Final Project: Roomba Coverage

Rita Kurban

Minerva Schools at KGI

Abstract

The purpose of this paper is to evaluate the effects of different strategies taken by a robotic vacuum cleaner in a square furnished room. I created a computer simulation to model this process, came up with four strategies and evaluated their performance using multiple metrics, analyzed the effectiveness of each strategy, and visualized the results. Since vacuum cleaners should be well-prepared for different environments, I used Monte Carlo methods to introduce some randomness and make this simulation more realistic.

Simulation Description

Roomba is a series of autonomous robotic vacuum cleaners sold by iRobot. They were first introduced in 2002 and rapidly gained popularity. Since then, multiple analogues have appeared on the market. Robotic vacuum cleaner's success is dependent upon cleaning the floor efficiently, its ability to avoid obstacles and its speed. Therefore, it is crucial to choose an effective strategy that will take all these criteria into account. The most effective way to try different strategies is to create a computer simulation since it doesn't require many financial resources and can be executed millions of times in a matter of minutes.

Models are approximations of the reality, that's why it's important to set priorities and understand what information people want to get from them. In our case, it is particularly important to measure the effectiveness of different strategies in various environments. Therefore, objective characteristics and metrics that compare strategies against each other are much more important than, let's say, esthetics which might be crucial for advertisement.

Assumptions

The model is based on several assumptions which I will describe in this section.

First of all, a room is represented as a square 2D matrix with the user-specified length of the side *self.size.* Since vacuum cleaners often operate inside houses, I assumed fixed boundary conditions where the area of the room is restricted by the walls. The rooms are usually not empty, that's why I randomly put obstacles there with a probability *probobst* which has a default of 0.1. It can be adjusted by the user based on how much furniture the room has. A random distribution of obstacles introduces the first level of randomness this model has. It gives an opportunity to use the same strategy in different environments. For the simplicity purposes, I assume that the obstacles are not moving (no people or pets).

The second level of randomness is the initial location of the vacuum cleaner. Some locations might be more beneficial than others in certain environments. For this reason, I initialize the vacuum cleaner randomly to minimize systemic bias its position can be associated with. I assume that the vacuum cleaner can move only in four directions: up, down, left, and right and that the floor is homogeneous: the vacuum cleaner should visit each tile only once to make it clean. This assumption is oversimplifying the reality because oftentimes there might be some dirty spots on the floor or the surface might be heterogeneous (e.g., carpet/wood/ceramics) which can influence the effectiveness of a certain strategy. However, it doesn't alter the relative performance of strategies. Therefore, it is reasonable to make this assumption to keep the simulation simple.

Regarding execution, Roomba makes only one step at a time. I also assume that it checks for obstacles before moving forward which makes it impossible for it to bump into walls or furniture. In real life, vacuum cleaners often crash into walls and other obstacles. In the

simulation, I will look at the idealistic situation where such breakdowns never happen, and the battery is infinitely charged.

Parameters and Strategies

For simulation to be flexible and fit multiple contexts it should have user-specified parameters. They are listed in the table below.

size (int)	The length of the square room. Can be used to adjust the size of the room.
probobst (float)	The probability of an obstacle in the room. Gives an opportunity to model any distribution of obstacles from a completely empty space to a maze-like rooms.
strategy (str)	"random", "clever", “combined” or "mixed" which correspond to a specific strategy Roomba follows. A more precise description is given below.
cleaned_once (boolean)	If True, the simulation stops after each tile has been cleaned at least once. The simulation can be adjusted to increase the number of times each tile should be cleaned.

Table 1. The parameters of the model

Random

The first strategy I came up with is *random.* First, the vacuum cleaner checks for obstacles and identifies possible directions it can move to during the next step. After that, it randomly chooses one of the available paths and makes one step forward. I quickly realized that this strategy is not effective at all since the vacuum cleaner doesn’t take into account how many times each tile has been cleaned. To account for that, I devised a new strategy *clever*.

Clever

This strategy is a significant improvement over the last one since it doesn't randomly choose one of the possible directions but looks at the number of times each tile was cleaned. It might seem not realistic at first, but some real-life vacuum cleaners have spot detection and can move in the direction of the dirtiest spot. As a result of implementing this strategy, the performance has increased significantly.

Mixed

This strategy is a slight modification of the previous one. In *clever,* it sometimes happens that several tiles in multiple directions are equally dirty. In this case, it would choose a path based on their order in the list with *up* and *left* being the first two. As a result, the distribution of the number of times each tile was cleaned was slightly skewed towards the tiles in this region. To account for that, I decided to mix the first two strategies (hence the name) and shuffle the list with directions so that in case we have multiple paths open, one of them will be chosen randomly, like in the first strategy. I was surprised that the performance of this strategy wasn't much higher. It can be partially explained by the fact that in most of the cases there is only one possible direction. I'll discuss it in the analysis below.

Combined

This approach can combine any of the two strategies described above into one. The ratio and the strategies used can be easily modified in the update function. The purpose of this strategy is to understand whether introducing more randomness into deterministic strategies can increase their performance.

Metrics

Gini Coefficient

Metrics are essential for making objective comparisons. Otherwise, it's difficult to understand whether the strategy is effective or not. One of the primary metrics that I used is Gini coefficient. Gini coefficient is a measure of statistical dispersion often used to represent the wealth distribution of a nation's residents. It is one of the most commonly used measurements of inequality which is perfect for our simulation since we aim to clean the entire room as fast as possible and have every tile cleaned just once. In case the coefficient is close to 1, it indicates that a tiny percentage of tiles have been cleaned many-many times, while some tiles were not cleaned at all. In contrast, if the coefficient is close to 0 (the number we aim for), all tiles were cleaned the same amount of times. By calculating the Gini coefficient at the end of the simulation, we can describe the entire distribution using a single number.

Number of Steps to Clean the Floor

This metric is vital for understanding how fast the vacuum cleaner cleans the entire floor given the strategy. It is closely correlated with the previous metric since the higher the inequality identified by the Gini coefficient, the longer it takes to clean each tile a specific number of times.

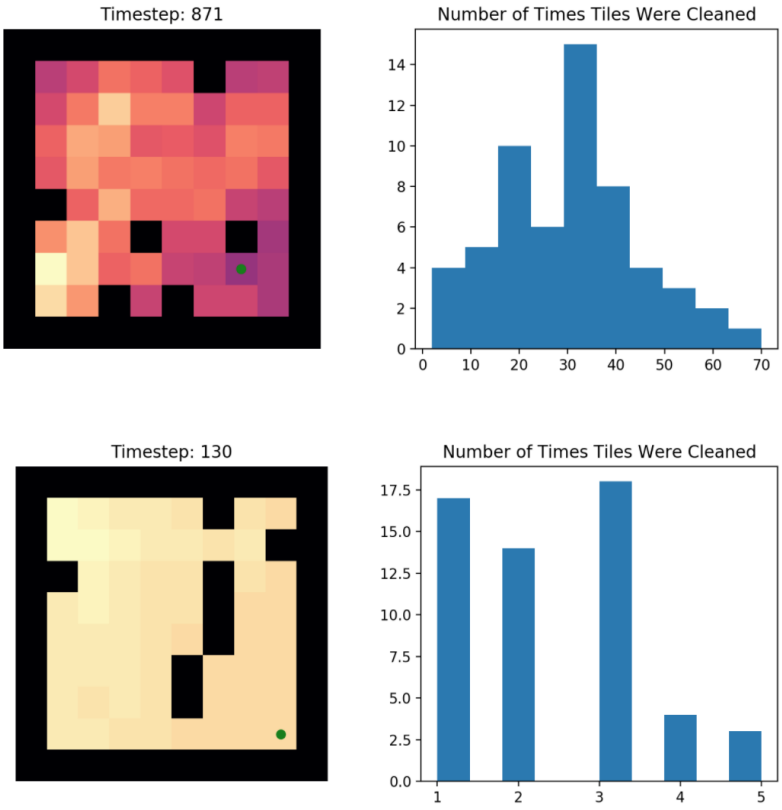
Min, Max, Mean

These three metrics are very useful for visual representation since they illustrate the discrepancy between the smallest and the biggest number of times a part of floor was cleaned and demonstrates a general trend this distribution follows. We aim for the low discrepancy between the maximum and the minimum number and for the mean of around 1.

Analysis

Final Distribution

One of the features of this simulation is that it can produce animations so that users can see the dynamic change in the system. On the plots below, you can see the final distribution of times required to clean the floor entirely. On the left, the room itself is illustrated where colors correspond to specific numbers. On the right, you can see the actual distribution plotted in the form of the histogram. To make the paper concise, I only analyze two main strategies: clever and random.

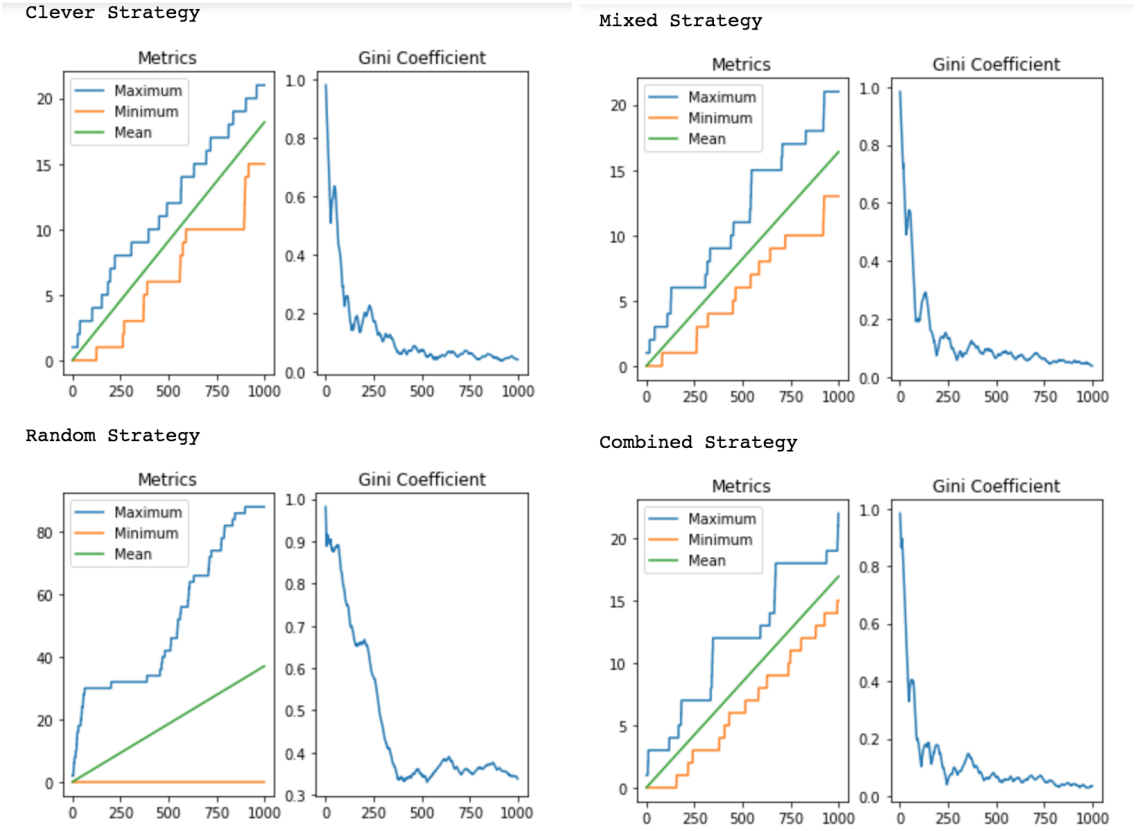


Pic. 1. The top two images correspond to the random strategy, the bottom two to the clever one. The plots on the left illustrate the cleaned room. The histograms on the right show the distribution of times the tiles were cleaned for each of the cases.

As you can see from the plots, it took almost 900 steps for the random strategy to clean each tile only once. Compared to that, clever strategy achieved the same result in just 130 steps. From the histograms, you can see that tiles were cleaned up to 70 times in the first case and only up to 5 in the second. Therefore, the vacuum cleaner requires much less time and energy (charge) if it uses a more efficient clever strategy. Let's now look at some other metrics.

Gini Coefficient and Min/Max/Mean

As I described in the section above, we aim for low Gini coefficients and a low discrepancy between the max and min number of cleanings:



Pic. 2. Metrics for 4 different strategies.

The plots illustrate that clever strategy significantly outperformed random strategy against all metrics. What is interesting is that combined strategy, which uses random strategy half of the times, performs not in the middle of these two strategies, but has an efficiency close to the one of the clever strategy. It means that even a bit of guidance makes random strategy much more effective. Mixed strategy, which introduces more randomness into clever strategy, has the highest minimum numbers and one of the lowest Gini coefficients which makes it a little bit more successful than the original strategy. However, as I've mentioned above, the difference is not significant.

Monte Carlo

The model and simulation incorporate randomness due to the random distribution of obstacles, the initial position of the vacuum cleaner, and the randomness incorporated into the strategies. It makes the model not deterministic and suitable for multiple contexts. However, randomness is also associated with uncertainty. To make sure that the results above were not obtained by chance, I ran the simulation using different strategies 1000 times and calculated some of the statistics. One thing to check is the number of steps since it is an obvious measure of the strategy effectiveness. It is also correlated with the distribution of cleaning times, the discrepancy between the high and low numbers, and the inequality (Gini index). Therefore, by calculating the uncertainty in this number, we can estimate the uncertainty present in the entire model.

First, 95% confidence intervals. The uncertainty is much higher for the random strategy since the entire process is entirely luck-based:

```
Confidence Interval for the Random Strategy: (679.2492801456895, 808.4241892420657)
Confidence Interval for the Clever Strategy: (123.7510514459396, 138.43842223827096)
```

In contrast, clever strategy has a small range of values which shows that the entire process is not entirely random and, therefore, the results are more certain. We can be 100% confident that clever strategy is more effective than the other one given these confidence intervals since their distributions don't even overlap. However, I didn't manage to identify a statistically meaningful difference in the performance of clever and mixed strategies which have very similar confidence intervals of similar width despite the fact that the second strategy has more randomness to it:

```
Confidence Interval for the Mixed Strategy: (127.72740197208633, 143.44785575987243)
```

I also calculated some other statistics of this three strategies with the help of the *scipy.stats* module:

```
Random
DescribeResult(nobs=98, minmax=(322, 1976), mean=743.8367346938776, variance=103782.11739953714, skewness=1.225839281
0940264, kurtosis=1.702406254411719)
Mixed
DescribeResult(nobs=97, minmax=(63, 235), mean=135.58762886597938, variance=1520.9948453608247, skewness=0.0406894632
6037552, kurtosis=-0.6535942843293729)
Clever
DescribeResult(nobs=95, minmax=(58, 203), mean=131.09473684210528, variance=1299.5760358342666, skewness=0.0025475385
074511126, kurtosis=-0.8214631087332207)
```

If we look at the variance it becomes obvious how much higher it is for random strategy and how much more effective other two strategies are based on their means.

Recommendation

For people who want to program robotic vacuum cleaners, it is necessary to incorporate some randomness so that the machine can quickly adjust to changing circumstances and can deal with obstacles at different places. Another important consideration is dirt indicators. They will

significantly increase the performance of the vacuum cleaner compared to competitors that move in random directions without taking into account any additional information. In this specific model, clever strategy won against all metrics compared to the random one. This model can also be used to simulate a specific situation by choosing an appropriate room size and placing the pieces of furniture at their respective places as well extended to contain more strategies.

Limitations and Further Improvements

Even though this model is a nice way to implement and evaluate multiple strategies as described above, it has its limitations and could be further extended. First of all, the vacuum cleaner might have detectors that could make it possible for the machine to move more than one step at a time and track the obstacles from a distance. This would make the simulation more realistic because in real life we rarely see a floor as a discrete integer grid, rather as continuous space where the agent operates. Secondly, it might be possible to implement moving objects, such as pets and people and simulate how the vacuum cleaner reacts to a sudden intervention. Thirdly, it's necessary to account for scenarios when the vacuum cleaner bumps into walls and devise some ways it can deal with such problems. The list of extensions can be further expanded based on the information users want to extract for the simulation. However, if our goal is to compare different strategies, it might make more sense to create a simple but clear model which easily differentiates between several strategies and compares them against objective metrics.