Automated Trash Collection using Markov Decision Processes

Robert J. Moss
Stanford University, Computer Science
Stanford, CA, 94305
mossr@cs.stanford.edu

***Abstract*—Municipal solid waste collection systems normally collect trash on a prescribed schedule and have the potential to unnecessarily collect exhaustively. The problem of reducing the frequency of garbage truck collection visits can be modeled as a Markov decision process (MDP) and solved efficiently. We split this problem into two phases: resource allocation of garbage trucks and optimized route planning given trash locations. The resource allocation phase is solved using the A* path finding algorithm to determine if alternative agents are closer to trash locations. The route planning phase is formulated as an MDP and solved in a dynamic environment using model-based and model-free approaches. For problems with relatively small state-spaces, value iteration (model-based) guarantees that the optimal policy is found. We compare the results of trash collection policies for an oracle and a baseline against a learned value iteration policy. We also analyze the benefit of multi-agent allocation in a larger city context. Results show that an optimized collection policy can reduce the usage of garbage trucks compared to a standard baseline exhaustive collection plan and therefore limit unnecessary emissions output.**

I. Introduction

Garbage trucks get on average 4.63 miles per gallon [1], thus are in a position to reduce their frequency of use. Normally, garbage trucks are deployed daily/weekly/bi-weekly to collect trash, depending on the town or city [2]. The simple fixed schedule means the usage of these inefficient trucks could potentially be reduced. This work attempts to show that this frequency can be reduced using reinforcement learning to find an optimal plan for allocating garbage trucks in a dynamic town and city environment.

The problem of optimizing the paths for garbage trucks to travel to sites and collect trash can be modeled in various ways. We split the problem into two phases: resource allocation and planning. For phase one, resource allocation, we determine if multiple agents would reduce the overall travel of a single agent to trash locations within a grid city. We use the A* path finding algorithm [3] to determine the shortest path to all trash locations from specific origin locations for up to four agents, and will deploy other agents if they are closer to the trash locations. For phase two, planning, we formulated the problem as a Markov decision process (MDP) and solved it using value iteration, Q-learning, and Sarsa(λ) to create optimal policies on the fly [4]. The policies represent the plan to follow to optimally collect trash within a city.

Our aim is to show that optimizing the allocation and planning of garbage truck deployment would help minimize the frequency of scheduled trash pickups. This minimization in turn could reduce unnecessary trash collection visits, could save cost on gas for the garbage trucks, and ultimately reduce the use of these large vehicles. This would also help to limit the emissions output from garbage trucks. Empirically, garbage trucks are loud and disruptive, causing disturbances in neighborhoods on a weekly basis. These trucks are also extremely inefficient [1]. Nevertheless, trash collection is a necessity in society, but the impact of the inefficient trucks has the potential to be reduced. Garbage trucks produce 2.23 kg of CO_2 per mile on average [1]. With the preservation of limited natural resources and reduction of carbon emissions output in mind, we purpose a reformulation of the trash collection problem to solve it efficiently.

Similar decision support systems to solve the municipal trash collection problem have been proposed. Specifically, L. Santos et al. proposed a multi-vehicle multi-route spatial decision support system for efficient trash collection in Portugal using a heuristic technique called path-scanning [5], [6]. A system was proposed to detect residential trash fill-level and use shortest path planning to find optimal routes for trash collection [7]. Another system used an integer programming model for locating garbage accumulation points that could be used within a decision support tool for municipal waste management systems [8]. Lastly, a study was done to characterize the trash collection problem and determine optimal collection paths that cover all residential blocks [2]. Absent from these approaches is an automated system that can incorporate the notion of reward into the decision of route planning in a dynamic environment—we purpose such a system.

While this work purposes an automated planning system, it does not assume the garbage vehicles are autonomously driven. Recent studies have shown that autonomous vehicles for solid waste collections could reduce costs for municipal governments [9]. Knowing the potential benefit of autonomy, the design of the purposed system abstracts the planning to the vehicle location level, therefore could be applied to either manned or unmanned vehicles.

The paper is structured as follows: Section II details the technical approach to solve the problem; describing the simulation environment in Section II-A, the model in Section II-B, the investigated algorithms in Section II-C, the software implementation in Section II-D, and addresses assumptions in Section II-E. Section III highlights the results and analysis from this work; defining an oracle and baseline

Code available at https://github.com/mossr/TrashMDP.jl

in Section III-A, analyzing different reward models in Section III-B, analyzing multi-agent allocation in Section III-C, and discussing the emissions impact of the system in Section III-D. Finally, Section IV addresses the applicability of such a system in the real-world and provides concluding remarks.

II. Approach

A. Environment

A simulation environment was built to test, validate, and tune the system. The environment is modeled as a grid with certain cells marked as roads and other cells marked as trash site locations. Two versions of the environment were tested: a 10×10 "town" with 4 trash locations (seen in Figure 1) and a 19×19 "city" with 36 trash locations (seen in Figure 4). The environment propagates forward every time step and models the accumulation of trash at the fixed trash locations. These locations each have a fixed (x, y) coordinate, a trash fill-level percentage $[0 - 100]$, and a trash fill-rate $[1 - 10]$. The fill-rate is fixed per location, while the fill-level is increased per time step based on the fill-rate. We set the simulation clock, i.e. time steps, to be a single day. We run a single environment step, solve the MDP to get an optimal policy on the fly, and then execute that policy. The full simulation is run over 364 simulated days or 52 simulated weeks. We set a *collection threshold* of 80% to be the fill-level in which the trash is deemed ready to be collected.

B. Model

Due to the cost/reward nature of this problem, a state-based approach was a good candidate, more specifically, modeling the problem as an MDP. Our approach was to formulate the garbage truck planning problem as an MDP and solve it efficiently with a variety of algorithms within our simulation environment. The agents in this problem are garbage trucks and will be referred to as the agents going forward. Multiple agents can be deployed during the resource allocation phase.

1) State-Space: The state is given by the (x, y) location of the agent in the fixed grid world. The state-space is strictly limited to the roads and the trash locations as shown in Figure 1. Thus, the state-space dimensionality is only 48 states for the 10×10 grid and 316 for the 19×19 grid (see Figure 4).

This MDP has no terminal state as the trash can continue to accumulate at the locations and the agent can continue to collect the trash. We terminal the simulation after a fixed maximum time step (set to 364 days, or 52 weeks).

2) Action-Space: Modeling this problem in a grid world means we are only worried about the actions that the agent can take in a 2D Cartesian plane. The action-space is the set of all actions {`nothing`, `up`, `down`, `left`, `right`}, where `nothing` is a non-action for the agent to remain stationary and wait for more information to become available through the propagation of the environment.

3) Reward Function: Multiple reward functions have been explored and tested in this approach. The two reward functions shown in Figure 2 are piecewise functions R of the fill-level L. This allows us to model reward as a function of the trash accumulation. Both reward functions have identical

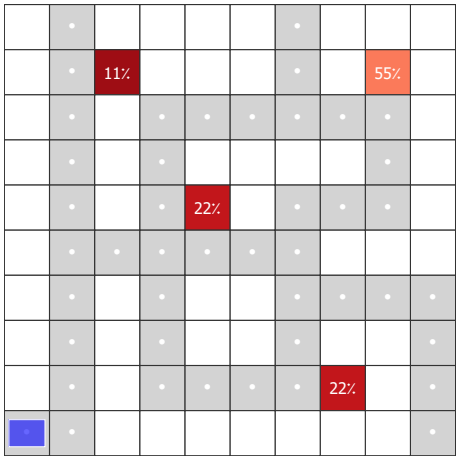


Fig. 1: Gray cells represent roads, red cells represent the trash accumulation locations (where the color indicates the fill-level at that location), and the blue rectangle is the agent. White dots indicate the current optimal action at that cell (where the dot is the `nothing` action).

behavior below the *collection threshold*, where the agent would be penalized for visiting sites with little to no trash. The *constant* reward function outputs constant reward when above the *collection threshold* to help the agent collect all nearest trash locations—further explored in Section III. The *decreasing* reward function outputs decaying reward as a function of the fill-level relative to full to penalize allowing locations to get close to the maximum fill-level F. All rewards are adjusted by a multiplier of 100 for more resolution in relative comparisons explored further in Section III. Note that a discount factor of $\gamma = 0.99$ was selected for this MDP formulation.

$$R(L) = 100 \begin{cases} \sqrt{\frac{L}{F}} - 0.9 & \text{if } L < \text{threshold} \\ \sqrt{\frac{F-L}{F}} & \text{if } \textit{decreasing} \\ 1 & \text{if } \textit{constant} \end{cases}$$

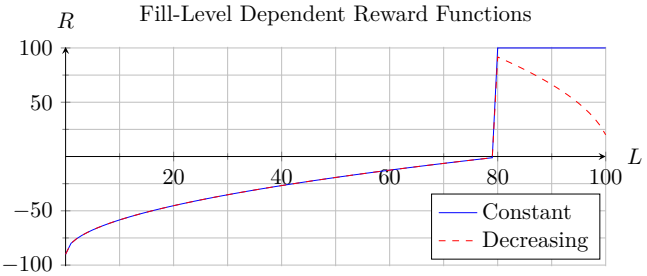
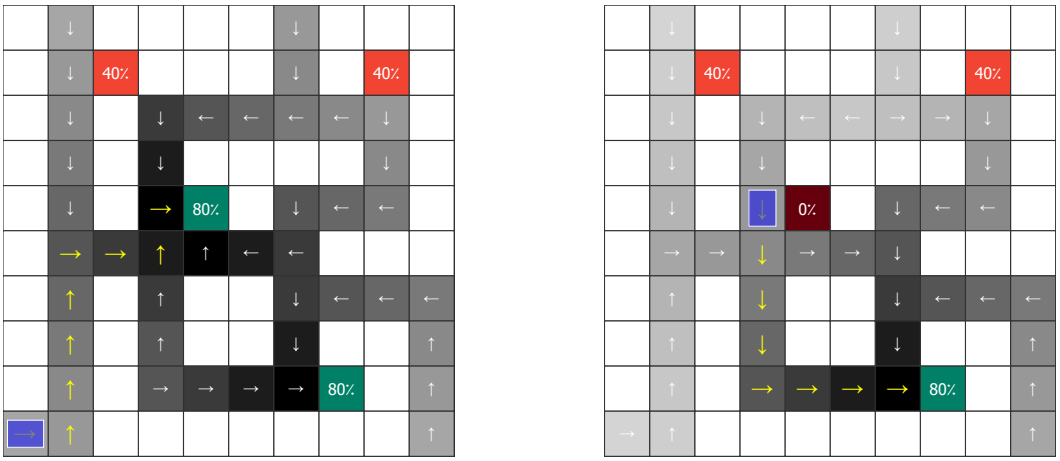


Fig. 2: Piecewise *constant* and *decreasing* reward functions based on fill-level with an 80% collection threshold.

4) Transition Function: The transition function is deterministic as the agent can move to another road cell or location cell deterministically. The transition function also handles the case where the action is `nothing` and will transition to the current state with probability 1.



(a) Collecting trash from 1st site location

(b) Collecting trash from 2nd site location

Fig. 3: Example policy plots for a single day. Arrows indicate the optimal action in that state, where yellow arrows highlight the travel path. The color scale of the roads indicate the Q-values from the policy, showing the gradient that the agent follows. The green cells indicate trash locations above the *collection threshold*, thus deeming them ready to be collected.

C. Algorithms

For the first phase, resource allocation of agents, we solve the problem using the A* path finding algorithm [3]. We use A* to determine the shortest paths to all trash locations and split those locations into partitions assigned to specific agents. We limit the maximum number of agents to four, and each agent has a unique origin point within the grid city.

For the second phase, efficient route planning, several algorithms were investigated to solve for the optimal policy. Notably, value iteration, Q-learning, and Sarsa(λ). Value iteration is an appropriate choice because we are using a *model-based* approach with full knowledge of the entire state-space, where the state-space is relatively small. Therefore, we can learn the transition function T and reward reward function R directly. With a Bellman residual of $1e-6$, which is used as a threshold for Q-value convergence, value iteration converges in about 160 iterations. Therefore, the maximum iterations was set to 200 to provide a bound. Value iteration is guaranteed to converge to the optimal policy if all states can be explored [4]. Figure 3 shows example policies learned from value iteration.

We implemented a Q-learning approach to solve for an optimal policy. Being a *model-free* approach, we do not learn the transitions and rewards directly, but build up an estimate as the simulation runs. We chose this to compare the model-based and model-free approaches—also the implementation is relatively fast, specifically due to the small state space. Parameters for Q-learning include a learning rate $\alpha = 0.1$, a total of 200 episodes, and the frequency to evaluate the trained policy at every 50 episodes (to reduce computation time). We chose not to implement an exploration strategy like epsilon greedy because we want to agent to deterministically take the optimal action leading to trash site locations.

Choosing Sarsa(λ) as another model-free approach was specifically due to the feature of eligibility traces that comes with this algorithm. Because we've included a `nothing` action, and Q-learning and value iteration can be "shallow" in their spreading of utility to neighboring states, we wanted to test out an algorithm that back-tracks utility in a single iteration.

Given the relatively small state-space, all three algorithms converge to the identical optimal policy, with value iteration running 15 times faster than Q-learning. Thus, further results will focus solely on value iteration as it finds the optimal policy efficiently.

D. Implementation

The implementation of the MDP is done in the Julia programming language [10] leveraging the POMDPs.jl package [11]. Julia is a well-equipped language for such a problem as it is close to C in speed and as expressive as Python.

The design of the system models a grid world where the agent can move to neighboring cells in search for a reward. The major deviations from a standard grid world are the restrictions the agents has in the environment, namely, requiring the agent to move only on the roads. Normally, grid world problems have a fixed reward that an agent has to find, where the implementation of this system leverages the speed of Julia to propagate the environment one time step, solve the MDP using value iteration, follow the optimal path output by the policy, resolve the policy to determine if more locations are ready to be picked up, and then collect reward metrics. All of that in a single iteration of the "real-time" system. Because the implementation relies on the dynamic rewards in the environment, which are functions of the current fill-levels at each location, learning policies during run-time was the most appropriate approach.

E. Assumptions

There are a few assumptions being made and we will address them in this section. The first assumption is that the

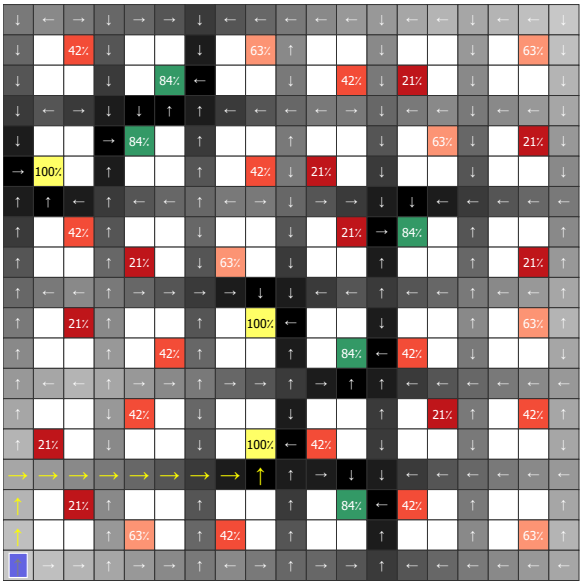


Fig. 4: A single agent in the 19×19 grid city with 36 trash locations and overflowed accumulated trash shown in yellow.

fill-level at each location is reported to the agent. We are assuming that each location is part of a network of sensors that indicate, without error, the actual fill-level of trash accumulation at that particular location. This assumption is not paramount as cities are introducing smart waste bins more frequently [7]. Another assumption we are making is that the trash truck has unlimited fill capacity. This means the truck can stay out and continue to collect trash during a single day (i.e. time step) without having to deposit the garbage into some central dump. What may seem obvious, but we also assume that each location is adjacent to a road so that the garbage truck can reach it. Lastly, an assumption that we want to address is the maximum fill-level. Once a trash site location hits the maximum fill-level, they will stay at that level. The reward is still positive, but the agent prioritizes collecting trash closer to the *collection threshold* before it gets too full.

III. Results and Analysis

This section will detail the analysis of the results from various experiments. Analysis of oracle and baseline policies versus the value iteration policy will be explored. This section also covers experiments of sweeping scheduled collection frequencies and their resulting metrics. Metrics include mean reward, overflow count (the number of times trash was collected when the fill-level was "full"), and mean path length in miles. Experiments with multi-agent allocation will also be analyzed. The following results were collected on the 19×19 grid city that has 36 trash site locations, as seen in Figure 4.

A. Oracle and Baseline

Oracle and baseline algorithms were implemented to provide a bounded comparison to the value iteration policy. The chosen baseline algorithm will collect all trash locations at a fixed frequency in units of days. This models how standard trash collection in cities and towns operates—collecting everywhere on a fixed schedule. We use the A* path finding algorithm to iteratively determine the closest path between the current trash locations. This gives us an accurate measure of the total path length when the baseline travels to every location. The A* algorithm uses the Manhattan distance as a heuristic, which is helpful given the problem is modeled as a grid city with road restrictions. The heuristic relaxes the problem and allows for the search to bypass the roads.

The chosen oracle algorithm will collect trash from all locations when they are deemed to be ready. The oracle is not deployed on a fixed schedule, rather is dynamic to the needs of the city. The oracle uses Euclidean distance from each current trash site location to get a straight path to locations.

B. Reward Model Analysis

The *constant* and *decreasing* reward models shown in Figure 2 are compared using the oracle, baseline, and value iteration policies in the next sections. Analyzed metrics are the mean reward, overflowed count, and mean path length. The results are from the 19×19 city, run for 364 simulation days or 52 weeks.

1) Mean Reward Analysis: Figure 5 highlights the mean rewards when sweeping frequency of collection for a single agent in the 19×19 city. Notice that the baseline results show that given the policy of collecting *every* location at the given frequency, the large negative mean reward is expected. Looking at the bottom figure, we compare the value iteration policy against the oracle in more detail (omitting the baseline). Focusing first on the *constant* reward model given by the solid lines, we see that the value iteration policy hovers around the mean rewards of the oracle below about

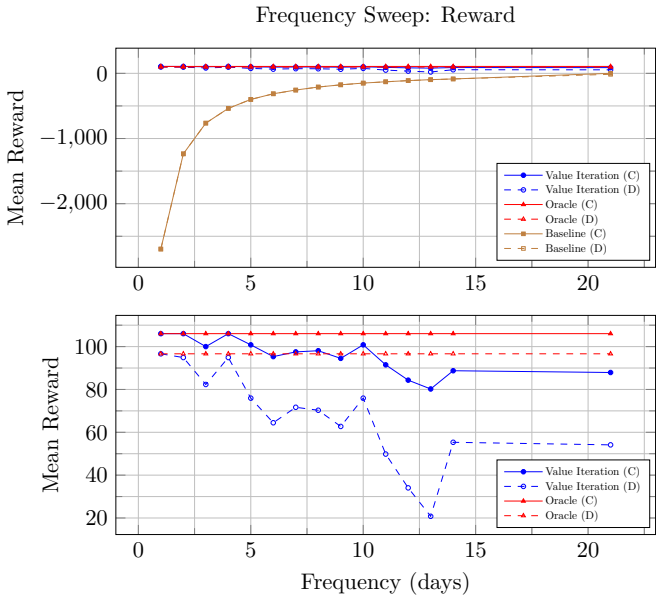


Fig. 5: Mean reward when sweeping frequency of collection for the 19×19 city. Bottom figure omits the baseline results. Solid lines represent the *constant* reward model and dashed lines represent the *decreasing* reward model.

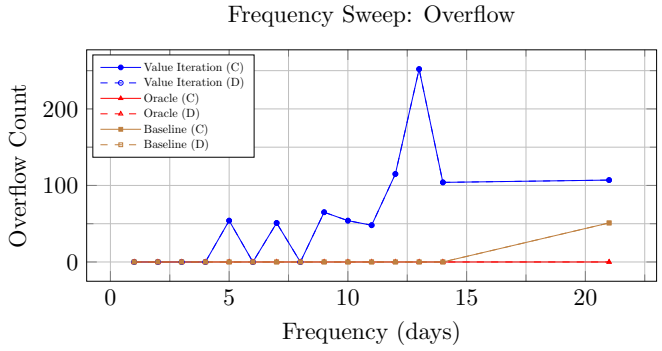


Fig. 6: Overflow count when sweeping frequency of collection for the 19×19 city. Solid lines represent the *constant* reward model and dashed lines represent the *decreasing* reward model.

a frequency of 10 days. When the frequency is set to two or four days, notice that the value iteration policy gathers higher mean rewards compared to the oracle. This can be characterized as behavior resulting from clearing out trash locations quickly, and allowing them to fill up once again. Thus, collecting more constant reward.

Now focusing on the *decreasing* reward model in the dashed lines, we can see that the value iteration policy never gathers more reward than the oracle. This is due to the design of the oracle—collecting trash at the exact day they're ready, i.e. met the *collection threshold*. Because collecting at the *collection threshold* gives the agent the maximum reward, the oracle always achieves the maximum reward. Because the value iteration policy waits until the scheduled frequency has arrived, it could allow for trash to accumulate past the *collection threshold*, thus gathering less overall reward, yet still positive.

Notice that when the frequency is one, the value iteration and oracle policies produce the same mean rewards. Thus is because the value iteration algorithm will determine which locations are ready to be picked up then solve for the optimal policy. Therefore, when the frequency is set to one day, the value iteration algorithm will sample the current locations daily to see if any are ready for collection, thus collecting the trash when the reward is the highest. Also, notice for the *constant* reward model when the frequency is two or four, value iteration achieves the same mean reward as the oracle. This can be explained based on the selected fill-rates of this particular city, where they happen to land within the *collection threshold* and the maximum fill-level during the collection cycle. The reward is constant regardless of the fill-level between the *collection threshold* and the maximum fill-level. This helps explain why the *decreasing* reward model does not quite achieve the same maximum reward as the oracle for frequencies of two and four, but is fairly close.

2) Overflow Analysis: When analyzing how often trash was collected when the fill-level was at its maximum, i.e. overflowed, we found some surprising results with the value iteration policy. In Figure 6, focus on when the frequency is set to days between 4–9. This sawtooth behavior is counter-intuitive. Initially, we thought that increasing the collection frequency would therefore increase how often overflowed trash was recorded. But, that is not the case. Consider the frequencies of five and six days. Based on the 36 locations all with a particular fill-rate, there can be cases where collecting trash at a lower frequency, say every five days, can result in other sites being just about ready to be picked up. Thus, waiting another five days, those sites would be overflowed by that point. Yet, if we set the frequency to six days, this allows sites close to the *collection threshold* to reach it, thus collected during that collection cycle.

Other interesting results from the overflow analysis are the similarities between the *constant* and *decreasing* reward models. There are no notable differences between the *constant* reward model and *decreasing* reward model for the value iteration policy. This result is expected as changes to the values of rewards above the *collection threshold* should not affect the amount of overflowed collection visits. Notice that the oracle never collects overflowed trash, per its design.

Notice that the baseline only collects overflowed trash when the collection frequency is set to 21 days or 3 weeks. This is based on the fill-rates of the 19×19 city trash site locations. Given a large window of time between collection visits, there are bound to be locations that reach their maximum in that 3 week span, even though the baseline collected all trash during the prior collection cycle.

Lastly, notice the spike in the overflowed count for the value iteration policy at a frequency of 13 days. This is a product of the selected fill-rates for this city. The explanation is similar to the sawtooth behavior around frequencies 4–9.

3) Mean Path Length Analysis: An important metric when trying to reduce the usage of garbage trucks and optimize their frequency is the mean path length of a given collection deployment. Given that each cell in the grid city represents a square mile, the units of the mean path length is in miles. Note, after an agent has collected all trash locations, we send the agent back to its origin using A* and count that path length as well.

First, looking at the top plot in Figure 7, which includes the baseline, we can see that since the baseline visits every site during its collection cycle, the path length is extremely high when the collection frequency is low, and gradually reduces based on the spread out collection frequency. Notice that the oracle and value iteration policies always achieve a smaller mean path length, regardless of collection frequency.

The bottom plot in Figure 7 omits the baseline results to focus on the comparison between the oracle and value iteration policies. Observe that the *constant* reward model is always at or below the *decreasing* reward model. This result is expected because the design of the *decreasing* reward model emphasizes collecting trash at sites closer to the *collection threshold* first. Thus, leaving the overflowed trash for the end of its collection route, therefore potentially going past locations in the process of collecting others. When the reward model is constant, the agent is deployed to collect

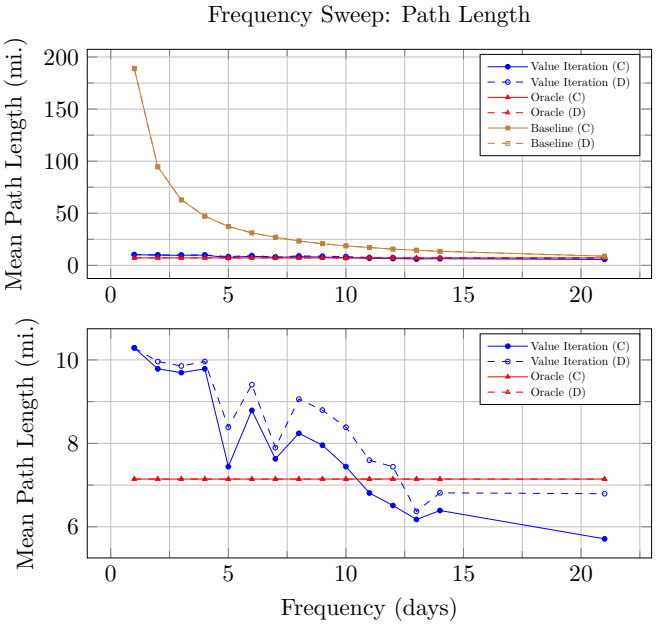


Fig. 7: Mean path length in miles when sweeping frequency of collection for the 19×19 city. Solid lines represent the *constant* reward model and dashed lines represent the *decreasing* reward model.

trash and will collect the closest trash site locations, in that order. This is highlighted in the fact that the *constant* reward model is always at or below the *decreasing* reward model.

As a reminder, we used Euclidean distance to measure the path length for the oracle. What we observe in Figure 7 is that the value iteration policy can achieve a shorter mean path length above a collection frequency of 11 and 13 days, for the *constant* and *decreasing* reward models respectively. Yet, referencing back to the mean rewards in Figure 5 and overflowed count in Figure 6, we can see selecting a collection frequency above about 10 days results in a degradation of the reward and overflowed metrics. Based on these results, we will continue analysis solely with the *constant* reward model.

C. Multi-Agent Analysis

The first phase of the system is to determine if other agents could also be deployed to help collect the available trash. We use the A* path finding algorithm to determine if trash locations are closer to specific agents at each corner of the city. We limit the number of agents to be between 1–4, with each agent occupying a specific corner. See Figure 9, which shows the same time step as Figure 4 but with four agents occupying partitions of locations within the grid city.

Figure 8 plots the mean path length for the value iteration policy over the 19×19 city for 1–4 agents, sweeping collection frequency. Intuitively, when a total of four agents are selected, the overall path length is minimized—except in the case for the collection frequency of 21 days. Notice that the single agent value iteration policy slightly outperforms the four agent policy during this 21 day collection frequency.

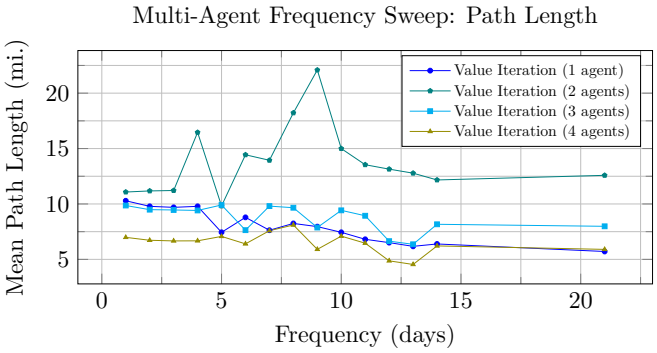


Fig. 8: Mean path length in miles for multi-agent allocation when sweeping frequency of collection for the 19×19 city.

This is due to many of the trash locations being ready to be collected, therefore, each agent has to travel to almost all of the locations. When it's just a single agent, the agent traverses the entire city optimally. With four agents, they focus on their city partitions.

D. Emissions Analysis

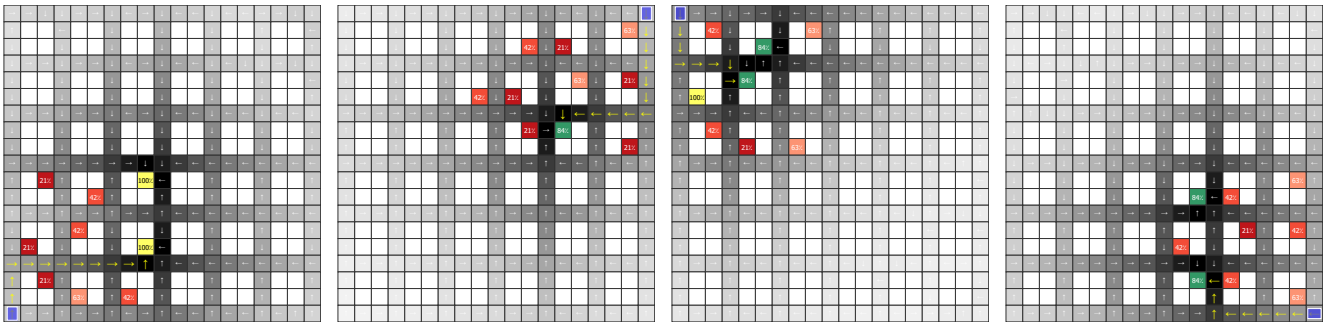
The aim of this work is to show that an automated trash collection system can be optimized to reduce frequency of visits, thus reducing the total carbon emissions output by the garbage trucks and save cost on gas per day. Using emissions data and miles per gallon (MPG) data for garbage trucks from [1], we have compared the value iteration policy using the collection frequency of eight days against the baseline and oracle policies. Based on the results from Section III-A, we chose to analyze a collection frequency of eight days. The choice of eight days weights out maximizing rewards, but more importantly, minimizing path length and is the highest frequency with an overflow of zero.

The emissions output results and cost of gas per day are highlighted in Table I. As expected, the oracle is ideal with respect to emissions output and gas cost. Focus on the value

TABLE I: Average emissions output and gas cost.

Num. Agents	Algorithm*	Emissions (CO_2 kg/day)†	Gas Cost (USD/day)‡§
1	Oracle	15.95	$6.02
	Baseline	52.18	$19.69
	Value iteration	18.41	$6.94
2	Oracle	16.50	$6.22
	Baseline	44.73	$16.87
	Value iteration	40.69	$15.35
3	Oracle	12.40	$4.68
	Baseline	43.07	$16.25
	Value iteration	21.58	$8.14
4	Oracle	7.85	$2.96
	Baseline	38.03	$14.39
	Value iteration	18.10	$6.83

* Collection frequency of every eight days.
† Average of 2.23 kg of CO_2 per mile for garbage trucks [1]
‡ Average of 4.63 miles per gallon for garbage trucks [1]
§ Diesel price in California of $3.904 per gallon [12]



(a) Agent 1 (b) Agent 2 (c) Agent 3 (d) Agent 4

Fig. 9: Multi-agent allocation given starting origins at the four corners of the 19×19 grid city.

iteration and baseline results. The value iteration always outperforms the baseline policy for all number of agents allocated. Notably, a single agent value iteration policy has a 64.7% reduction in daily emissions and gas cost compared to the baseline. A two agent value iteration policy has a 9.1% reduction, a three agent policy has a 49.9% reduction, and an four agent policy has a 52.4% reduction. From these results, we can see that an automated trash collection system, formulated as an MDP, can save money and reduce carbon emissions.

IV. Conclusion

An automated trash collection system can be widely applicable in a real-world setting. Certain municipalities have similar systems in place to plan routes [2], [5]. Expanding on this work could turn the system into a decision support tool for city waste management departments. Converting the grid world formulation to integrated real-world maps would be trivial as the problem is modeled as a graph between nodes. Interfacing with smart garbage bins in cities could provide the measurement of the fill-level at specific locations. Based on the static nature of trash collection schedules, a more dynamic/reactive approach could benefit towns and cities—both monetarily and environmentally.

In this paper we proposed a trash collection system in order to reduce the usage of garbage trucks. We split the problem into two phases: resource allocation of garbage trucks and optimal route planning. We described the formulation of the problem as a search problem using A* and an MDP to represent the state, actions, and rewards in the city environment. The MDP was solved using the value iteration algorithm to produce the optimal policy efficiently. The work also produced a method to tune the frequency of scheduled trash collection visits. Comparing against a naive baseline system and an oracle system showed that a value iteration policy has major benefits in both emissions output and cost of gas on a per day basis.

Acknowledgment

We would like to thank the Stanford Intelligent Systems Laboratory for their development of the following Julia packages: POMDPs.jl, DiscreteValueIteration.jl, and TabularTDLearning.jl. We would also like to thank Mykel Kochenderfer and Shushman Choudhury for their insight.

References

[1] Gurdas Sandhu, Henry Frey, Shannon Bartelt-Hunt, and Elizabeth Jones. In-use activity, fuel use, and emissions of heavy duty diesel roll-off refuse trucks. *Journal of the Air and Waste Management Association*, 65:306–323, 02 2015.

[2] Shan-Huen Huang and Pei-Chun Lin. Vehicle routing–scheduling for municipal waste collection system under the "keep trash off the ground" policy. *Omega*, 55:24 – 37, 2015.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[4] M. J. Kochenderfer, C. Amato, G. Chowdhary, J. P. How, H. J. D. Reynolds, J. R. Thornton, P. A. Torres-Carrasquillo, N. K. Ure, and J. Vian. *Decision Making Under Uncertainty*. MIT Press, 2015.

[5] Luís Santos, João Coutinho-Rodrigues, and John R. Current. Implementing a multi-vehicle multi-route spatial decision support system for efficient trash collection in portugal. *Transportation Research Part A: Policy and Practice*, 42(6):922 – 934, 2008.

[6] B.L. Golden, J.S. Dearmon, and E.K. Baker. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10(1):47 – 59, 1983.

[7] S. Dugdhe, P. Shelar, S. Jire, and A. Apte. Efficient waste collection system. In *2016 International Conference on Internet of Things and Applications (IOTA)*, pages 143–147, Jan 2016.

[8] Diego Gabriel Rossit, Sergio Nesmachnow, and Jamal Toutouh. A bi-objective integer programming model for locating garbage accumulation points: a case study. *Revista Facultad de Ingeniería Universidad de Antioquia*, (93):70–81, 2019.

[9] Benjamin Y Clark. The impacts of autonomous vehicles on local government budgeting and finance: Case of solid waste collection. *Available at SSRN 3471853*, 2019.

[10] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral Shah. Julia: A fresh approach to numerical computing. 07 2015.

[11] Maxim Egorov, Zachary N. Sunberg, Edward Balaban, Tim A. Wheeler, Jayesh K. Gupta, and Mykel J. Kochenderfer. POMDPs.jl: A framework for sequential decision making under uncertainty. *Journal of Machine Learning Research*, 18(26):1–5, 2017.

[12] Weekly retail gasoline and diesel prices. https://www.eia.gov/dnav/pet/pet_pri_gnd_dcus_sca_w.htm. Accessed: 12/13/2019.