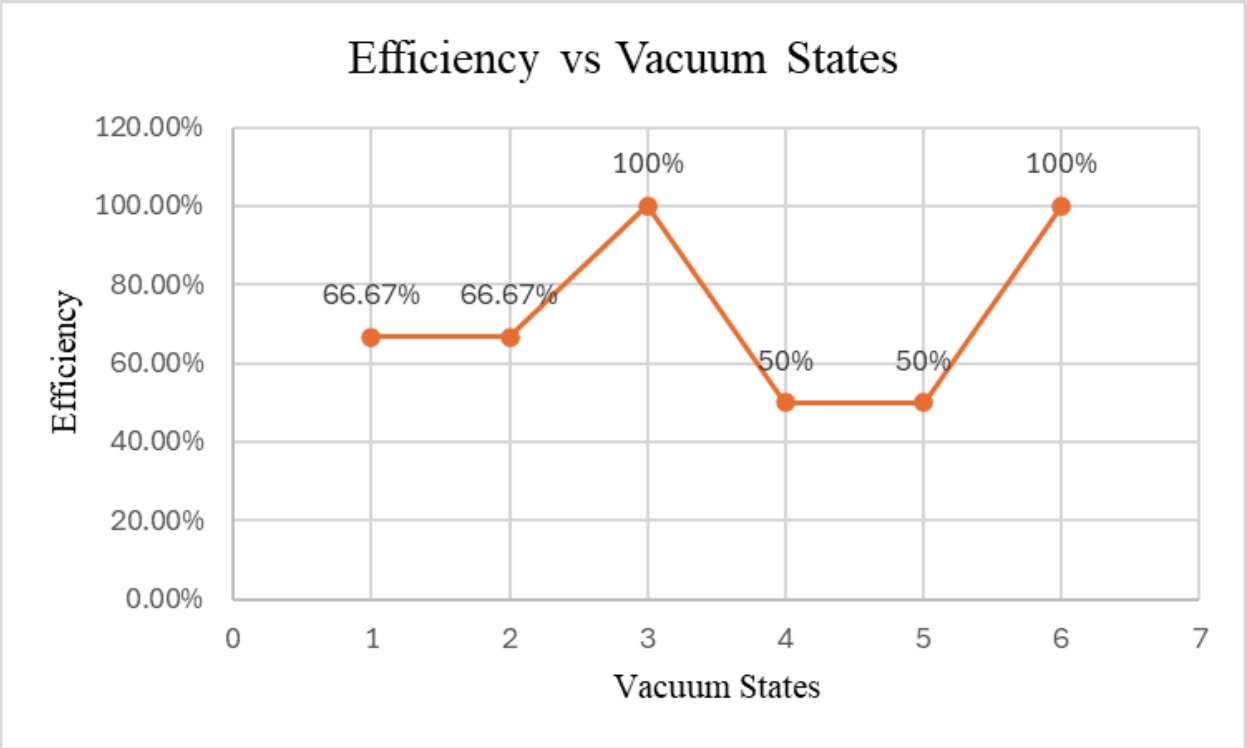
State	Performance	Dirt Cleaned	Steps taken	Efficiency
1	19	2	3	66.67%
2	19	2	3	66.67%
3	10	1	1	100%
4	9	1	2	50%
5	9	1	2	50%
6	10	1	1	100%

Table 1. Measured performance and efficiency of the different states for the vacuum environment.

if the environment is partially observable, the agent needs to take 2 steps just to discover that the other room is clean.



Graph 1. Comparison of performance in different vacuum states.

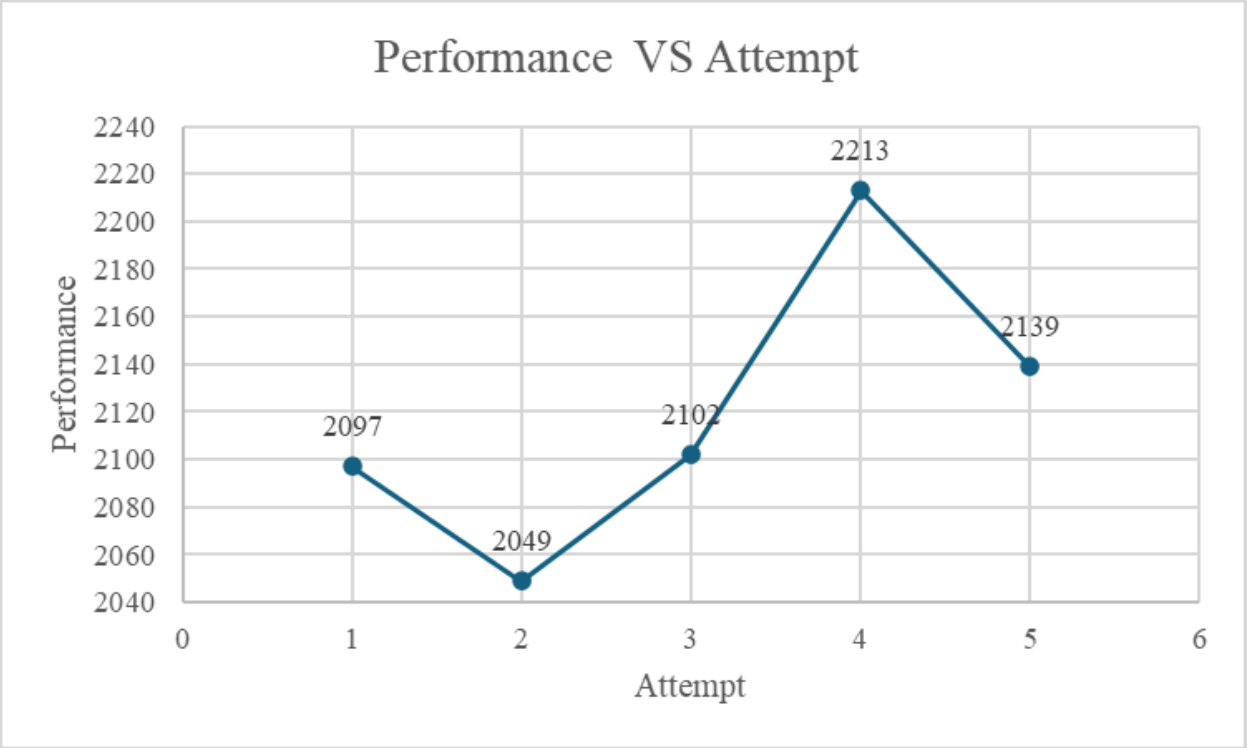


Graph 2. Comparison of efficiency in different vacuum states.

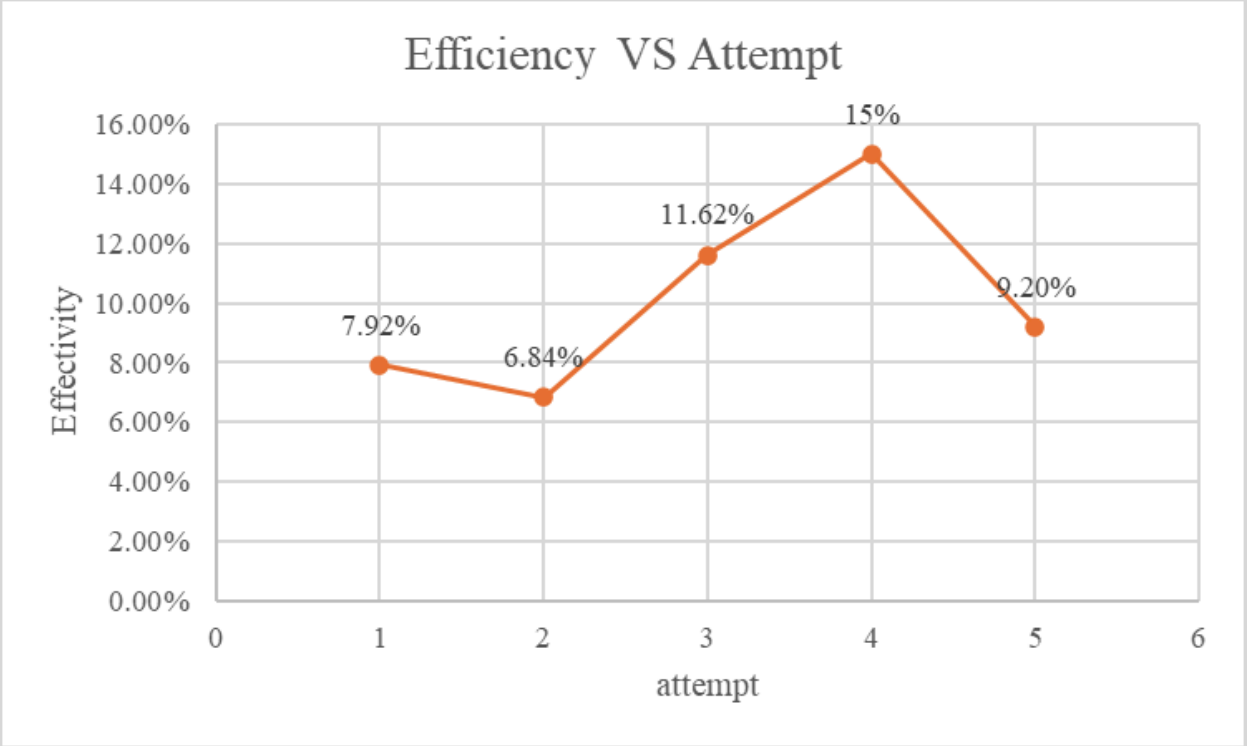
Figure 2: Environment for exercise 2.14 all dirt around the agent.

Test for Figure 2 with no state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	2097	24	302	7.92%	Yes
2	2049	24	351	6.84%	yes
3	2102	24	198	11.62%	yes
4	2213	24	180	15%	yes
5	2139	24	261	9.20%	yes

Table 2: Measured performance and efficiency of different attempts for the vacuum environment with the agent having no record of previous states for scenario in Figure 2.



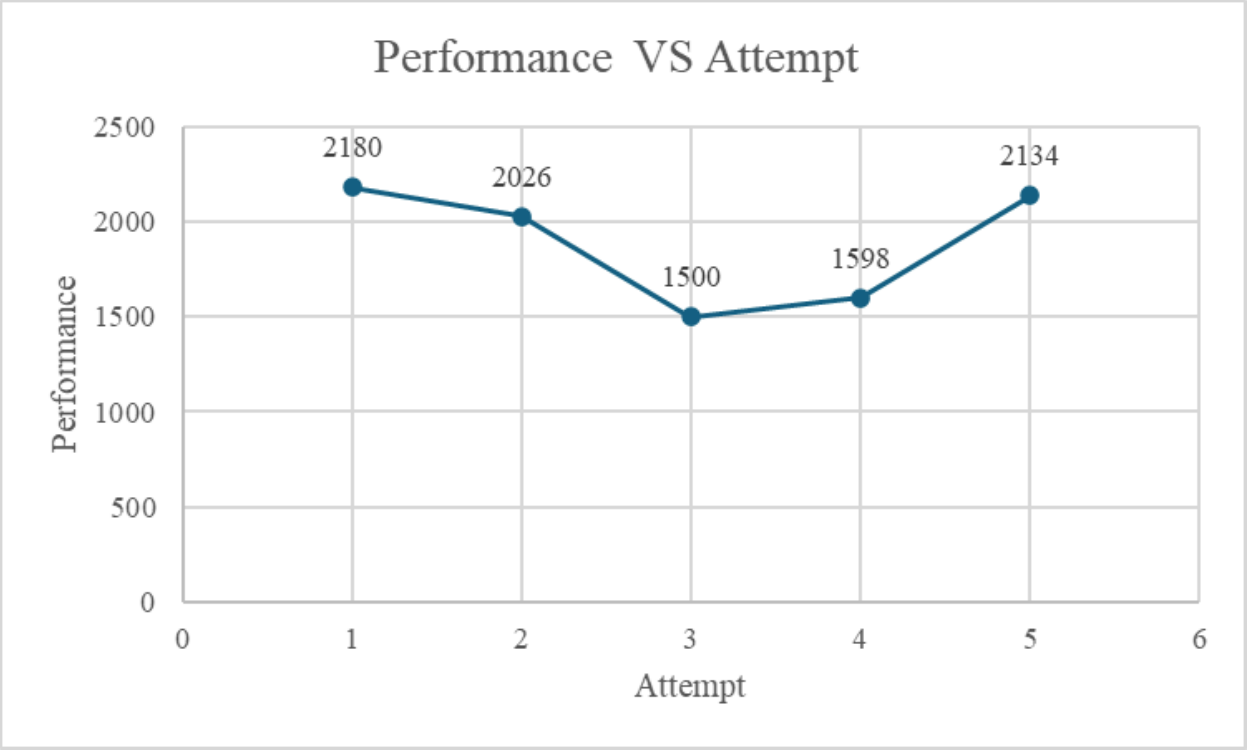
Graph 3. Comparison of performance in different attempts regarding the amount of steps with the agent having no recording of its states for scenario in Figure 2.



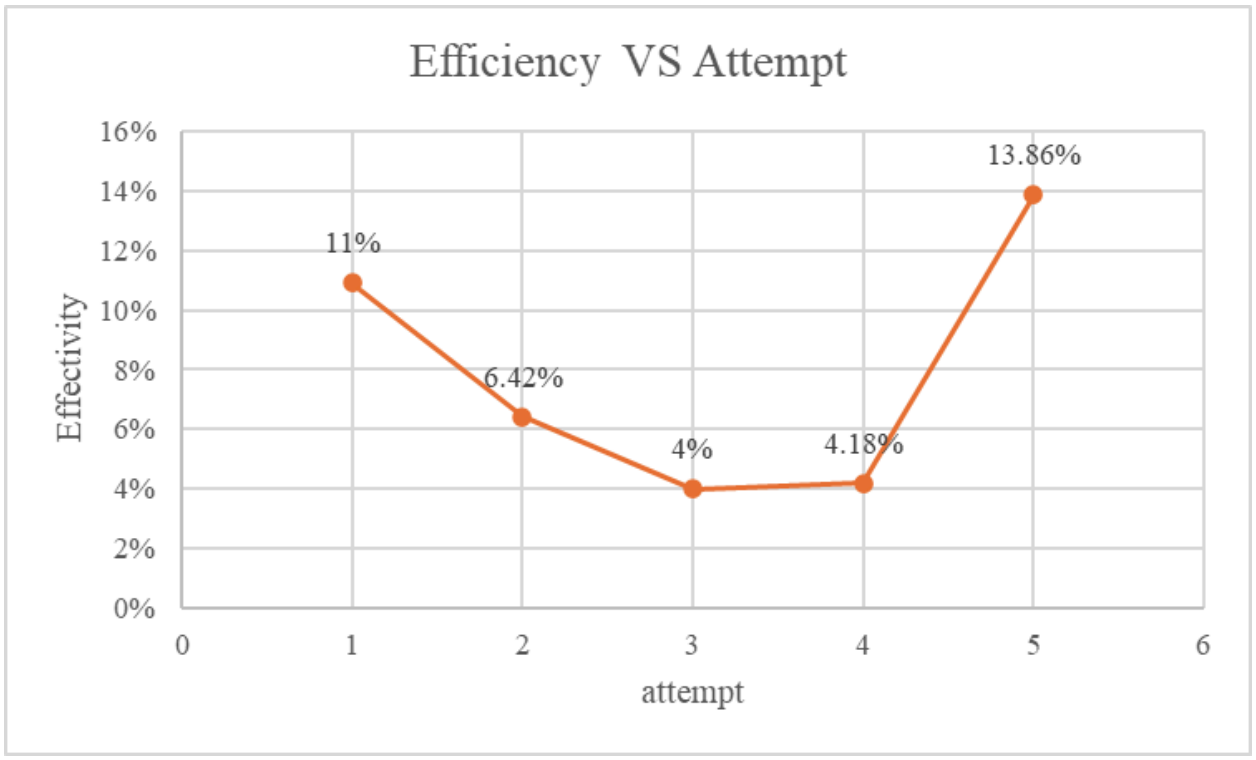
Graph 4. Comparison of efficiency in vacuum steps with no recording of states for scenario in Figure 2.

Test for Figure 2 with state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	2180	24	220	11%	yes
2	2026	24	374	6.42%	yes
3	1500	20	500	4%	no
4	1598	21	500	4.18%	no
5	2134	24	166	13.86%	yes

Table 3: Measured performance and efficiency of different attempts for the vacuum environment with agent having record of previous states for scenario in Figure 2.



Graph 5. Comparison of performance in different attempts regarding the amount of steps with agent having records of its states for scenario in Figure 2.

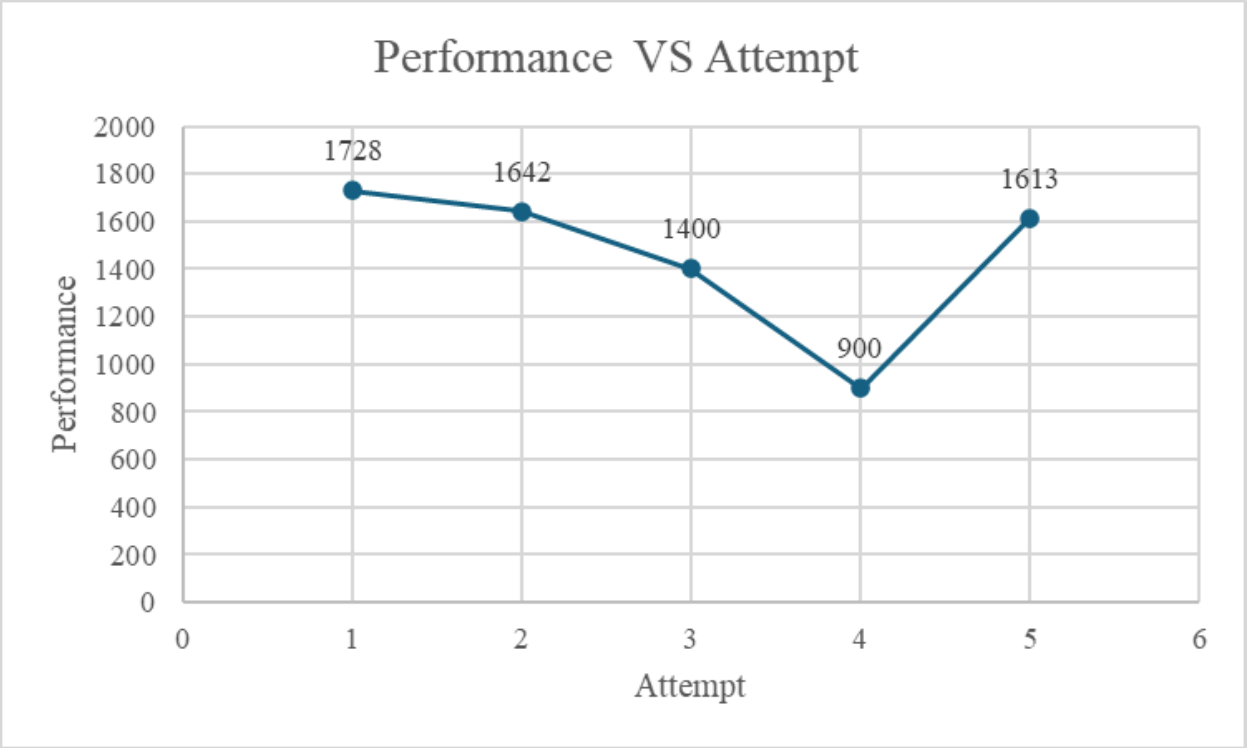


Graph 6. Comparison of efficiency in vacuum steps with recording of states for scenario in Figure 2.

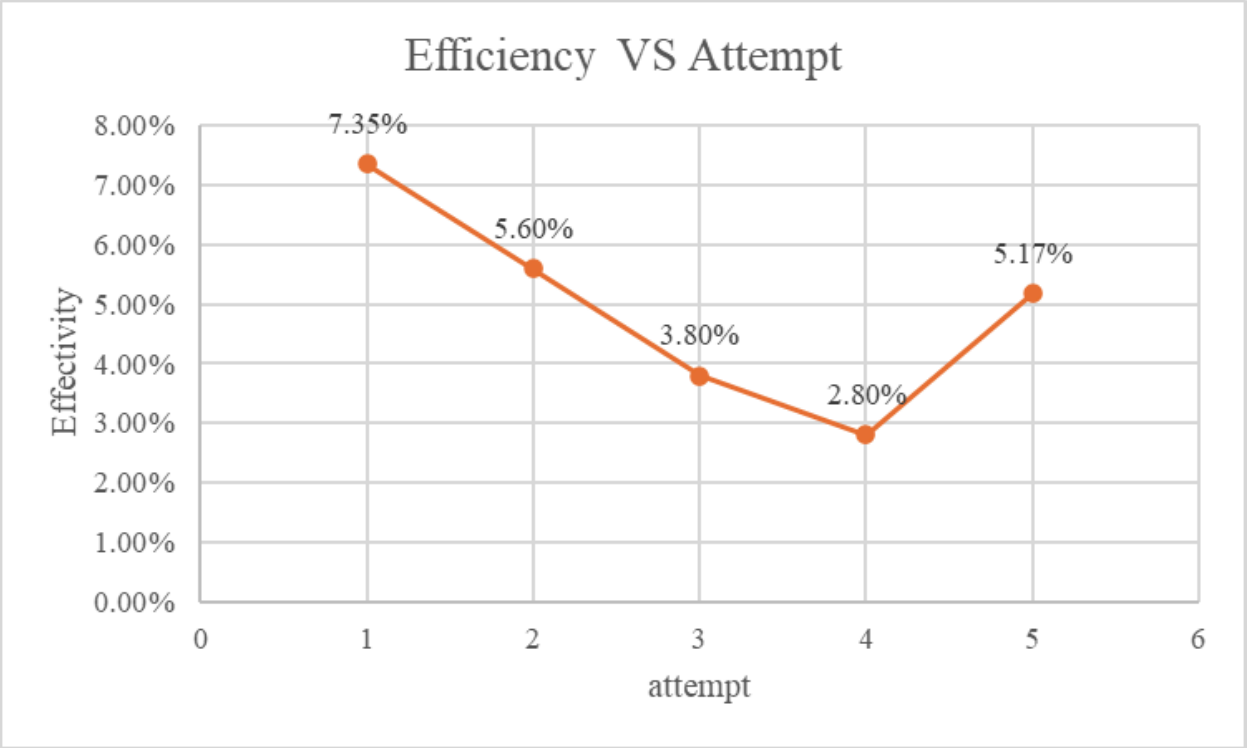
Figure 3: Walls in some areas around the agent.

Test for Figure 3 with no state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	1728	20	272	7.35%	yes
2	1642	20	358	5.60%	yes
3	1400	19	500	3.80%	no
4	900	14	500	2.80%	no
5	1613	20	387	5.17%	yes

Table 4: Measured performance and efficiency of different attempts for the vacuum environment with the agent having no records of previous states for scenario in Figure 3.



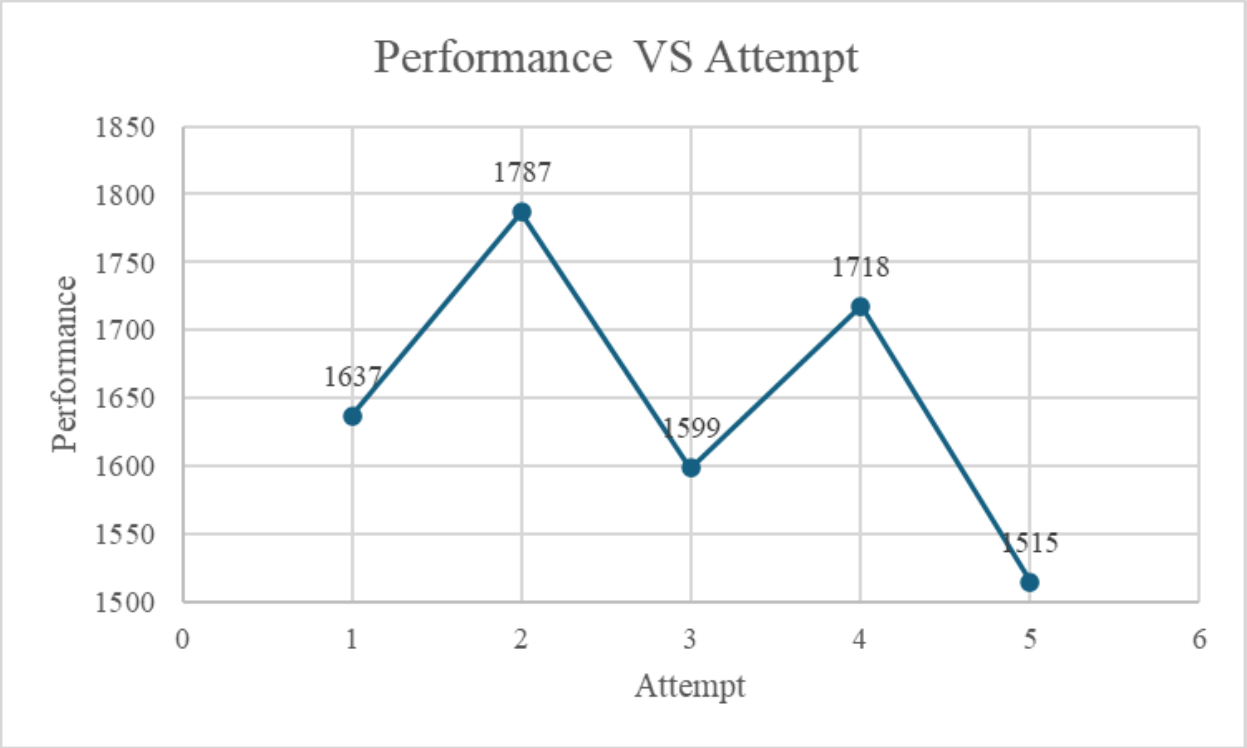
Graph 7. Comparison of performance in different attempts regarding the amount of steps with the agent not having records of its states for scenario in Figure 3.



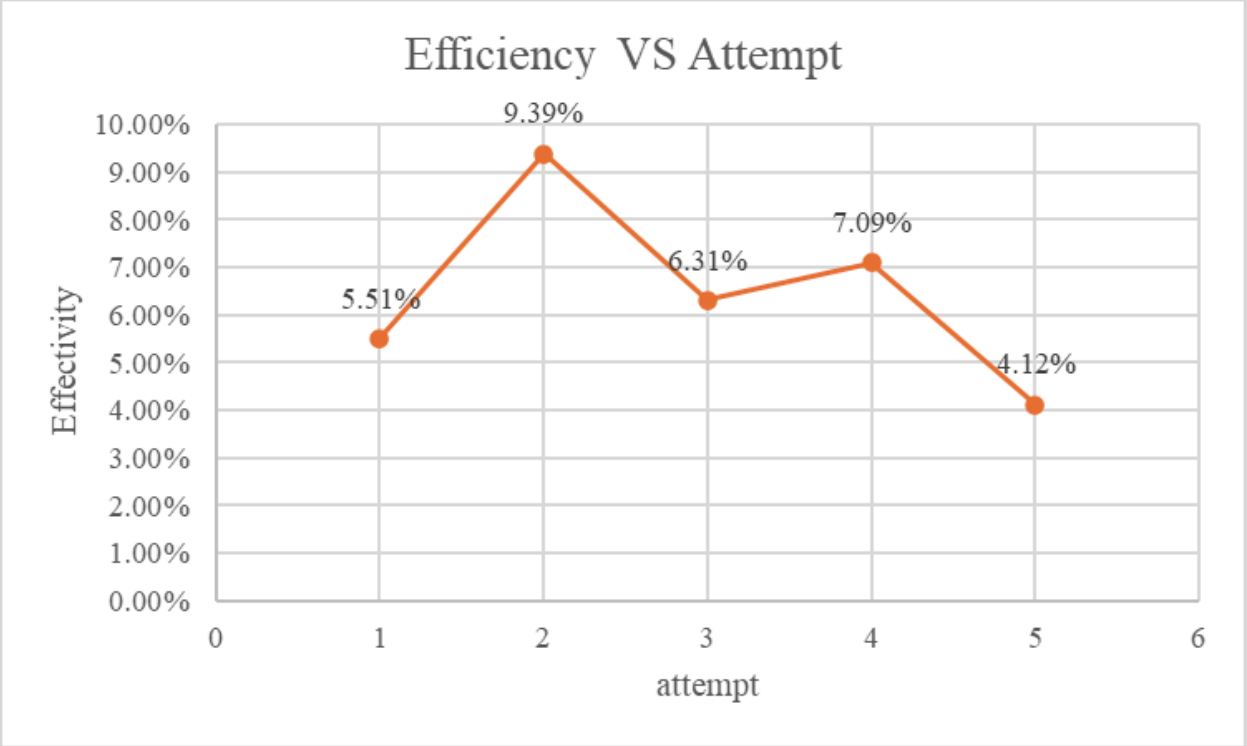
Graph 8. Comparison of efficiency in vacuum steps with no recording of states for scenario in Figure 3.

Test for Figure 3 with state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	1637	20	363	5.51%	yes
2	1787	20	213	9.39%	yes
3	1599	20	301	6.31%	yes
4	1718	20	282	7.09%	yes
5	1515	20	485	4.12%	yes

Table 5: Measured performance and efficiency of different attempts for the vacuum environment with agent having records of previous states for scenario in Figure 3.



Graph 9. Comparison of performance in different attempts regarding the amount of steps with agent having records of its states for scenario in Figure 3.

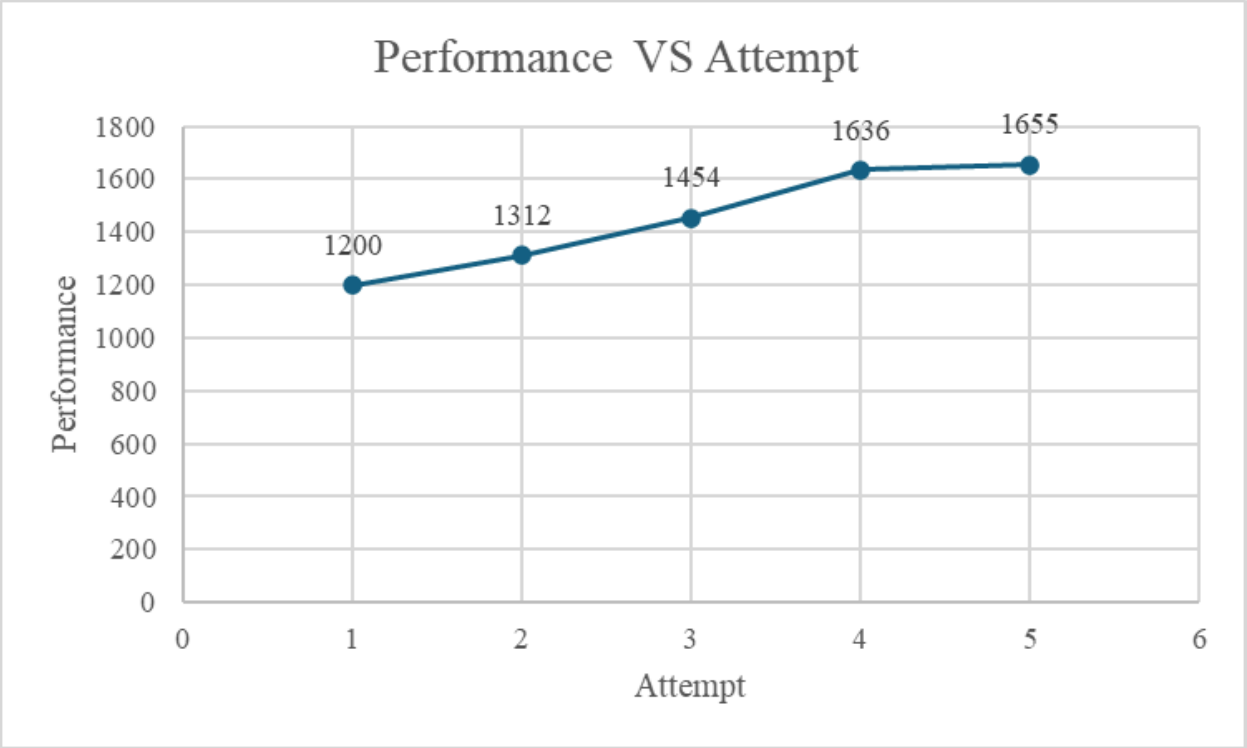


Graph 10. Comparison of efficiency in vacuum steps with recording of states for scenario in Figure 3.

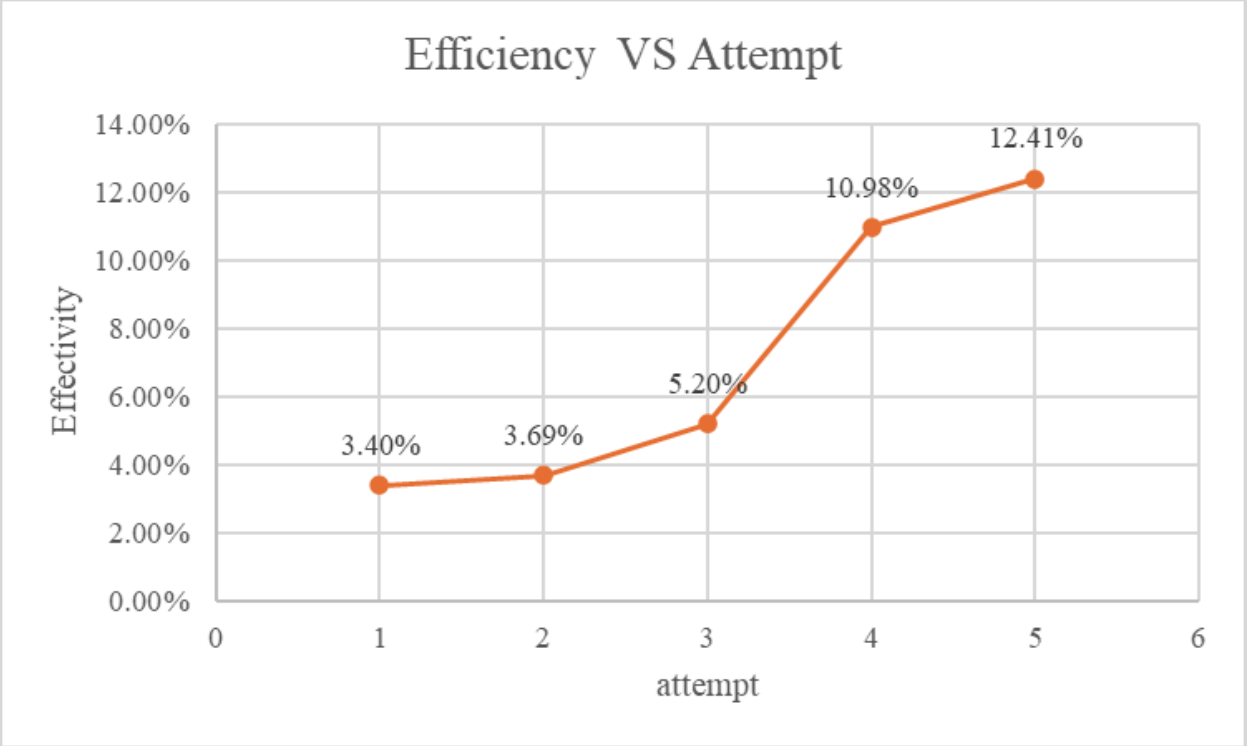
Figure 4: Environment variation test.

Test for Figure 4 with no state					
Attempt	Performance	Dirt cleaned	Steps taken	efficiency	Complete cleaned dirt
1	1200	17	500	3.40%	no
2	1312	18	488	3.69%	yes
3	1454	18	346	5.20%	yes
4	1636	18	164	10.98%	yes
5	1655	18	145	12.41%	yes

Table 6: Measured performance and efficiency of different attempts for the vacuum environment with agent having no records of previous states for scenario in Figure 4.



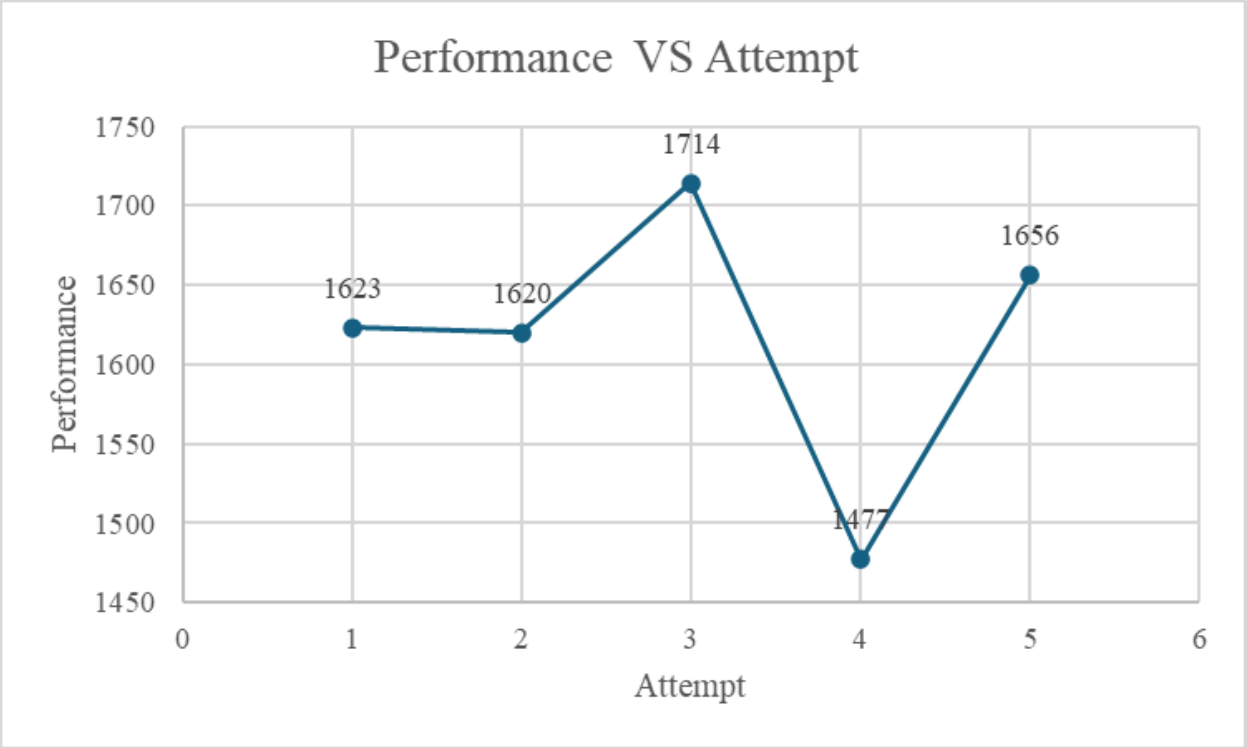
Graph 11 . Comparison of performance in different attempts regarding the amount of steps with the agent not having records of its states for scenario in Figure 4.



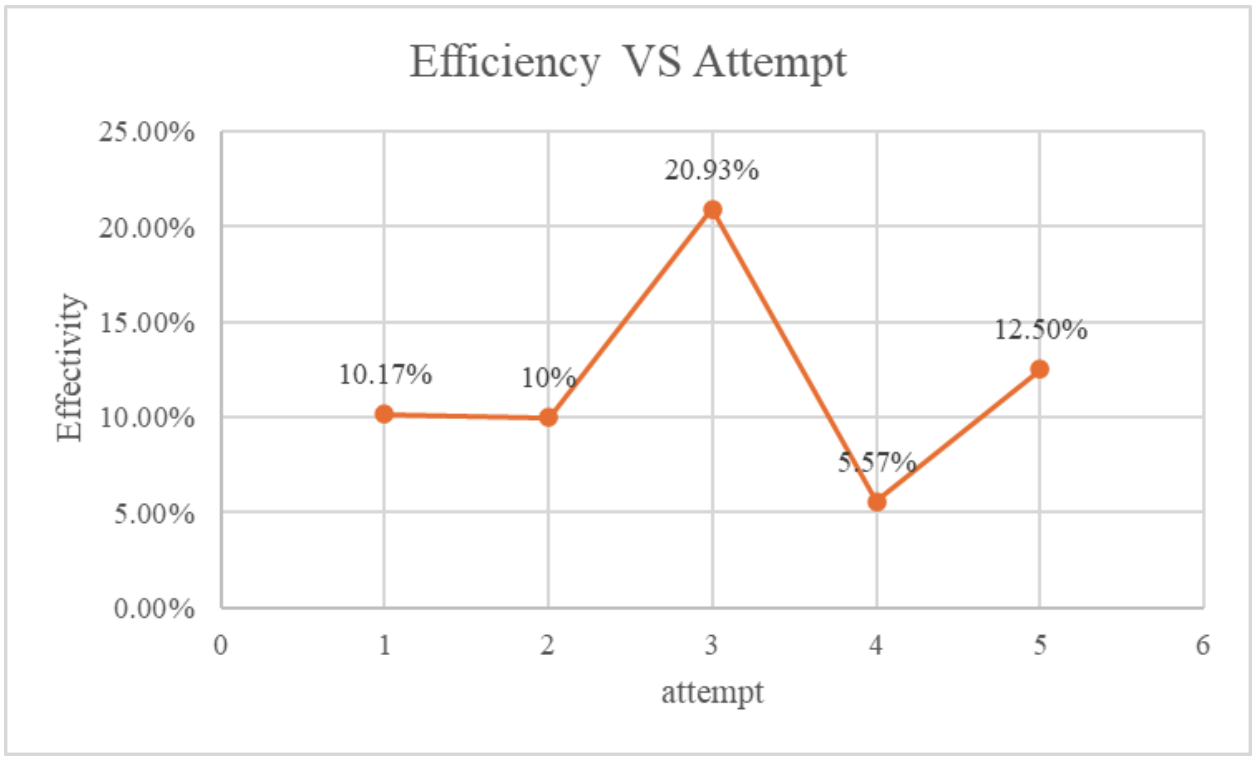
Graph 12. Comparison of efficiency in vacuum steps with no recording of states for scenario in Figure 4.

Test for Figure 4 with state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	1623	18	177	10.17%	yes
2	1620	18	180	10%	yes
3	1714	18	86	20.93%	yes
4	1477	18	323	5.57%	yes
5	1656	18	144	12.50%	yes

Table 7: Measured performance and efficiency of different attempts for the vacuum environment with the agent having records of previous states for scenario in Figure 4.



Graph 13. Comparison of performance in different attempts regarding the amount of steps with agent having records of its states for scenario in Figure 4.

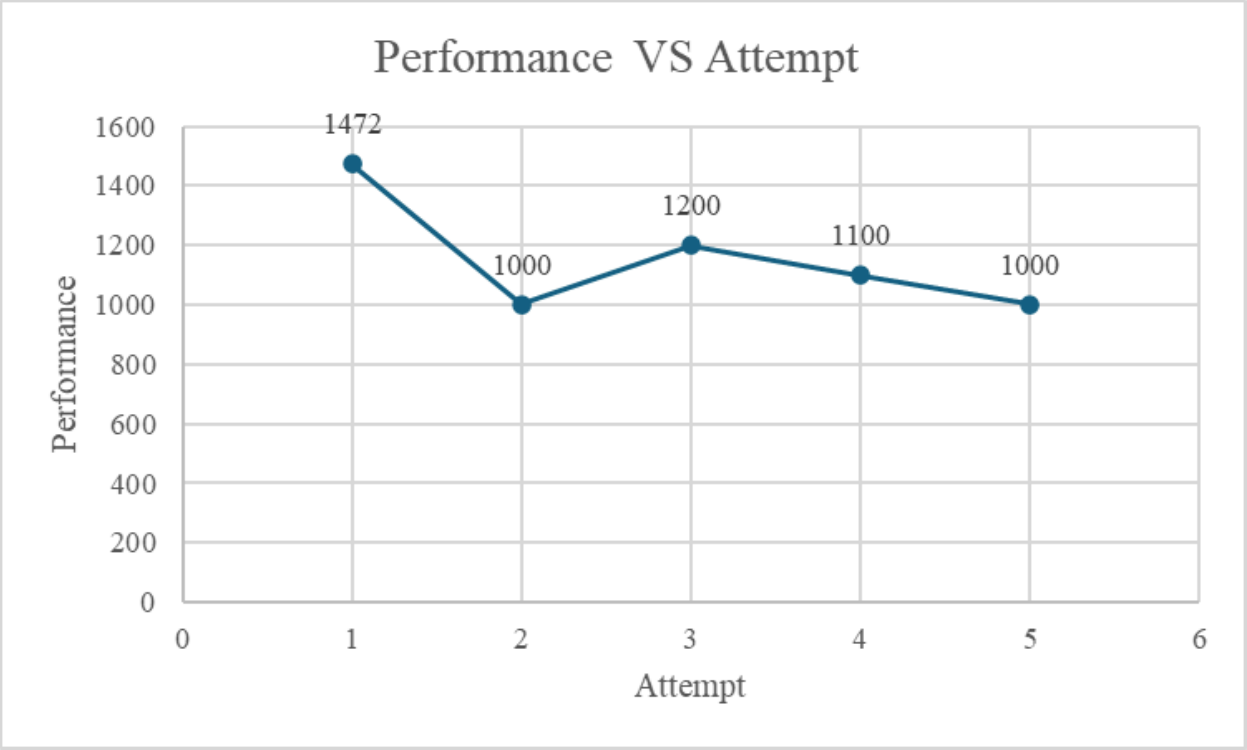


Graph 14. Comparison of efficiency in vacuum steps with recording of states for scenario in Figure 4.

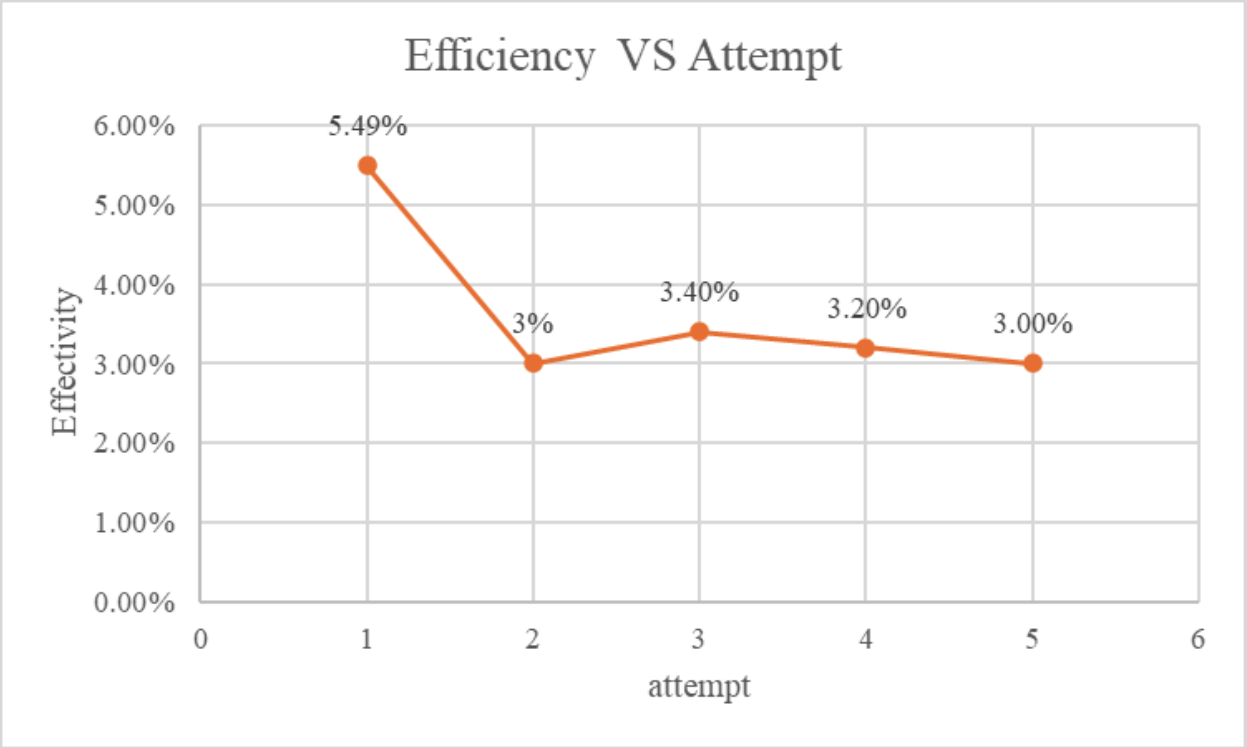
Figure 5: Final environment for testing.

Test for Figure 5 with no state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	1472	18	328	5.49%	yes
2	1000	15	500	3%	no
3	1200	17	500	3.40%	no
4	1100	16	500	3.20%	no
5	1000	15	500	3.00%	no

Table 8: Measured performance and efficiency of different attempts for the vacuum environment with agent having no records of previous states for scenario in Figure 5.



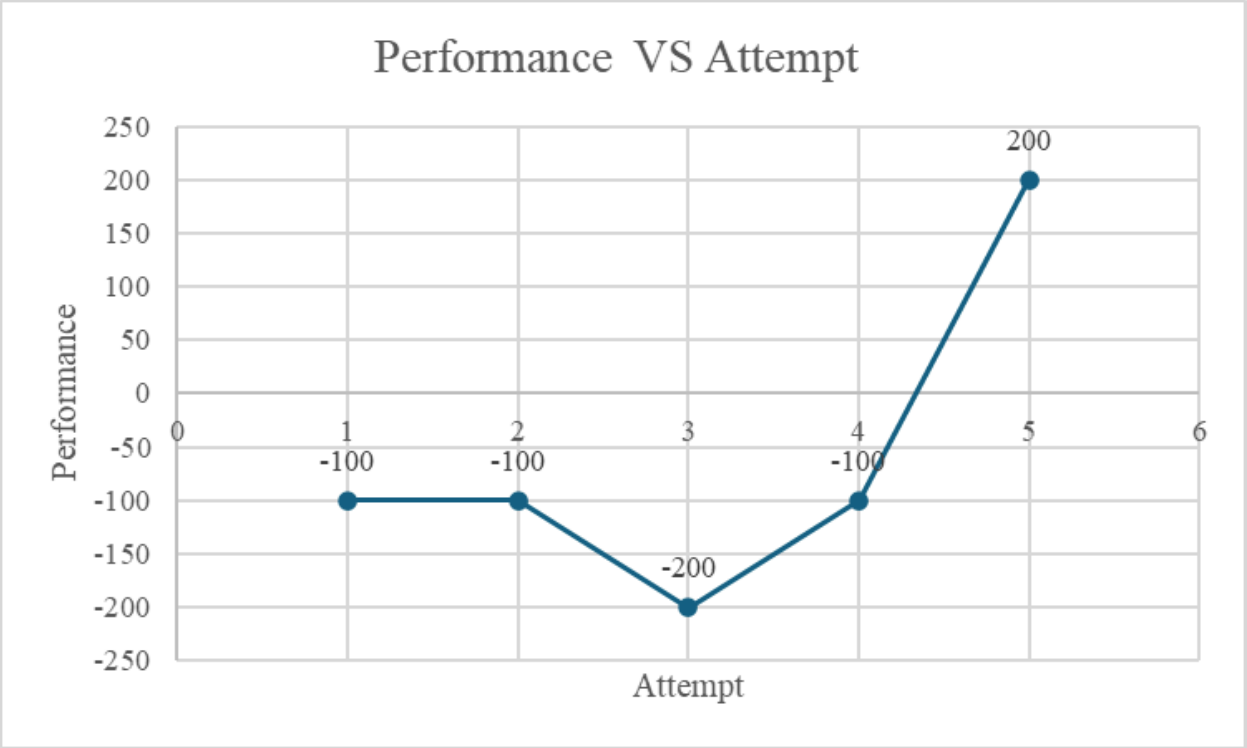
Graph 15 . Comparison of performance in different attempts regarding the amount of steps with the agent not having records of its states for scenario in Figure 5.



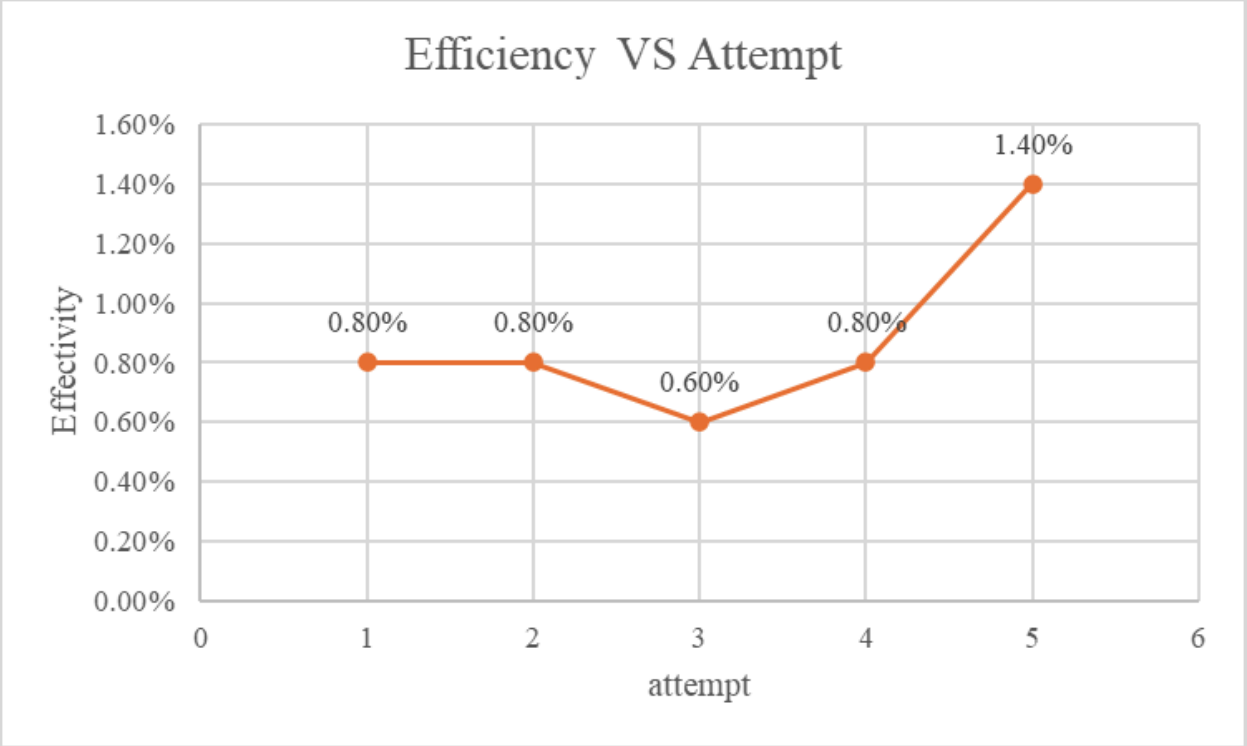
Graph 16. Comparison of efficiency in vacuum steps with no recording of states for scenario in Figure 5.

Test for Figure 5 with state					
Attempt	Performance	Dirt cleaned	Steps taken	Efficiency	Complete cleaned dirt
1	-100	4	500	0.80%	no
2	-100	4	500	0.80%	no
3	-200	3	500	0.60%	no
4	-100	4	500	0.80%	no
5	200	7	500	1.40%	no

Table 9: Measured performance and efficiency of different attempts for the vacuum environment with the agent having records of previous states for scenario in Figure 5.



Graph 17 . Comparison of performance in different attempts regarding the amount of steps with agent having records of its states for scenario in Figure 5.



Graph 18. Comparison of efficiency in vacuum steps with recording of states for scenario in Figure 5.

IV. Conclusion:

For an agent to be considered fully rational it should be able to find a solution to a problem in the most optimal way. This entails establishing a performance metric to define success. For both experiments, this meant having the vacuum agent be able to clean all locations with the least amount of path costs accumulated. For this to have been viable, the agent was heavily dependent on its two sensors: its current location and determining if a room is clean or not. This also required the agent to be able to remember when a room has already been cleaned as to prevent an infinite cycle.

For the first part of the experiment, the vacuum agent was limited to three actions: Left, Right, and Suck. Its performance and efficiency are fully dependent on which state the vacuum was initially. This entailed that even though the problem only required exactly four actions to be solved, fewer actions could be executed if we knew in what state the agent started in. The states that required the agent to conduct one action were the most effective, but this did not entail that the agent knew whether or not both locations were cleaned. The agent only knew if its current room was cleaned, and the program knew if the agent had reached the goal states. This was due to the fact that the environment for the agent was partially observable. For the agent to be sure that both rooms were cleaned, it would have had to traverse each room at least once, leading to a lower performance due to the added verification step. In order to maximize the performance of this environment, the agent would have to keep track of cleaned areas. For example, if the agent is currently in a room and performed a suction or verified that the room was clean, it would keep track and continue moving to the next room. While this implementation would require the agent to still travel both rooms once, it would prevent it from verifying areas that had already been cleaned.

For the second part of the experiment, the agent was expanded to have 3 more actions: Up, Down, and Idle, as well as expanding the size of the grid to a 5x5 square, excluding the grids that were automatically walls. The approach was to create two separate programs, one where the agent has memory of previous locations and one where it does not. In both cases, the agent will perform the suck operation first and then decide on one of the other five movement operations at random. Due to the random operations being performed, the efficiency of both approaches suffered as the agent did not choose the best possible option and usually wasted various cycles bumping with boundaries or being idle. A simple reflex agent would be much more effective, especially one that was designed to go over each space in a simple pattern. As a general assessment of the values obtained from exercise 2.14, the answer of which one is better, having no memory of the state or having memory of the state, is much more complex than it seems. Both states did well in respective scenarios; in fact, they equally had the same amount of tests in which they outperformed the other. By this logic, we can't use the tests to claim one agent is better than the other, as the 50% ratio would contradict this. In the scenarios in which a lot of dirt needed to be sucked and there were non-linear obstacles in the way, the agent without state

memory did better. On the other hand, the scenarios where the obstacles were linear, that is, multiple obstacles were close to each other in proximity, the agent with state did better.

From these two conclusions, we can infer that the reflex agent with state memory will usually do better when it has obstacles that limit its movement. Since it has memory, it can determine where it's been to avoid making the same mistake over and over again. However, it is not good at covering very large areas because the memory restricts its ability to search a wider range of areas. This is due to the fact that a simple reflex agent with a limited memory state can't be perfectly rational if it does not have all the information of its surroundings. It is also not good in small environments where the memory affects its ability to leave an area. An issue that was observed is that the limit of the memory can force the agent with state memory into a NOOP loop. If its memory is three blocks long and goes to a corner, there is no way for it to return and finish sucking the rest of the dirt because every single block around it has been visited. The total random nature of the agent without state allows it to eventually clean all the dirt in a specific area and cover larger amounts in less time.

References:

[1] J. Vega, “Intelligent Agents,” *https://online.upr.edu*, Feb. 6, 2023. [Online]. Available:https://online.upr.edu/pluginfile.php/4927943/mod_resource/content/3/Chapter%202%20Video%202.pdf. [Accessed Mar. 5, 2025].

[2] M. Lynch, “Maximizing Efficiency: How AI Optimizes Environmental Performance,” *praxie.com*, June. 10, 2024. [Online]. Available:https://praxie.com/optimizing-environmental-performance-with-ai/ . [Accessed Mar. 3, 2025].

[3] Colorado State University Global, “How Does AI Actually Work?,” *csuglobal.edu*, 2019. [Online]. Available:How Does Artificial Intelligence Work? | CSU Global . [Accessed Mar. 3, 2025].

[4] Aimacode, “aimpa-python,” *github.com*, Feb. 1, 2016. [Online]. Available:https://github.com/aimacode/aima-python . [Accessed Mar. 3, 2025].

Code Repository:

https://github.com/Dahyna-Martinez/Agents-and-Task-Management