A performance survey of coverage algorithms for simple robotic vacuum cleaners

ROBIN GUNNING

Bachelor's degree in Computer Science
Date: May 14, 2018
Supervisor: Jana Tumova
Examiner: Örjan Ekeberg
Swedish title: En prestandaundersökning av täckningsalgoritmer för enkla robotdammsugare
School of Computer Science

Abstract

English abstract goes here.

Sammanfattning

Abstrakt på svenska.

Contents

Chapter 1

Introduction

Most cheap automatic robot vacuum cleaners (from here on called robot cleaner) today combines three or four different path planning algorithms to get the expected coverage as fast as possible. This has sparked the interest to develop efficient algorithms to reduce cleaning time and be as efficient as possible.

There are several types of robot cleaners of available to the public today, however the price range of these robot cleaners varies greatly and there is no set standard of what sensors a robot cleaner should have. This makes almost every robot cleaner unique in the way of getting a good coverage of the room it is supposed to clean. Furthermore different robot cleaners and companies has their own algorithms developed to maximize coverage and minimizing the dust on the floor.

Usually the price of the robot cleaner seems to correlate with the number of sensors and how advanced the sensors are [6]. Cheaper models often only have a single sensor which is a frontal bumper (Figure. 1), a micro switch which gets pressed when the robot cleaner bumps in to a wall or another object.

Figure 1: Skantic Robot Cleaner 10 [7]

More expensive models often utilize optical sensors to measure how many times the wheels have rotated and some models even use lasers and radar to build virtual maps of the rooms the robot cleaner operate in. Rooms come in all different sizes and shapes and different algorithms are likely to have different strengths depending on the type of room. The standard algorithms for cheap robot cleaners are Boustrophedon, random walk, spiral and wall follow [4].

Boustrophedon travels forward until hitting a wall, when hitting the wall the robot reverses for a while then turns 90 degrees clockwise or counter clockwise then travels for a short period and turns another 90 degrees in the same direction as chosen when hitting the wall. That makes the robot cleaner do an 180 degrees turn while not cleaning the same space twice. When reaching the opposite wall the robot cleaner performs the same set of instructions as before but in the opposite direction. This makes the robot cleaner zig-zag between two walls while traversing the room.

Random walk turns a random amount of degrees and travels in a straight path until it hits a wall.

Spiral is a counter clockwise spiral from in to out and wall follow when hitting a wall travels in a small half circle until it hit the wall again and continues doing this around the room or until the robot cleaner switches algorithm [4].

Chapter 2

Background

2.1 Previous research

Most previous research seem to use cellular decomposition which divides the room into several cells and performs the algorithm in each cell. Each cell is made up of a part of the room, making a new cell at the critical point for each room. The critical point is where the connectivity of a cell changes, for example when an obstacle divides the room. According to Choset and Pignon [2], fewer cells are better as fewer number of cells minimize the number of zig-zag motions the robot has to make, but more cells are guaranteeing the robot exhaustively covers the entire environment however this can be countered by making the boustrophedon algorithm have a shorter ″side step″.

Some previous research has been done in the field of comparing different algorithms for unexplored environments and while the best coverage per minute is achieved when using all the algorithms combined, boustrophedon is the next best performing algorithm [].

Boustrophedon cellular decomposition is where the environment is decomposed into smaller cells which makes it easier to get more coverage. Combining cellular decomposition with graph theory to generate an Euler tour guarantees complete coverage of the known work space while minimizing the traveled path by the robot. [5]. Using landmark-based topological coverage or grid-based methods improves the coverage of algorithms like boustrophedon [3].

2.1.1 Algorithms

This section contains background of the studies and used algorithms for this thesis.

Boustrophedon

Boustrophedon means ox-turning in ancient Greek, and mimics an ox when plowing and makes the robot zig-zag from opposite walls while traversing the room length wise[2][5]. The pseudocode (Algorithm. 1) is a direct translation from the boustrophedon flowchart from Hasan [4]. The algorithm starts by going upwards and then makes the robot cleaner turn 90 degrees right when the front bumper is triggered, and then travels a distance equal to the robot cleaners diameter and turns 90 degrees right again making the robot face downwards. The robot cleaner then travels downwards till the bumper is triggered again. The robot cleaner will now turn 90 degrees left, and then travels a distance equal to the robot cleaners length and turns 90 degrees left again making the robot cleaner face upwards. After this the algorithm restarts. The path generated by the algorithm is depicted by (Figure. 2).

```
Start direction up
Let one roboUnit = the diameter of the robot
if collision detected then
    Move backwards one roboUnit
    if if count is odd then
        Turn 90 degrees clockwise
        Move forwards one roboUnit
        Turn 90 degrees clockwise
        add 1 to count
    else
        Turn 90 degrees counter clockwise
        Move forwards one roboUnit
        Turn 90 degrees clockwise
        add 1 to count
    end
else
    Move forwards
end
```

Algorithm 1: Boustrophedon [4]

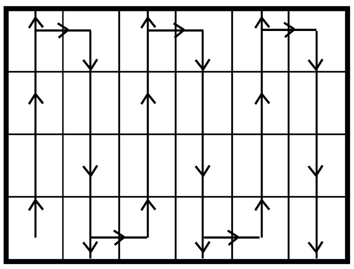

Figure 2: Boustrophedon

In the case where the room/space is known beforehand the boustrophedon algorithm gains a lot from dividing the room into smaller parts for better coverage. As the obstacles and furnitures in the room are known and divided into cells where the obstacles are, the robot cleaner can easily plan it's boustrophedon path for every cell. When all the cells have been visited and cleaned, the entire room is cleaned [2]. There is a difference between following the longer or the shorter wall of the room, there will be no difference in coverage however there are some small gains if the robot cleaner follows the long wall of the room as the robot cleaner does not have to make as many turns as when following the short wall which saves battery.

Spiral

The spiral algorithm makes the robot cleaner move in an increasing spiral path. This algorithm works best if the robot is placed in the middle of the room and has enough space to perform the spiral correctly. When the robot cleaner starts this algorithm it starts a right or left hand side spiral from the center point outwards until it the front bumper is triggered. The pseudocode (Algorithm. 2) is a direct translation from the spiral flowchart from hasan's report [4].

```
while no collision do
    Move forward
    Turn right according to turning angle
    Decrease turning angle
end
```

Algorithm 2: Spiral [4]

When that happens the robot cleaner stops the spiral algorithm and proceeds with another algorithm since starting the spiral algorithm when close to a wall is meaningless [4]. The path generated by the algorithm is depicted by (Figure. 3).

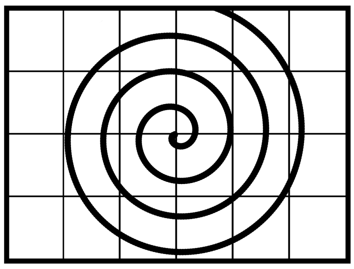

Figure 3: Spiral walk

Random walk

The random walk is a very simple algorithm that only needs a simple front bumper sensor. The robot cleaner just moves in any direction until it hits an obstacle and the front bumper is pressed. When the bumper is pressed the robot cleaner generates a pseudo-random number which decides how much the robot cleaner should turn. From here the algorithm starts over. (Algorithm. 3) is a direct translation from the random walk flowchart from hasan's report [4].

```
Move forward
Let one roboUnit = the diameter of the robot
if collision detected then
    Move backwards one roboUnit
    Turn 180 degrees + or - random(90 degrees)
else

end
```

Algorithm 3: Random Walk

(Figure. 4) depicts how one of the paths the generated by the algoritm might look like.

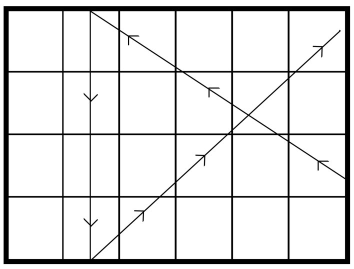

Figure 4: Random walk

Wall follow

The wall follow algorithm moves forward until the front bumper is triggered, at which point it turns 180 degrees away from the wall and then starts a circular motion until it hit the walls again.The algorithm repeats this procedure until the entire room has been covered along the Walls. This algorithm stays near the walls and never sways far away from them, this means it only cleans the perimeter of the room [4]. This algorithm is seldom used alone but instead combined with the spiral algorithm. (Algorithm. 4) is a interpretation of the wall follow flowchart from hasan's report [4] and the half circle motion is from my own robot cleaner [1].

```
Move forward
if collision detected then
    Turn 180 degrees
    Move forward with a pre set angle speed
else

end
```

Algorithm 4: Wall follow

(Figure. 5) depicts how one of the paths generated by the algoritm might look like.

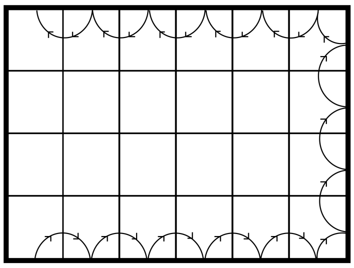

Figure 5: Wall follow

Spiral + Wall follow

This is a combination of the spiral and the wallfollow algorithm. When the front bumper is triggered while using the spiral algorithm, the robot cleaner switches to the wall follow algorithm. This is useful for when the robot cleaner is starting in the middle using spiral. The algorithm switches only once. So when going from spiral to wall follow there is no way to switch back to spiral.

All combined

The all combined algorithm is using all of the above described algorithms and using a timer to change between them. Starting with boustrophedon, switching over to random walk and from there switching to spiral and when the front bumper is triggered switching to wall follow and then starting the whole cycle over again.

2.2 Problem statement

Robots with only one bumper sensor have limited options in what algorithms can be used to get coverage in a room. *This work aims to perform a series of empirical tests to classify which of these algorithms get most coverage or if a specific combination of these algorithms is better than another combination.* My hypothesis is that Boustrophedon will perform better in every room that only has 90 degrees corners and with a maximum of 2 pieces of furniture. If that is the case and you have a sparse funitured room no algorithm except boustrophedon is needed to get a coverage of 80% as fast or faster than all the algorithms combined.

The robot cleaner in this simulation only has a front bumper which allows it to sense when it hits an obstacle, no other sensors are present. This means the robot cleaner has no knowledge of the world at the beginning and no knowledge at the end. Boustrophedon, spiral and wall follow are all deterministic algorithms while random walk is not. Floor space will be represented by white pixels and walls and obstacles will be black pixels and already cleaned area will be painted with red pixels.

Chapter 3

Method

3.1 Simulation

By making a simulation of a robot cleaner with only a bumper sensor and providing it with a room, the simulation is able to compare and present which algorithm is best suited for that type of room. Other ways to test this by using a real robot cleaner, and clean real rooms. As far as we know there are no funding for this, and making a simulation makes it possible to try out each algorithm separatly and also come up with new algorithms.

The simulation is written in Python and is a graphical implementation of the robot and the room, the robot only has one sensor which is a front bumper that spans 120 degrees on the front of the robot [4].

The robot cleaner leaves a trail of red to show where the robot cleaner has cleaned (Figure. 6). Due to how computational heavy counting colored pixels, every second the algorithm runs the simulation counts how much coverage the robot has accumulated.

At 100 seconds running time, the simulation ends and the results are presented to the user.

The algorithms used in this simulation are constructed using the flow charts found in Path Planning Algorithm Development for Autonomous Vacuum Cleaner Robots [4].

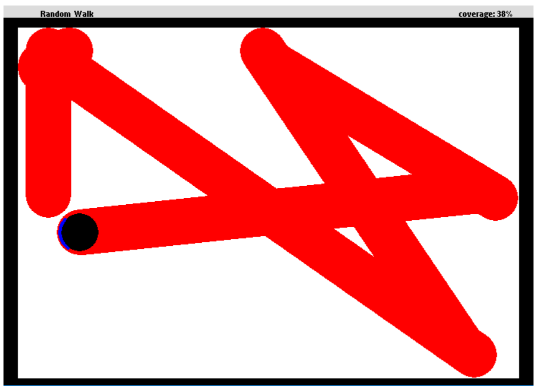

Figure 6: The simulator in running the random walk algorithm

3.1.1 Rooms

There are four different types of rooms, two of which are entirely empty and two that are sparsely furnitured. Corners are only angled at 90 degrees. The rooms are in the form of a bitmap image file which makes it easy to create new rooms for the simulator to run.

This is the standard room, rectangular and empty.

Figure 7: Room 1

This "room" consists of several spaces with half walls and small corners that divide the room.

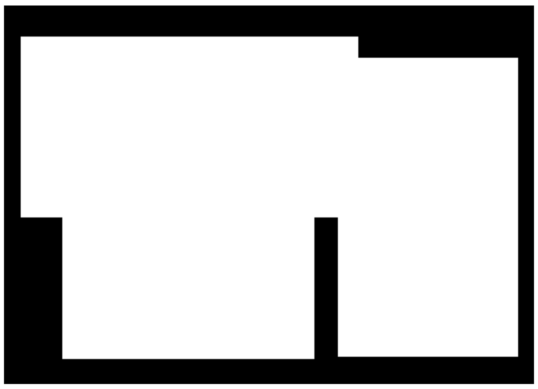

Figure 8: Room 2

This is the standard room but furnished with a small sofa and an arm chair. The sofa and the arm chair are represented by four points in the room since that is where the legs are. The sofa and the arm chair are tall enough for the robot cleaner move underneath them.

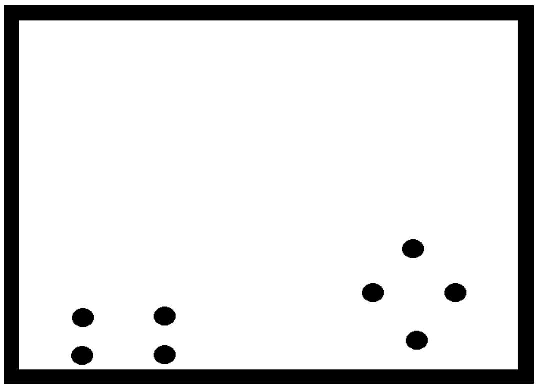

Figure 9: Room 3

This narrow passages with a sofa and an arm chair.

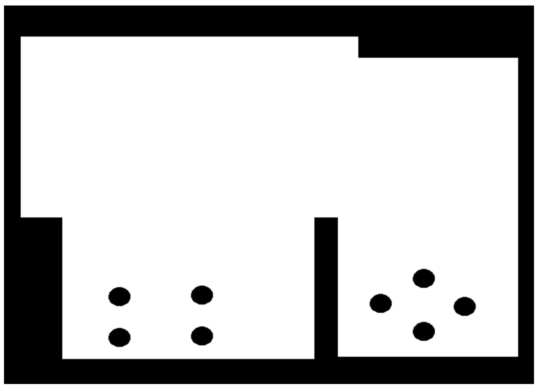

Figure 10: Room 4

The simulation starts by counting the area of the room by counting the white pixels inside of the walls, then the algorithm to use is chosen.

3.2 Benchmarks

As a benchmark, the simulator will run all four of the algorithms with a timer to switch between them. This is what some cheap robot cleaners use [4]. This is used to as a control to compare if any of the other algorithms are good enough to be used alone.

Chapter 4

Results

The results come from running the simulation several times for each type of room. The non-deterministic algorithms were tested by computing the mean of 10 separate runs, while the deterministic algorithms were tested with a single run. The non-deterministic algorithms are Random Walk and All Combined, all other algorithms will get the same result every time. The reason some algorithms get stuck at a certain percentage of coverage is that the robot cleaner got stuck and this is how the algorithms should work acording to the flowchart [4].

4.1 Room type 1

These are the result from room type 1, the empty room. Starting in a corner (Figure. 11), and starting in the middle of the room (Figure. 12).

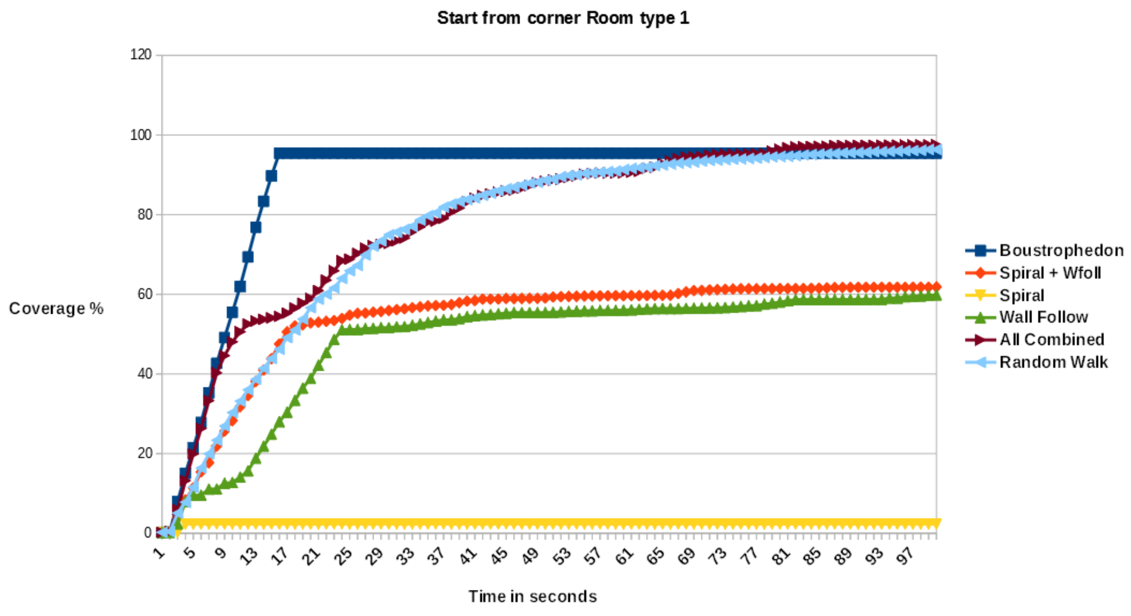

Figure 11: Results when starting in a corner in room 1

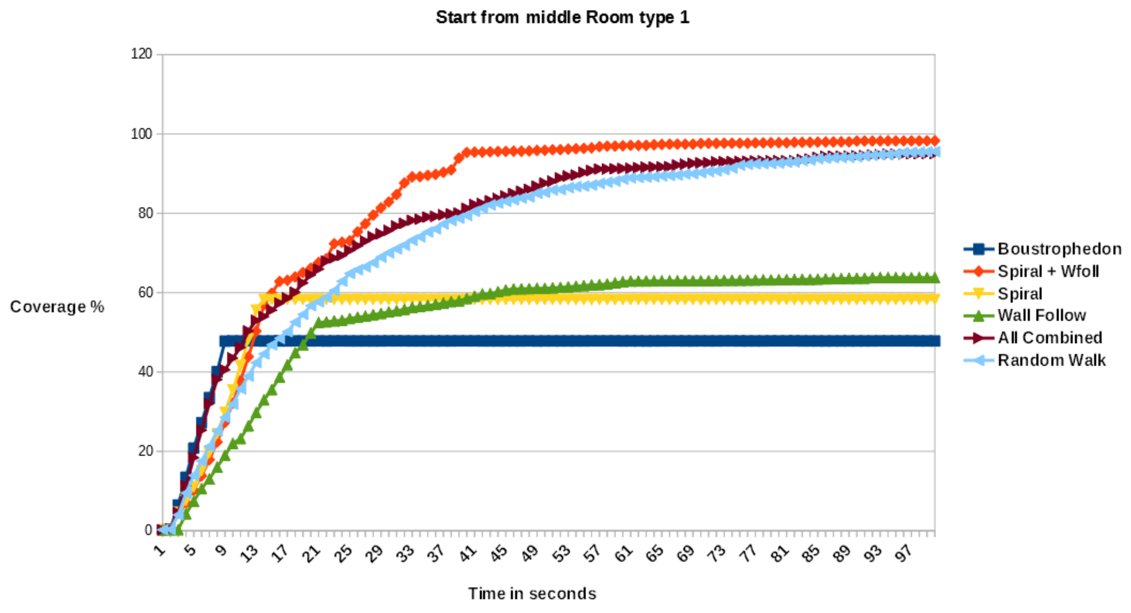

Figure 12: Results when starting in the middle in room 1

4.2 Room type 2

These are the result from room type 2, the room with half walls and small corners. Starting in a corner (Figure. 13), and starting in the middle of the room (Figure. 14).

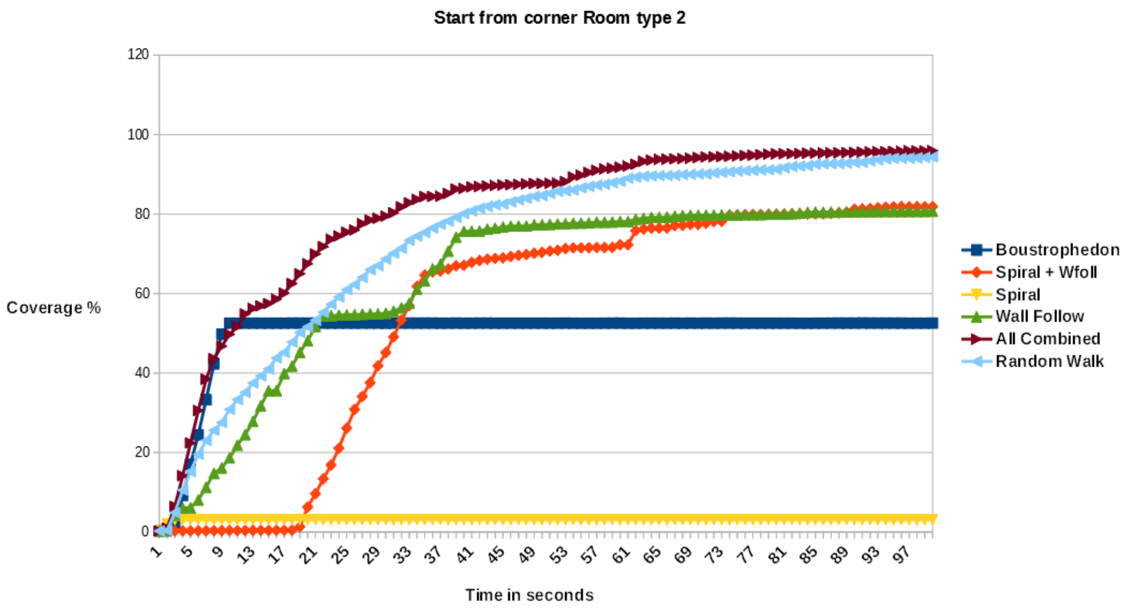

Figure 13: Results when starting in a corner in room 2

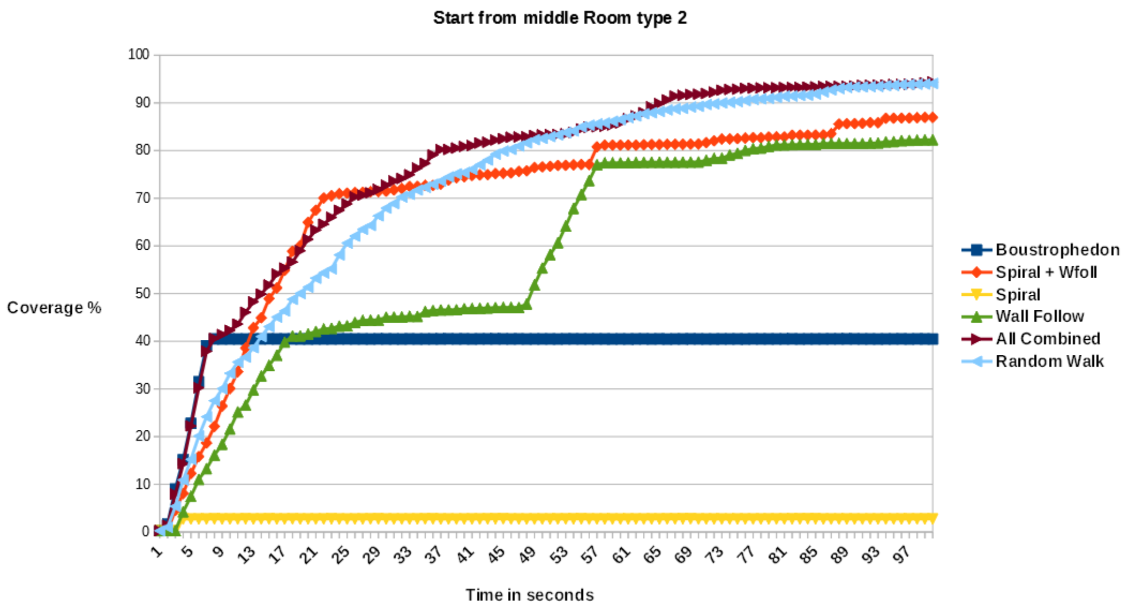

Figure 14: Results when starting in the middle in room 2

4.3 Room type 3

These are the result from room type 3, the standard room but furnished. Starting in a corner (Figure. 15), and starting in the middle of the room (Figure. 16).

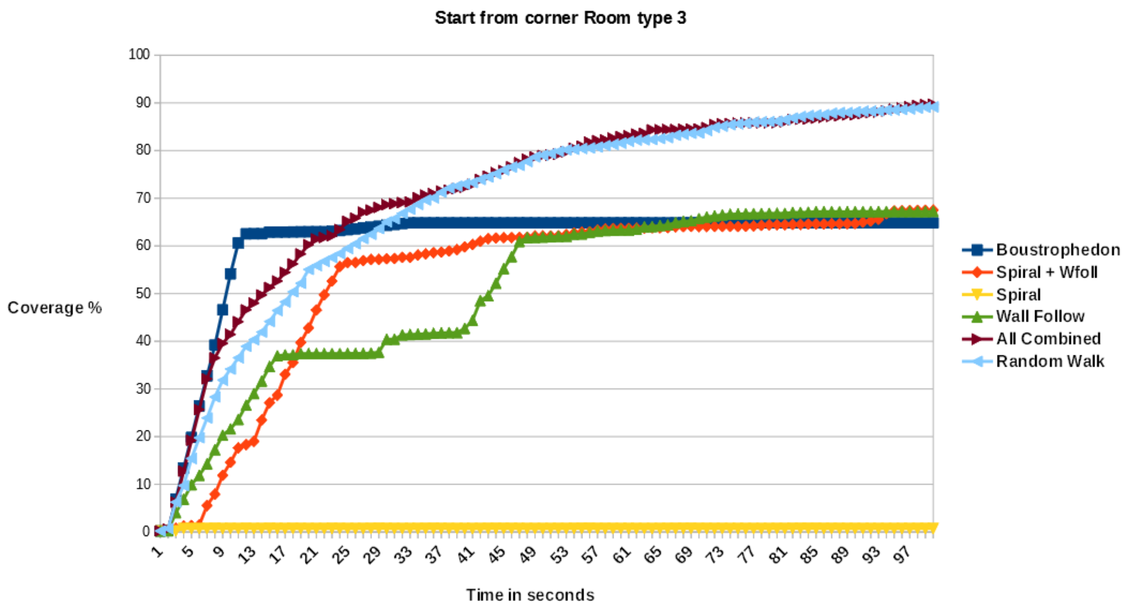

Figure 15: Results when starting in a corner in room 3

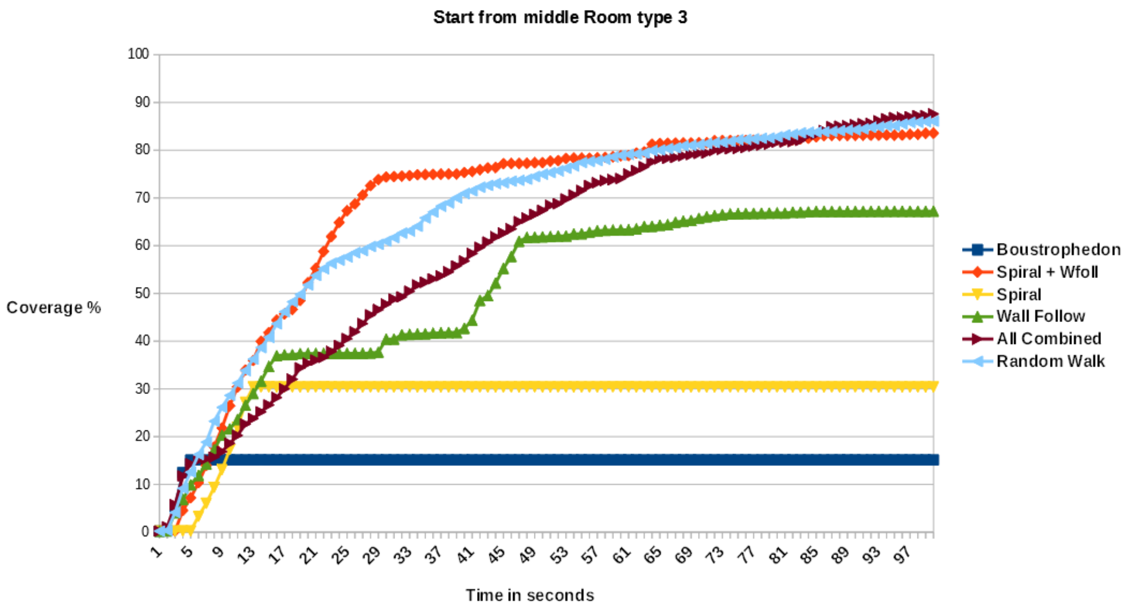

Figure 16: Results when starting in the middle in room 3

4.4 Room type 4

These are the result from room type 4, the room with half walls and small corners when furnished. Starting in a corner (Figure. 17), and starting in the middle of the room (Figure. 18).

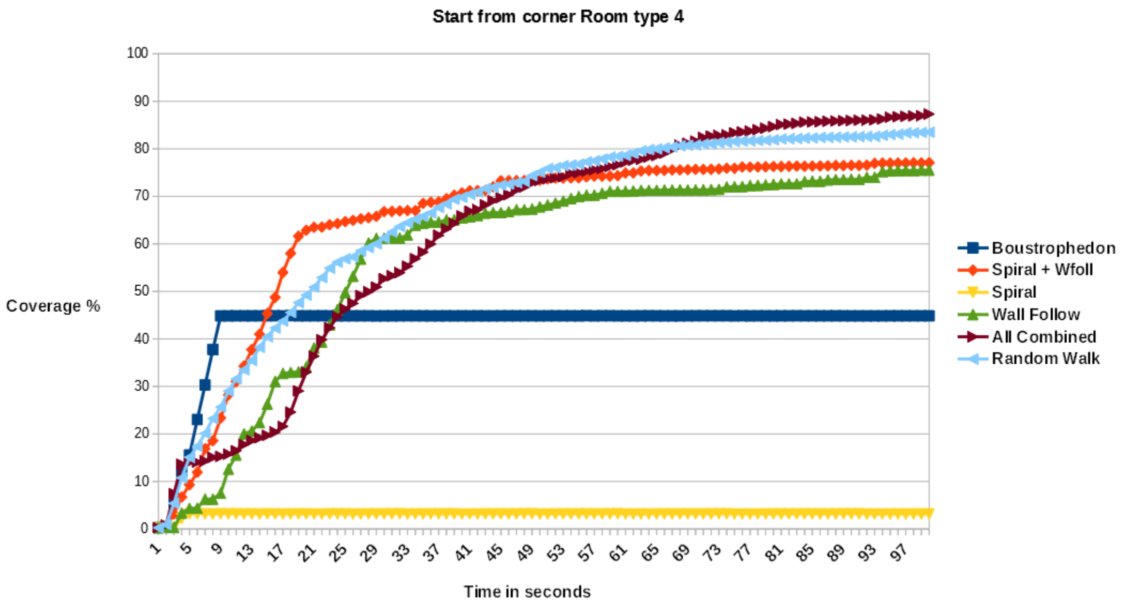

Figure 17: Results when starting in a corner in room 4

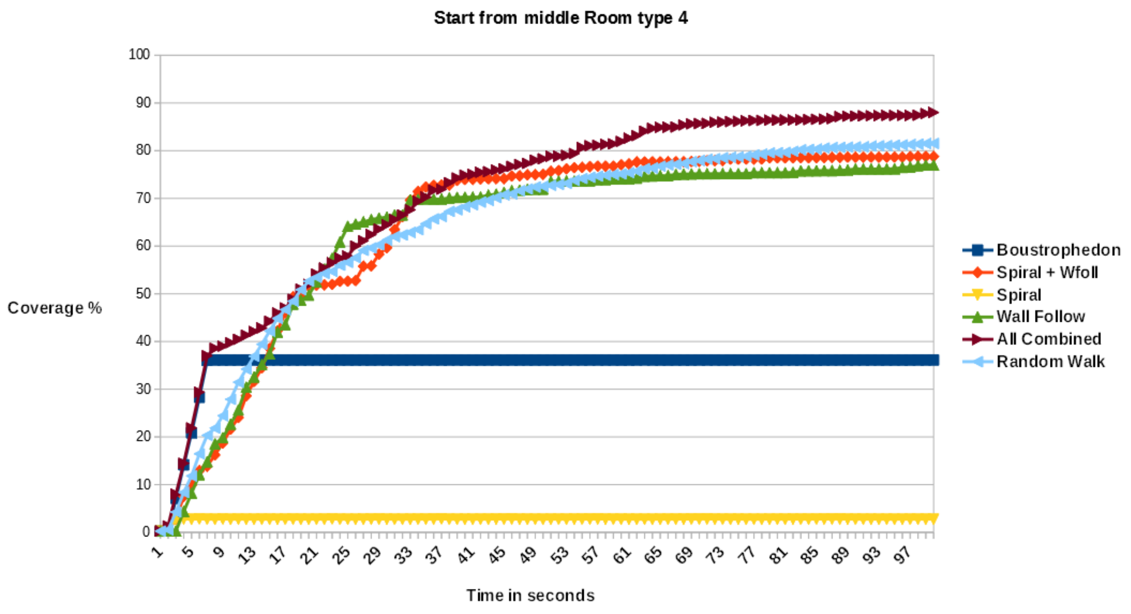

Figure 18: Results when starting in the middle in room 4

Chapter 5

Discussion

Most of the time, the non-deterministic algorithms got stuck and could not move past the obstacle in their way. Especially spiral which is terrible if the robot cleaner does not start near the middle of the room. Even boustrophedon got stuck a lot of time, and when the robot cleaner hit the most right wall the boustrophedon could not do anything. The implementation I found of the boustrophedon algorithm did not really meet my expectations, maybe that is just because I did not have enough information about the boustrophedon. However random walk surprised a whole lot. I originally thought that random walk would travel where it has already been befor and therefore clean already clean space, and it does. However it is not that ineffective.

[discuss improvements to boustrophedon.]

Chapter 6

Conclusion

Boustrophedon by itself is no enough to clean rooms, unless the room is totally empty and the robot cleaner starts in the southern most left corner of the room.

[write about the idea of improved boustrophedon]

Bibliography

[1] *Centurion RDE 125*. `http://robotdammsugare.org/centurion-rde125/`. [Online; accessed 24-Mars-2018]. 2018.

[2] Howie Choset and Philippe Pignon. "Coverage Path Planning: The Boustrophedon Decomposition". In: *International Conference on Field and Service Robotics*. Jan. 1997.

[3] Enric Galceran. "A survey on coverage path planning for robotics". In: *Robotics and Autonomous Systems* 61 (2013), pp. 1258–1276.

[4] Kazi Mahmud Hasan. "Path Planning Algorithm Development for Autonomous Vacuum Cleaner Robots". Electronics and Communication Engineering Discipline Khulna University, Bangladesh, 2014.

[5] Ioannis Rekleitis. "Efficient Boustrophedon Multi-Robot Coverage: an algorithmic approach". School of Computer Science, McGill University, Montreal, Canada, 2009.

[6] *Roomba - Dammsugartyper*. `http://www.irobot.se/Home-Robots/Dammsugning`. [Online; accessed 16-February-2018]. 2018.

[7] *Scantic robot cleaner 10*. `http://robotnyheter.se/2013/12/04/ny-robotdammsugare-for-499-kronor-hos-netonnet/`. [Online; accessed 16-February-2018]. 2018.

Appendix A

Unnecessary Appended Material