Multi-Agent House-Cleaning Environment: Vacuum Cleaner and Dust-bin

Alexey Markin, Anugrah Saxena, XinXin Yang

amarkin@iastate.edu, anugrah@iastate.edu, yangxin@iastate.edu

Abstract

This project introduces an abstract model for autonomous cleaning task in a smart home environment. We consider two cleaning agents: Vacuum Cleaner and a Smart Dustbin. The project aims at comparing the performance of a Vacuum Cleaner in a single-agent environment to the performance of the two agents (Vacuum Cleaner and Smart Dustbin) in a multi-agent environment. For our project, we made certain assumptions for Vacuum-Cleaner (VC) and Smart Dustbin (DstB), such as, constant number of cleaning actions per room, capabilities for VC and DstB, etc. The VC agent uses a heuristic search for planning actions ahead of time. However, due to the unpredictable dynamic environment, we implement the Incremental Heuristic Search approach to handle different problems. The DstB agent can reach for VC and gather dirt collected by it. In addition, it might go to a designated Dump room to empty the collected garbage. The designed simulation environment is able to feed percepts to the agents and update model based on agents' actions. Meanwhile, the smart home environment is used to facilitate the communication between the agents and coordinate them. The experimental evaluation shows that introduction of a Smart Dustbin agent allows us to significantly improve the overall performance in house-cleaning task.

1. Introduction

Robot vacuum cleaners have been on the market for almost two decades, ever since the late 90's. As for today, the robot vacuums have several advanced features: they could be manipulated with mobile apps via Internet; could work continuously for couples of hours then recharge and resume cleaning; could use a full suite of sensors to map and adapt to real world clutter and furniture for thorough coverage. Introducing robot vacuum cleaners to human being's life can help people maintain a cleaner environment on a daily basis, so people can focus on things of possibly higher importance.

A robot vacuum cleaner can sense obstacles to avoid travel room by room and clean the floor. However, there are still a few things that require human attention. One significant disadvantage, as we see it, is that people have to empty the dust container when it is about to get full, otherwise, a vacuum cleaner will not be able to finish the assigned cleaning job. In addition, people want rooms to be cleaned as soon as they get dirty without commanding the vacuum cleaner every time.

Therefore, in our project we propose the integration of vacuum cleaner agent to the smart home environment. Smart Home Environment is assumed to be able to gather relevant information, such as current occupancy of rooms by inhabitants. Moreover, the smart home is capable of detecting when a room should be cleaned. The agent has to cope with dynamic and stochastic environment, so it uses Incremental Heuristic Search to find optimal action plans (see section 3).

Further, we introduce a second agent - Smart Dustbin that works in parallel with Vacuum Cleaner in a Smart Home Environment to help it perform the cleaning task. Dustbin has the functionality to locate the vacuum cleaner, load trash, and travel to the designated room to dump the trash. Vacuum cleaner and Dustbin communicate with each other through Smart Home in order to complete the cleaning job. The entire system aims at our agents to work collaboratively to improve the overall performance.

1.1. Environment Model

We assume that our environment could be represented as a connected undirected graph: nodes representing rooms; edges representing pathways. We also assume that Smart home sensors can provide full information about the room-occupancy by the inhabitants.

- Vacuum-cleaner will not clean any room occupied by inhabitants; Vacuum-cleaner will stop working as soon as it is given a percept that inhabitants enter current room (we are making these restrictions in order to prevent any unanticipated incidents [4]);
- The vacuum-cleaner has the capability to transfer trash to the dustbin;
- In order to clean a room VC should initially perform a constant number of discrete cleaning actions

- Each action for both agents takes a constant amount of time (for example, one minute)

For clarity, we describe the agents' environment properties as follows:

- Fully observable: We assume the Vacuum-Cleaner and Dustbin work in a Smart-Home Environment where sensors can detect all aspects that are relevant to assign the actions for both agents;
- Stochastic: Occupancy states of the rooms is unpredictable;
- Dynamic: The environment can change anytime while both agents perform actions;
- Multi-agent: We have two agents - a Vacuum Cleaner and a Smart Dustbin;
- Discrete: agents' actions are performed per a single discrete time step.

2. Vacuum Cleaner

2.1. Vacuum Cleaner agent

Vacuum Cleaner agent (VC) cleans the house based on the percepts given to it by the Smart Home Environment. Its lifelong goal is to keep the house clean, that is, at a particular moment of time we want to maximize the number of rooms cleaned. In addition, we want Vacuum Cleaner to "save" energy: do minimal number of actions needed to perform its task. In our model, the Vacuum Cleaner performs the following actions:

- CLEAN current room
- MOVE to an adjacent room
- DUMP dirt, if it is currently in a Dump room, or Dustbin agent (described in section 3) is in the same room as VC, and requests a trash transfer.
- STAY - do nothing for the next time step

Since the environment is discrete, all these actions are taken in single time-step (for example, one minute). Note that in our model in order to fully clean a room, Vacuum Cleaner needs to perform A cleaning actions, that is, clean a room for A time steps. Other actions, such as MOVE and DUMP are assumed to be fully performed in a single time step.

In addition, we assume that each cleaning action increases the load (gathered dust/trash) by a constant number of units T. While the total trash capacity of the agent is denoted by C. When the agent reaches its full capacity (cannot perform any cleaning actions), it should either go to a Dump room to release trash, or transfer its load to a Dustbin agent in a multi-agent environment.

The major restriction that we enforce on VC is that it cannot clean a room when it is occupied - this is partially motivated by [4]. In our opinion, this approach will be helpful in future smart home environments where any inhabitant can work without any sort of hindrance due to autonomous agents functioning in the house. Note also that this restriction requires VC to stop cleaning, when someone enters the room. For our simulations, we also consider two cases: VC could visit the occupied rooms (but not clean them aka non-restricted setup), and VC can never enter an occupied room (restricted setup).

2.2. Vacuum Cleaner Problem

At a certain time step, some of the rooms are dirty and non-occupied. The goal for Vacuum Cleaner is to find an optimal plan of actions in order to clean all such rooms. In general, we need to find the shortest path in the house graph that visits all the nodes (rooms) in order to clean them - we will refer to this problem as VC problem.

Claim: VC Problem is NP-hard

Proof: We are going to show a reduction from Hamilton Path problem to VC problem. Assume we are given a graph $G = (V, E)$ and asked to determine whether it contains a Hamilton Path. Then we mark all the rooms in a graph as dirty, and solve a VC problem for every possible starting node $v \in V$ If for any starting position $v \in V$ VC solver has found a simple path (does not visit any vertex more than once), then we have clearly found a Hamilton Path. Otherwise, G does not contain a Hamilton path. We can prove it by contradiction: if there exists a Hamilton Path for G starting from vertex v_0, then this path is a shortest possible path connecting all the nodes. Therefore, VC solver was supposed to find it - contradiction.
Summing it up, we see that VC problem is NP-hard.

Reduction to Travelling Salesman Problem

One of possible ways to solve the VC Problem is to reduce it to a Travelling salesman problem (TSP). We denote a set of vertices that are dirty and non-occupied by V_d. Assume that VC is located at node v_0. Then we create a new full graph $G' = (V_d \cup \{v\}, F)$. The vertex set for G' contains all the nodes that a solution to VC problem should visit. For any edge $e = (u, v) \in F$ we set the weight e to be the length in edges of a shortest path from u to v in G.

Proposition: A solution to a TSP problem on G' gives us a solution to the initial VC problem.

Proof idea: Assume that TSP solver found an optimal path P. Then, we substitute all the edges (u, v) in P by a shortest path from u to v ($P_{u,v}$) in the initial graph G. That way we obtain an augmented path P' for the initial graph.

We omit the full proof, because our main approach to solving the VC Problem is through A* search.

2.3. Heuristic Search for VC Problem

The advantage of A* heuristic search is that it could be both fast (for a good heuristic function) and exact (if the function is consistent). To represent a state for a Vacuum Cleaner we use the following parameters:

- Load (an integer in interval $[0, C]$);
- Current room: vertex in a house graph;
- Occupancy states for all the rooms (nodes);

- Remaining cleaning steps for each room (each value is an integer in interval $[0, A]$).

We are using the following heuristic function to estimate states:

$$H(state) = (roomsToClean - 1) \cdot moveCost + remainingCleanAction \cdot cleanCost$$

Here, $roomsToClean - 1$ is the lower bound on a number of MOVE actions, and $remainingCleanAction$ is the exact number of CLEAN actions that VC has to perform in order to clean the whole house (except for those rooms that are currently occupied). This heuristic function is admissible by the construction - it defines a lower bound on the needed actions cost. In addition, this function is clearly consistent.

The major issue with the VC Problem is that the environment is dynamic and stochastic. Therefore, the "planned" A* actions might become unavailable over time, or some new actions might become available, e.g. dirty room becomes non-occupied. Therefore, sometimes we need to rerun the A* algorithm to get a new action plan. This approach is called Lifelong Planning [2] or Incremental Heuristic search in AI literature. In our project, we use the following strategy for rerunning A*:

- In case new action becomes available we rerun A* right away;
- In case some action becomes unavailable, we delay the A* re-execution until the state, where we cannot perform the planned actions anymore (lazy update).

This approach is partially taken from the Generalized Adaptive A* algorithm [1].

3. Dustbin

Dustbin agent

In a multi-agent environment, the inclusion of a smart dustbin agent is to facilitate smooth working of VC, i.e., a Vacuum Cleaner can continue to do its cleaning task without constant need to go to a designated Dump room. The Dustbin agent (DstB) waits for VC's communication signal to reach its location and collect garbage from it. Later, when DstB gets full it dumps its garbage to the disposal (Dump) room. Note also that a Dustbin agent can move around the smart home even if the rooms are occupied. In our model, a Dustbin (similarly to Vacuum Cleaner) can perform the following actions per time step:

- MOVE to an adjacent room;
- LOAD trash from a VC agent;
- DUMP trash in a designated Dump room;
- STAY - does not perform any action for the next time step.

Dustbin problem

We propose the following strategy for a Dustbin agent:

1. When Dustbin gets full, it needs to find the shortest path to the Dump room
2. When Dustbin receives the percept that VC has reached its capacity, it finds the shortest path between its current location and VC location;

Note that this approach, as well as our VC strategy could be easily extended to weighted graphs, that is, when we need different number of MOVE actions to navigate between rooms.

Dustbin algorithm

1. Shortest path to Dump room. Note that to find a shortest path from any room to a Dump room we need to perform a Breadth-First search once, starting from the Dump room, because the house topology does not change over time
2. Shortest path to VC is calculated through a single BFS run as well.

4. Experimental Evaluation

In order to carry out the evaluation of our system, we created a simulation of house map representing a graph (see picture below) where rooms were represented as nodes of the graph and pathways were represented as edges. We had a total of 10 rooms to clean and a specific dump room to collect all the trash bags. The initial state of the system had all the rooms as dirty and VC & DstB were placed in room 1. Both the agents were fed percepts by the Smart Home Environment at every time step based on random schedule generated by our system.

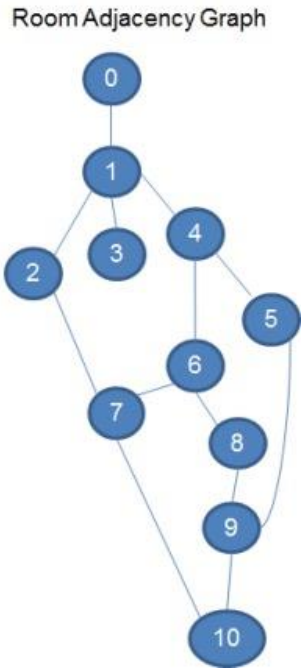

Figure 1. Room adjacency graph

4.1. Performance Measure

Our approach to calculate performance of the system is based on the total number of cleaning steps taken by our system agents (numerator of the performance function) and the total cost for actions taken so far (denominator). In that way, we are taking into account rooms that are partially as well as fully cleaned.

Here is the evaluation function:

$$\frac{\mathrm{n} \cdot \mathrm{A} - \mathrm{remainingCleanSteps}}{\log(\mathrm{totalActionCost})},$$

where,

- n = number of room in the house (excluding dump room);
- $remainingCleanSteps$ = remaining cleaning actions required to clean the whole house;
- $totalActionCost$ = cost of all actions performed so far.

This measure estimates the overall "cleanliness" of a house opposed to the taken action cost. That way, our goal is to maximize the first, and minimize the second.

4.2. Occupancy Schedules

In order to simulate stochastic environment, we generate "schedules" for each room. Schedules define at which time steps the rooms are occupied. In addition, it shows, which occupancy intervals result in rooms becoming dirty. In order to do that we define the following probability parameters:

- Probability of continuing occupancy: $P(O(t+1) \mid O(t)) \approx 0.9$
- Probability of room becoming occupant: $P(O(t+1) \mid \neg O(t)) \approx 0.05$
- Probability of room becoming dirty at a certain occupation step: $P\big(D(t) \big| O(t)\big) \approx 0.05$

One can notice that for longer occupancy intervals the probability of a room being dirty is higher (since the probability of a room becoming dirty is constant at each time-step, when it is occupied). And this is exactly what we want to model.

4.3. Simulation Setup

For simulation purpose, our Smart Home is given a randomly generated occupancy schedule for each room throughout the day (1440 minutes with each time-step of size one minute).

Occupancy might result in room becoming dirty. We want our schedule to be categorized into Busy and Relaxed Schedules with the former having rooms occupied and dirty for longer time interval as compared to the later schedule. Therefore, we generate two schedules with the following parameters:

- Relaxed: $P(O(t+1) \mid O(t)) = 0.9;\ P(O(t+1) \mid \neg O(t)) = 0.03;\ P(D(t) \mid O(t)) = 0.05$
- Busy: $P(O(t+1) \mid O(t)) = 0.95;\ P(O(t+1) \mid \neg O(t)) = 0.05;\ P(D(t) \mid O(t)) = 0.1$

In addition, we categorize VC functionality as follows:

- **System with Restrictions**: Vacuum cleaner cannot enter any occupied room.
- **System without Restrictions:** Vacuum cleaner can visit occupied rooms, but cannot perform CLEAN action there.

4.4. Evaluation Results

At every time-step, we calculate the performance measure of VC+DstB multi-agent environment and VC single-agent environment.

For Busy Schedule we found out that when VC is restricted from going into occupied rooms, the average daily performance of VC+DstB is 29% better in comparison to VC (figure 2). When the VC is not restricted to go inside a room occupied by inhabitants, then the average daily performance of VC+DstB is 22.6% better than that of VC (figure 3).

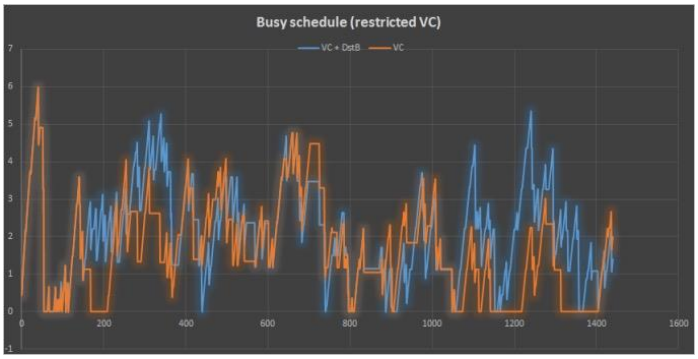

Figure 2. Busy schedule, restricted VC

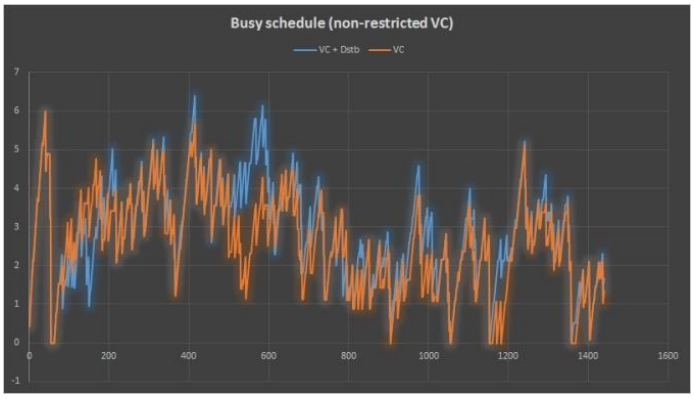

Figure 3. Busy schedule, non-restricted VC

For Relaxed Schedule we found out that when VC is restricted from going into occupied rooms, the average daily performance of VC+DstB is 10% better in comparison to VC (figure 4). When the VC is not restricted enter a room occupied by inhabitants then the average daily performance of VC+DstB is just 1% better than that of VC (figure 5).

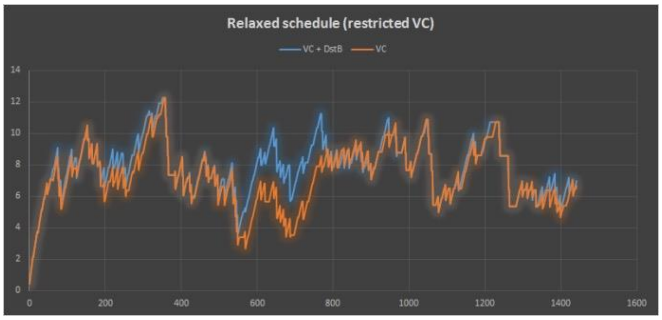

Figure 4. Relaxed schedule, non-restricted VC

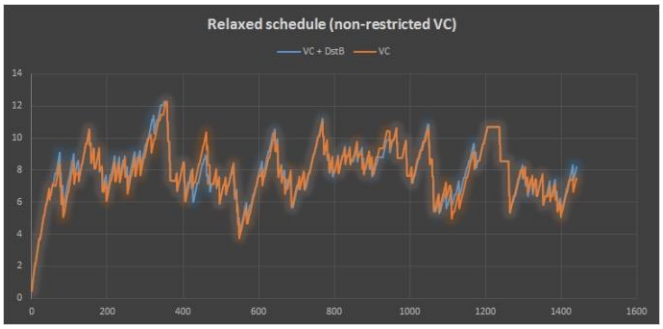

Figure 5. Relaxed schedule, non-restricted VC

5. Future Work

Since, we have made certain assumptions for our system experimentation, namely, equal room size, and equal action timing. We would consider enhancing our system configuration by relaxing these assumptions. Our approach for Dustbin might be considered as naive. Note that Dustbin MOVE action cost affects the overall performance of the whole system. Thus, we plan to adapt possibly more intelligent strategies to improve Dustbin MOVE action cost. One such approach is having DstB maintain a radial distance r with VC such that whenever VC moves outside the circle of radius r our dustbin will take MOVE actions to be in constant proximity of VC, and thus reducing the overall steps needed to be taken.

We would consider other heuristics and algorithms to increase the performance of VC. In addition, we plan to increase the scale of our implementation by considering organization level environment with a number of vacuum cleaner and dustbins collaborating with each other and having the capability of moving between floors via elevators.

6. Conclusion

These results of our futuristic multi-agent house cleaning environment show that collaborative work done by Vacuum Cleaner and Dustbin would surely make house-cleaning experience autonomous given the fact that companies implement load transfer for these two smart agents.

References

[1] X. Sun, S. Koenig, and W. Yeoh, "Generalized adaptive A*," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 469-476, 2008.

[2] S. Koenig, et al., "Lifelong planning A*" *Artificial Intell. J.*, vol. 155, pp.93 -146, 2004.

[3] F. Vaussard, J. Fink, V. Bauwens, P. Retornaz, D. Hamel, P. Dillenbourg and F. Mondada "Lessons learned from robotic vacuum cleaners entering the home ecosystem", *Robotics Auton. Syst.*, vol. 62, no. 3, pp.376 -391, 2014.

[4]"Vacuum-bot attacks sleeping women." CBS news. 10 February 2015. http://www.cbsnews.com/news/south-korea-woman-hair-stuck-in-robotic-vacuum-cleaner.

[5] "Vacuum-Cleaner robot for Organizations" http://intellibotrobotics.com/how/, http://avidbots.com.

[6] Burhams, D., and Michael Kandefer. "Dustbot: Bringing Vacuum-Cleaner Agent to Life." *Accessible Hands-on Artificial Intelligence and Robotics Education*: 22-24, 2004.

[7]"IIT Gandhinagar launches 'Talking' Dustbin" Ahmedabad Mirror. November 8, 2015 http://www.ahmedabadmirror.com/ahmedabad/education/IITGN-launches-talking-dustbin/articleshow/49716620.cms.

[8] "To promote cleanliness, 'smart dust bin' developed by youth from Rampur", UP. 29, 2015. http://social.yourstory.com/2015/05/smart-dust-bin-cleanliness/

[9] "Dustbot a robot that can collect garbage from homes" https://en.wikipedia.org/wiki/Dustbot

[10] "Various Domestic Robots based on their functionalitv" https://en.wikipedia.org/wiki/Comparison_of_domestic_robots