A performance survey of coverage algorithms for simple robotic vacuum cleaners

ROBIN GUNNING

Bachelor's degree in Computer Science
Date: May 3, 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]. The cheaper models often only have a single sensor which is a frontal bumper (Fig. 1.1), a micro switch which gets pressed when the robot cleaner bumps in to a wall or another object.

Figure 1.1: Skantic Robot Cleaner 10 [7]

The 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 probably have different strengths depending on what type of room it is.The standard algorithms for cheap robot cleaners are Boustrophedon, random walk, spiral and wall follow [3].

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.

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 [3].

1.1 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 cell. The critical point is where the connectivity of a cell changes, for example when a obstacle divides the room. According to Choset, [1] fever cells are better 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 [3].

Boustrophedon cellular decomposition was combined with graph theory to generate an Euler tour which 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 [2]

1.2 Background

1.2.1 Algorithms

This section contains background of the used algorithms.

Boustrophedon

Boustrophedon means ox-turning in ancient Greek, mimics an ox when plowing or bi-directional text [1][5]. The algorithm makes the robot cleaner turn 90 degrees right/left when the front bumper is triggered, and then travels a distance equal to the robot cleaners diameter and turns 90 degrees right/left again. The robot cleaner then travels forward till the bumper is triggered again. This time the robot cleaner turns 90 degrees the direction it didn't turn before, and then travels a distance equal to the robot cleaners length and turns 90 degrees in the same direction. After this the algorithm restarts.

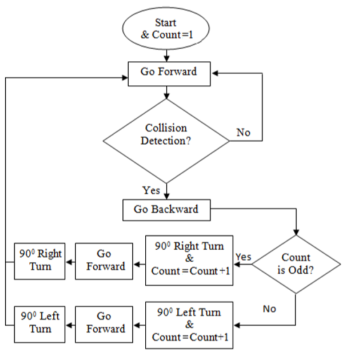

Figure 1.2: Flow chart of the boustrophedon algorithm [3]

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 [1].

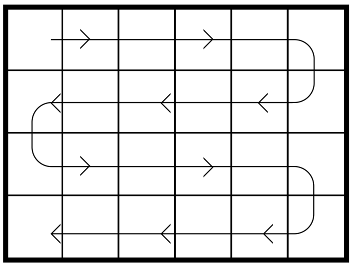

Figure 1.3: Boustrophedon

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 start this algorithm it starts a right or left hand side spiral from the center point outwards until it the front bumper is triggered.

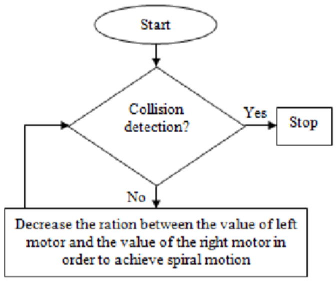

Figure 1.4: Flow chart of the spiral algorithm [3]

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. [3]

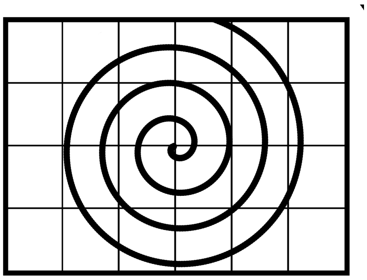

Figure 1.5: Spiral walk

Random walk

The random walk is a very simple algorithm that does not need any sensors but the simple front bump 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.

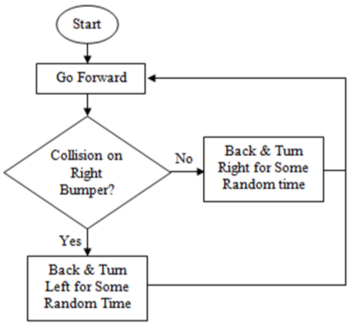

Figure 1.6: Flow chart of the random walk algorithm [3]

This is how one of the paths the algoritms might look like.

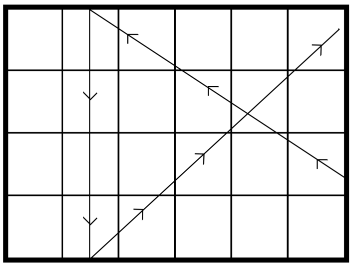

Figure 1.7: Random walk

Wall follow

The wall follow algorithm moves in a circular path and when the front bumper is triggered it turns away from the wall until the bumper is no longer pressed and then starts the algorithm over again until the entire room has been covered along the Walls.

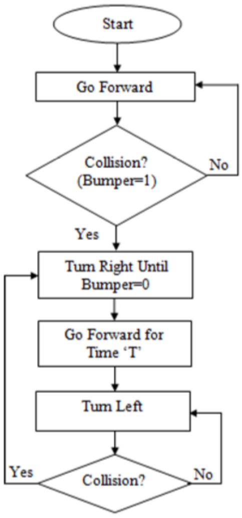

Figure 1.8: Flow chart of the wall follow algorithm [3]

The wall follow will follow the walls of the given room looking like this.

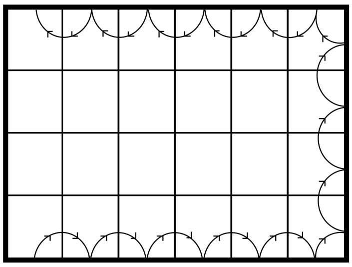

Figure 1.9: Wall follow

1.3 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 without a lot of obstacles. If that is the case and you have a sparse funitured room no algorithm except boustrophedon is needed to get a acceptable coverage of 80%.

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 2

Method

2.1 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. Unfortunately 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 100 degrees on the front of the robot. The simulation start by counting the area of the room by counting the white pixels inside of the walls, then the algorithm to use is chosen. The robot cleaner leaves a trail of red to show where the robot cleaner has cleaned. At every second the algorithm runs the simulation counts how much coverage the robot has accumulated.

At 80% coverage or 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 from hasan [3].

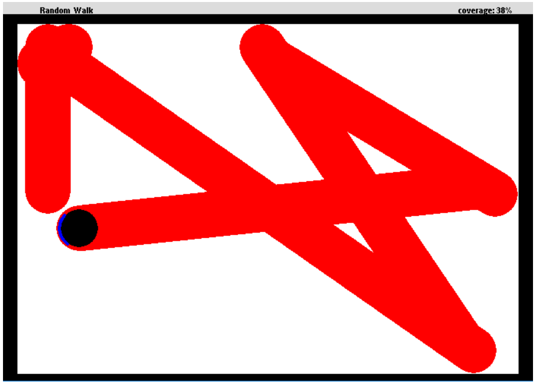

Figure 2.1: The simulator in running the random walk algorithm

2.1.1 Rooms

There are four different types of rooms, two of which is 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 2.2: Room 1

This ”room” consists of several spaces with narrow passages.

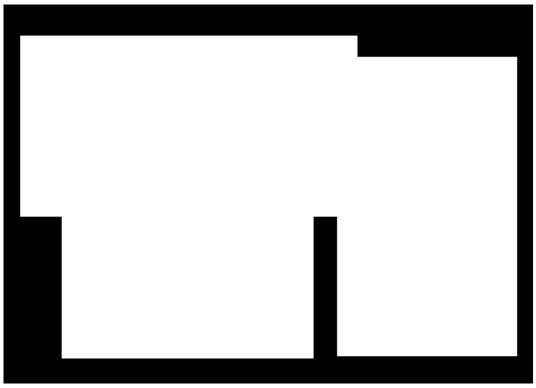

Figure 2.3: Room 2

This is the standard room but furnished with a small sofa and an arm chair.

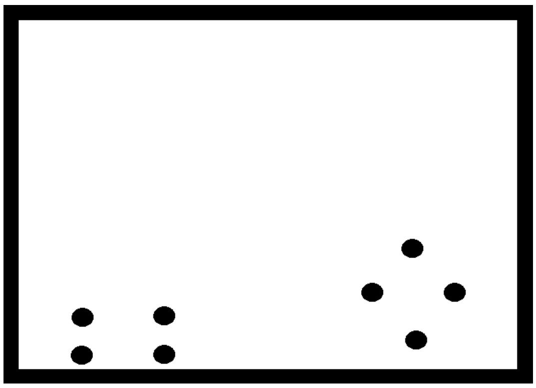

Figure 2.4: Room 3

This narrow passages with a sofa and an arm chair.

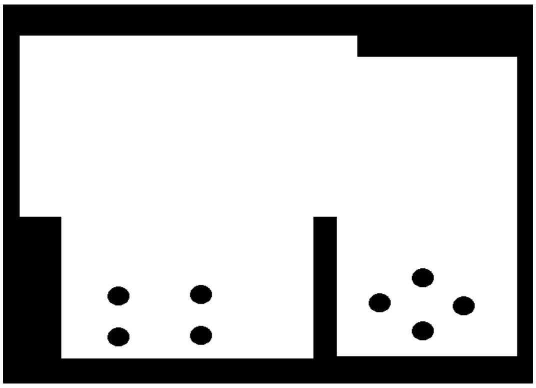

Figure 2.5: Room 4

2.2 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 [3]. [more about benchmark]

Chapter 3

Results

[more describing the results]

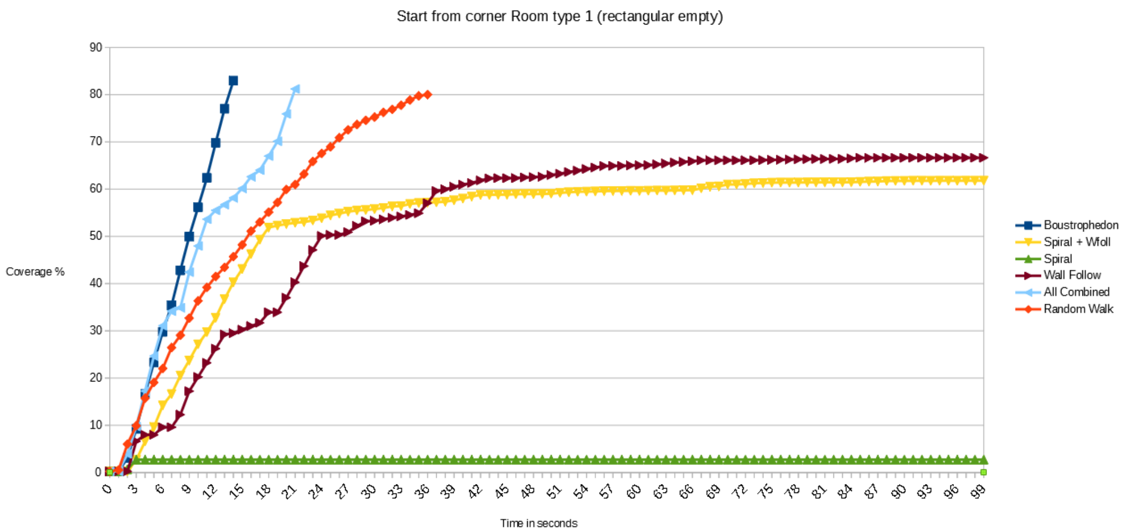

Figure 3.1: Results when starting in a corner in room 1

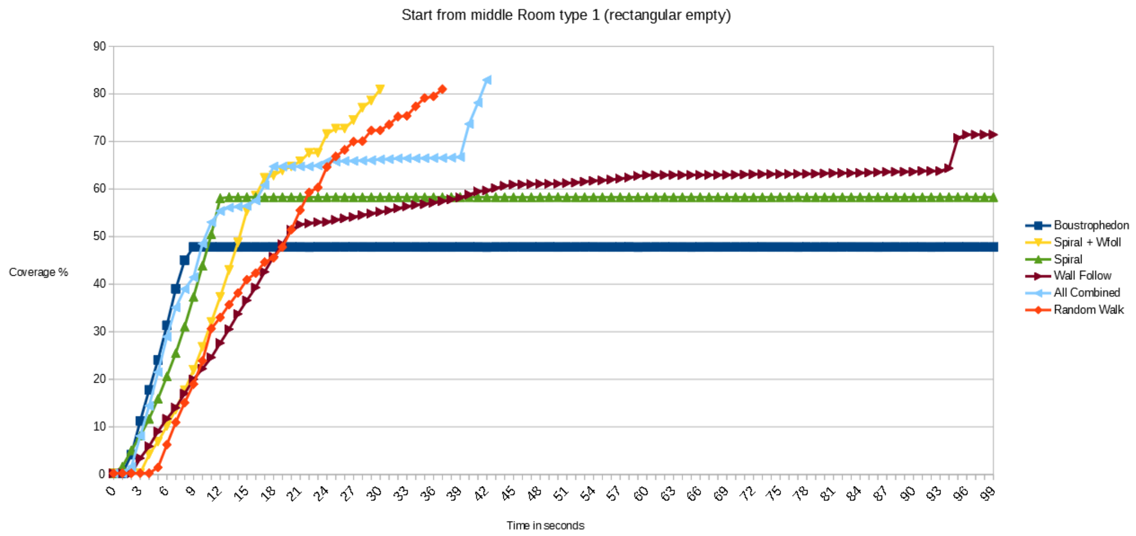

Figure 3.2: Results when starting in the middle in room 1

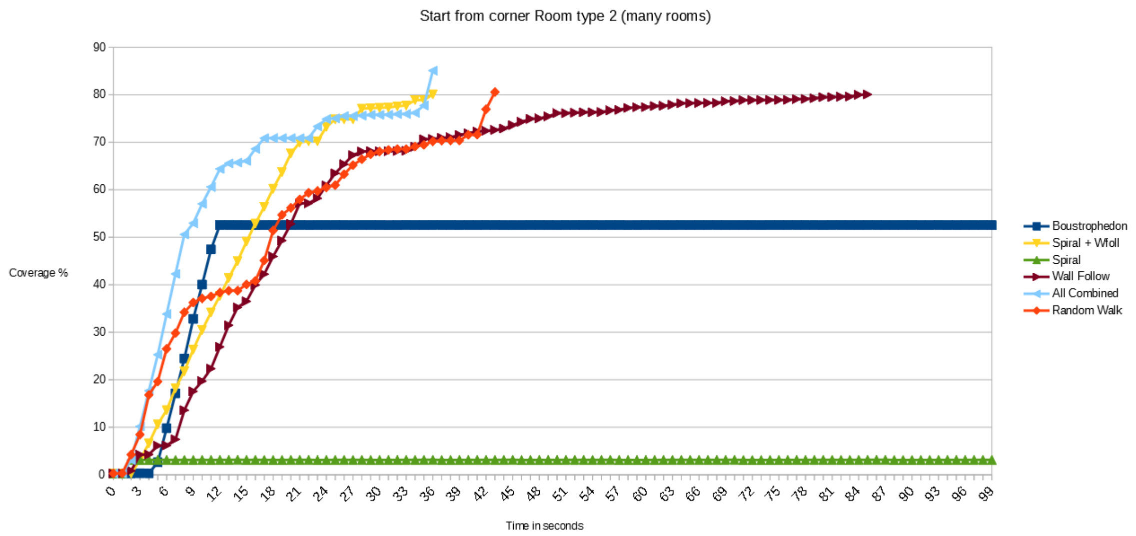

Figure 3.3: Results when starting in a corner in room 2

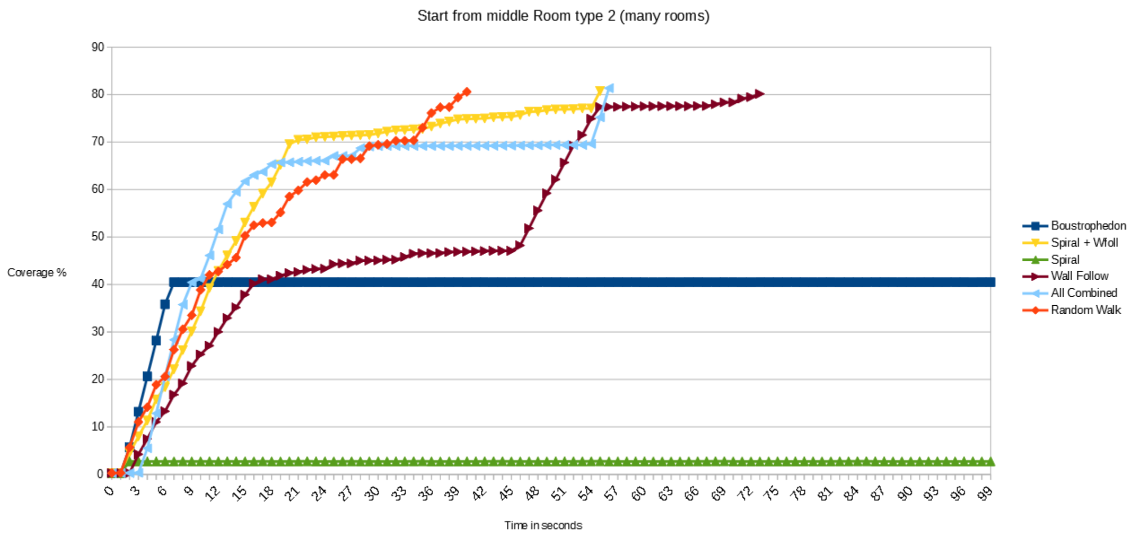

Figure 3.4: Results when starting in the middle in room 2

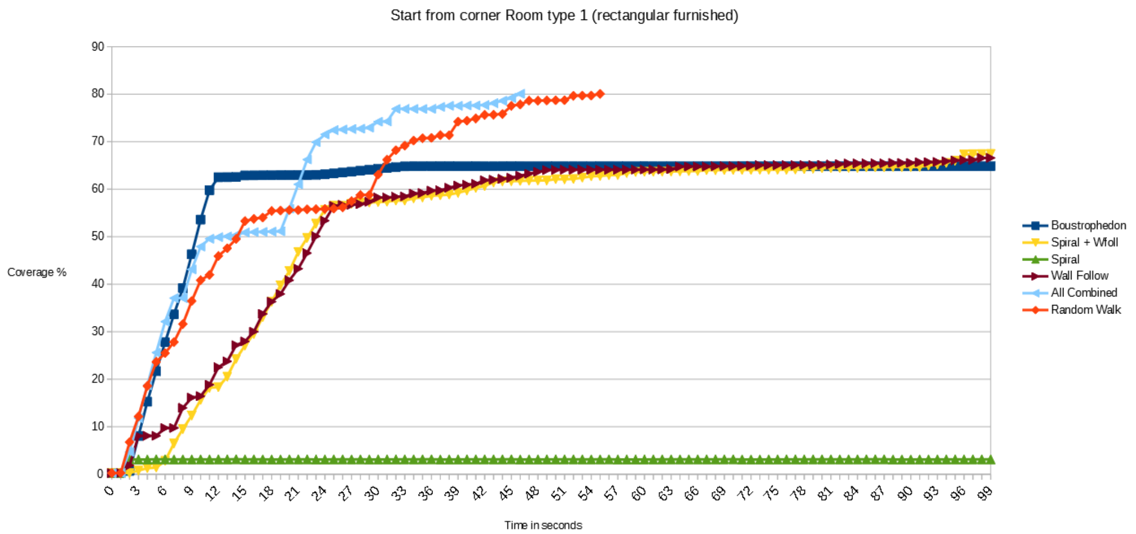

Figure 3.5: Results when starting in a corner in room 3

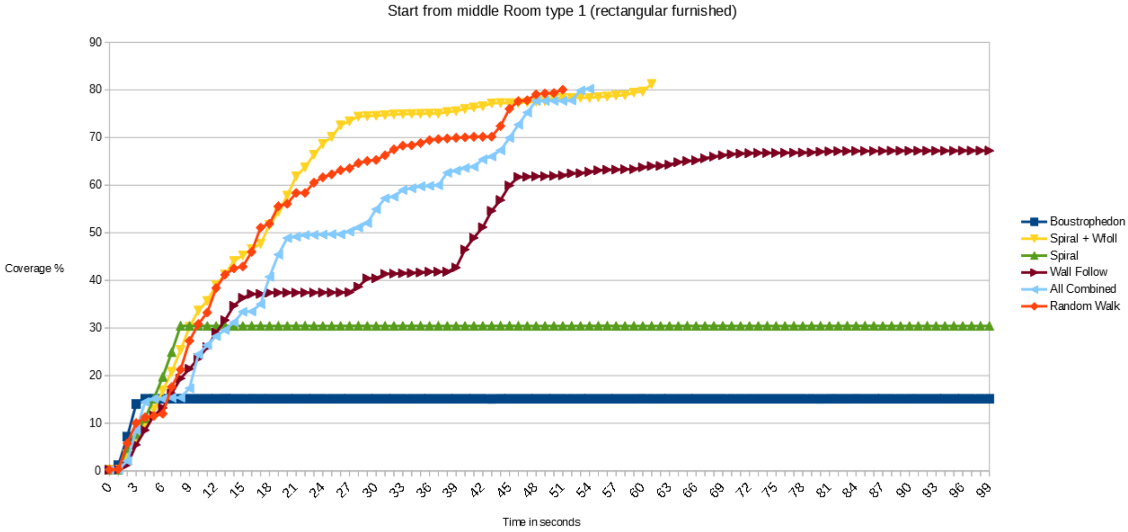

Figure 3.6: Results when starting in the middle in room 3

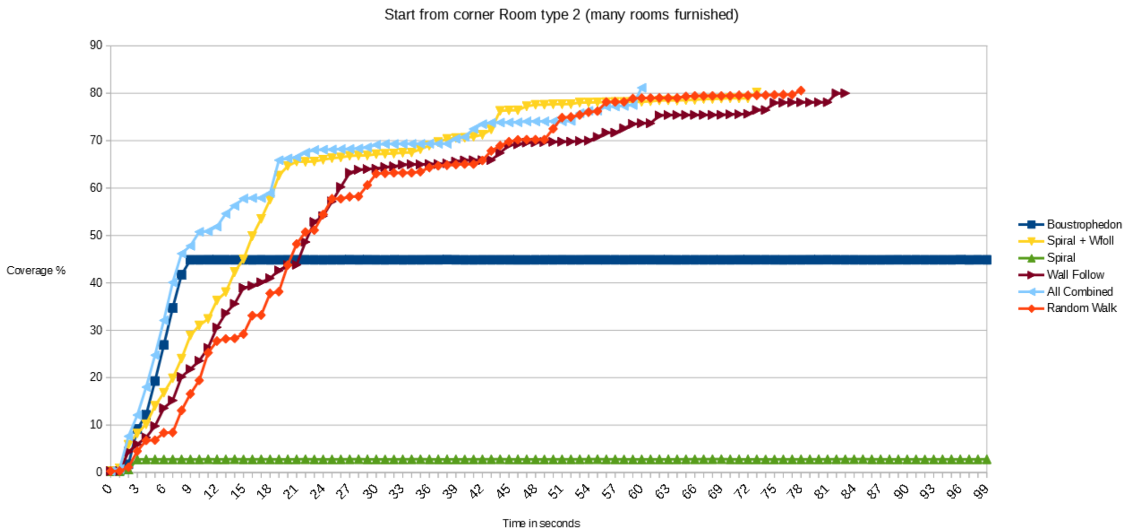

Figure 3.7: Results when starting in a corner in room 4

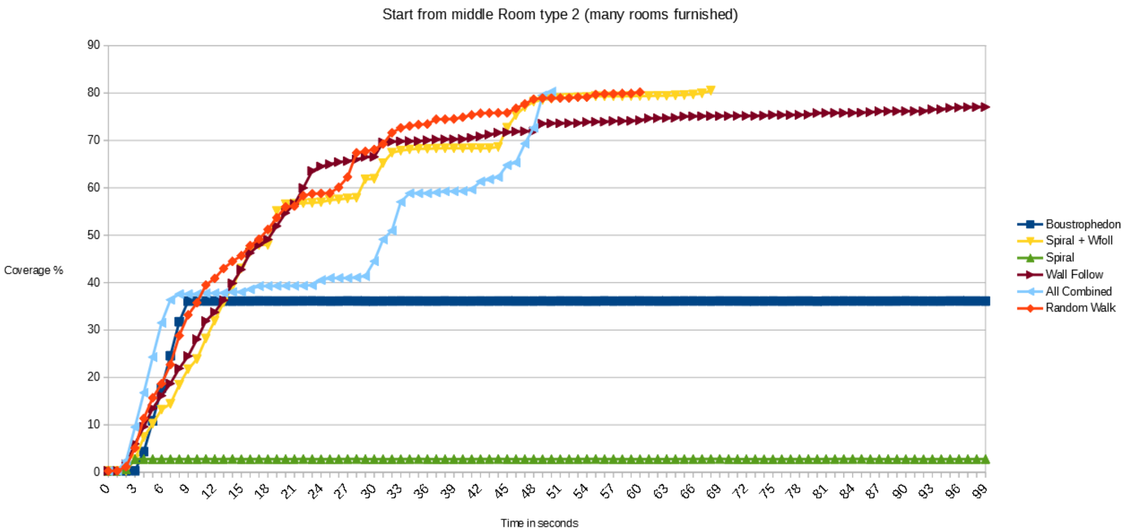

Figure 3.8: Results when starting in the middle in room 4

Chapter 4

Discussion

[Write about how bad the boustrophedon implementation is and how effective random walk is, maybe something about improvements to boustrophedon.]

Chapter 5

Conclusion

[write about how wrong i was. However it might be true for the improved boustrophedon etc ect.]

Bibliography

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

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

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

[4] *Henry George Liddell, Robert Scott, A Greek-English Lexicon*. `http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0057:entry=boustrofhdo/n`. [Online; accessed 16-February-2018]. 2018.

[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