Dynamic Allocation of Emergency Resources

CS221 Final Report

Tyler Romero (tromero1@stanford.edu)
Zachary Barnes (zbarnes@stanford.edu)
Frank Cipollone (fcipollo@stanford.edu)

***Abstract*—By dynamically allocating resources such as personnel and vehicles in emergency situations, we can provide a solution that minimizes damage and loss of life while also conserving taxpayer dollars.**

Keywords—public safety; emergency events; reinforcement; MDP; resource allocation; time series

I. Motivation

Each year, emergency responders assist in millions of critical events across the country. The size of an emergency response can scale from a single paramedic to multi-state crews fighting large fires. As emergency events, the time it takes first responders to arrive on scene is critical, with minutes often making the difference between life and death. Because of these factors, standard staffing and resource use is very high to make sure enough responders are available at any given time. These factors make emergency response an important potential application for optimizations. Thus, a model that makes dynamic decisions on allocation of resources would be extremely useful to government and department management.

II. Approach

A. Goals and Scope

Our goal is to use historic per day emergency incidents in a specific geographic region as an input for our model. We will frame this application as a reinforcement learning problem where training examples will be days of historic data drawn from the pool of emergency events for the region as well as potential relevant weather, geographic, structural, and demographic features that may be added if our simple data source is found lacking. This will allow our model to make an optimal allocation of resources over our region of interest at any given time.

The particular region of interest that will be explored is the city of San Diego. Specifically, we will be allocating fire trucks for the San Diego Fire Department. Our model is time sensitive and will be evaluated at time steps covering a 24 hour time period. Our model will be fed a transcript of incidents over a day as its input (fed at the corresponding time step), and will output the locations that the fire trucks have moved to at each time step. We will measure model performance by simulating our model over a test set of randomly selected days, and returning the average response time per incident.

A concrete example input for a timestep would be the following:

[FIRE, Vehicle Fire Freeway,
2016-08-18T23:51:56, (32.839, -117.144)]

And the output for the timestep would be a dictionary with a fire truck number as a key and a location destination (for the following timestep) as an output. E.g. for one timestep with 2 trucks:

{truck1: (32.839, -117.144), truck2: (32.822,
-116.909)}

This input, output behavior mirrors that of a real life decision-maker where incidents are reported and directives are given. Our different model implementations will be compared via average squared response time. This is an important real world metric, that is easily calculated using our model. We used squared response time in particular,

because, as emergency events, the longer until an event is left untended, the worse it may become.

B. Data Sources

Our source of historic emergency incidents is the San Diego Open Data Portal which provides every fire incident responded to in the last year. This dataset is comprised of the type, location, date and time, and category of severity for approximately 150,000 incidents [2]. Historic fire data will possibly be supplemented with weather [3], geographic [4], and demographic data [5] corresponding to the region of interest if needed.

C. Baseline and Oracle

A perfect approach, or oracle, would result from having all of the important knowledge of all future emergency incidents. Thus, this oracle would route trucks so that for any incident a truck is already present and available. A baseline approach would result from knowing nothing about future incidents nor making active routing decisions. This approach would for any given incident simply route the closest truck to the incident and nothing more.

After implementing the two approaches, we found that (depending on initial assumptions), the baseline had a quite high total error, whereas the oracle was very close to zero. This difference is exactly what we expected. The oracle knew where the incident would be taking place and was able to route fire trucks there before the incident actually happened. Therefore, there was always a truck ready to respond exactly where it was needed and when it was needed. On the other hand, the baseline approach had the trucks blindly wander around the city until there was an incident. Even the closest truck was usually far from the incident. Our solution will lie somewhere between these extremes.

D. Our Model

Our approach to a solution is to utilize reinforcement learning to learn in the uncertainty of not knowing when and where future incidents will occur. In this sense, that motion of our trucks is deterministic based on the action we select, however, there is an implicit transition probability to one of many successor states based on the probability of an incident occurring in any given grid cell.

The first important design decision we made was regarding the best way to represent each state in which a decision is needed. We need to account for a variable number of firetrucks, as well as possible different time step granularity and different truck speed limits. These are parameters we felt should be set by the user and not hardcoded into the model. We decided to force the state to hold the following information:

1. The position of each truck and it's current assignment
2. Every active incident on the grid of San Diego. An incident will be removed from the state after a truck occupies the same grid space for a certain number of timesteps.

Then, we could simply define the choice of action as we represented it in section *goals and scope*: a new assignment for each truck. Importantly, actions are *not* a specific truck movement. At each timestep, the trucks will be automatically moved (as fast as they can go, as defined by their speed limit) towards their current goal. The only action is an update on this goal directive.

We face an important challenge in modelling this problem with a Q-learning algorithm. We note that it is not feasible to iterate through every action in any given state of our model. To show why this is, we note that the list of actions for a state with n trucks and a dxd grid will include every permutation of assignments of each truck to each location on the grid. In other words, the size of the list of actions is:

$$O((dxd)^n)$$

This quickly becomes an infeasible number of actions to iterate through. In order to solve this problem, we are forced to abandon iterating through every possible action and use a different kind of action selection.

Action Selection

We wish to choose between all possible actions, but we can effectively limit the number of actions

we have to iterate through using the following reasoning. First, we will almost never want to assign a truck to an incident that a different truck could get to more quickly. Second, we don't want any two assignments to be too close to each other; it would generally be inefficient to have two trucks in the same place. Third, if there are free trucks and active incidents, these trucks should be assigned to these incidents. These principles limit the number of actions we need to iterate through whenever incidents are present. However, we still need to dramatically reduce the number of actions we consider.

Our proposed state selection is a randomized state selection, guided by heuristics. It takes into account the above three ideas in the following ways:

1. Rather than choosing the best *n* assignments for each of *n* trucks, we will be much better off iterating through selections of *n* points, and for each point, simply assigning the closest truck to that point.
2. We use a probability distribution that makes it unlikely that many points will be placed in the same vicinity
3. Before calculating any random point, we include all active incidents in our list of *n* points (if there are more incidents than trucks, we assign each truck to the closest incident).

Now, we can simply decide how many randomized states we want to look through, allowing us to easily work with the tradeoff of performance versus running time.

Having defined our states and actions, the second major design decision we had to make was how to pick the optimal feature selector.

Feature Selector

Fortunately, it is fairly clear which features of the state are important in the calculation of how good a state is. At a low level, we want trucks to maintain balanced positions relative to one another and we want to keep trucks in areas that have been shown to be accident prone.

Concretely, we include the following features:

1. Rounded mean distances between trucks
2. Rounded mean X position of the trucks
3. Rounded mean Y position of the trucks
4. Indicators on whether or not one or more trucks are currently in a specific subsection of the grid.

These features are designed to generalize well. This is important because we will have, as we said before, an infeasible number of states and actions to learn a policy for. If we included this information, we would expect to learn nothing because there would be so many features it would be unlikely we would see the same one twice (even if we could store all this information).

Reward Function

Defining a proper reward function for this problem was non-trivial. It may seem intuitive to reward a truck for handling an incident, however, this would not lead to effective learning regarding where to place firetrucks when there are no active incidents.

The reward function we settled on takes into account the squared distance from an incident to the nearest firetruck when a new incident occurs. We multiply this value by -1 in order to incentivise our model to minimize the magnitude of this value.

Hyperparameters

Our model relies on several hyperparameters: exploration probability, discount factor, number of random actions to generate, and number of subsections to divide our grid up into (see feature #4). We experimentally determined the optimal value of each hyperparameter. Please see the plots in *Appendix A* for a clear visualization of how each hyperparameter impacts the effectiveness of our model.

Exploration probability tends to not affect the results very much. This is because the appearance

of incidents forces our model to explore anyways. We settled on an exploration probability of 0.05 for our final model, finding that it performs marginally better than its counterparts.

Number of actions to generate per state has a dramatic affect on on the performance of our model, up to a point. Note that if we were to only generate one action per state, our model would be equivalent to our baseline! We found that generating any number of actions over ten offers diminishing returns. For our final model, we generated twenty actions per state as a compromise between optimal results and efficiency.

Our model tends to be very sensitive to discount factor. Very large or small discount factors perform poorly; discounts between 0.4 and 0.6 tend to perform consistently well. We used a discount of 0.6 in our final model.

Finally, we needed to determine the optimal number of subsections to used in our fourth feature. The addition of this feature in general greatly improved the performance of our model. We found that using nine subsections tends to perform the best and the most consistently.

E. Results

When pitted against the baseline, our model offered a reasonable performance increase! Please see *Appendix B* for a chart that compares our model and a table summarizing our results.

As expected, our oracle performs extremely well. Having knowledge of when and where an event will occur offers a huge competitive advantage. No realistic model could hope to compete.

The baseline model performs moderately well. The greedy solution to this problem is not terrible.

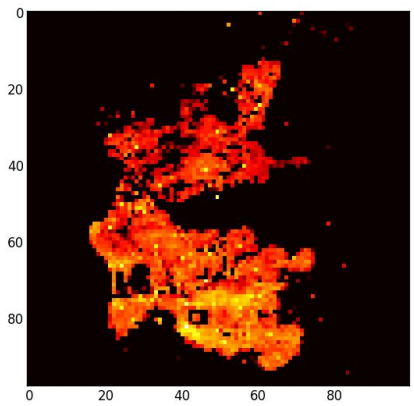
Our model beats the baseline by a substantial margin by better balancing the position of the trucks around the grid, and by favoring areas of the grid in which a large number of incidents occur.

In order to verify that learning occurs, we plotted the average squared response time that the Q-Learning model achieves on January 31st. 2009 after training it on the previous 1, 10, and 31 days. The results of this experiment can also be viewed in *Appendix C*. It is clear that training the model over a larger number of days tends to improve its performance.

F. Analysis

Our model performs well by balancing the location of the trucks around our grid representation of San Diego, and by preferring areas where a higher number of incidents occur. Our features that indicate the average position of the trucks and the presence of trucks in certain subsections of the grid help to ensure that this occurs.

After seeing that our model outperforms the baseline, we desired to visualize the underlying distribution that our model is learning on. Below is a heatmap of all of the incidents that occur in San Diego in 2009.



Because there is a well-defined underlying distribution, our model is well suited to this problem. Since the greedy baseline model gains no knowledge regarding this distribution, it cannot perform as well as our Q-Learning model.

G. Next Steps

Looking ahead, we plan to make our model more detailed. For example, adding Ambulances that can respond to non-fire emergencies, and allowing for our model to dispatch different numbers of fire trucks at different times in order to meet demand at peak times of the day.

H. Literature Review / Similar Applications

A similar application of machine learning to help emergency responders was done by Bayes Impact. They analyzed Seattle police report data in order to

determine ways in which Seattle could better deploy officers with the goal of minimizing serious and violent crime [6].

I. Source Code

Source code for our project can be found at https://github.com/tyler-romero/fire-prediction.

III. References

[1] Haynes, Hylton JG. “Fire loss in the United States during 2015.” National Fire Protection Association. Fire Analysis and Research Division, 2016.

[2] *San Diego Open Data Portal*. The City of San Diego, n.d. Web. 21 Oct. 2016.

[3] *Weather Underground - Weather Forecast & Reports*. N.p., n.d. Web. 21 Oct. 2016.

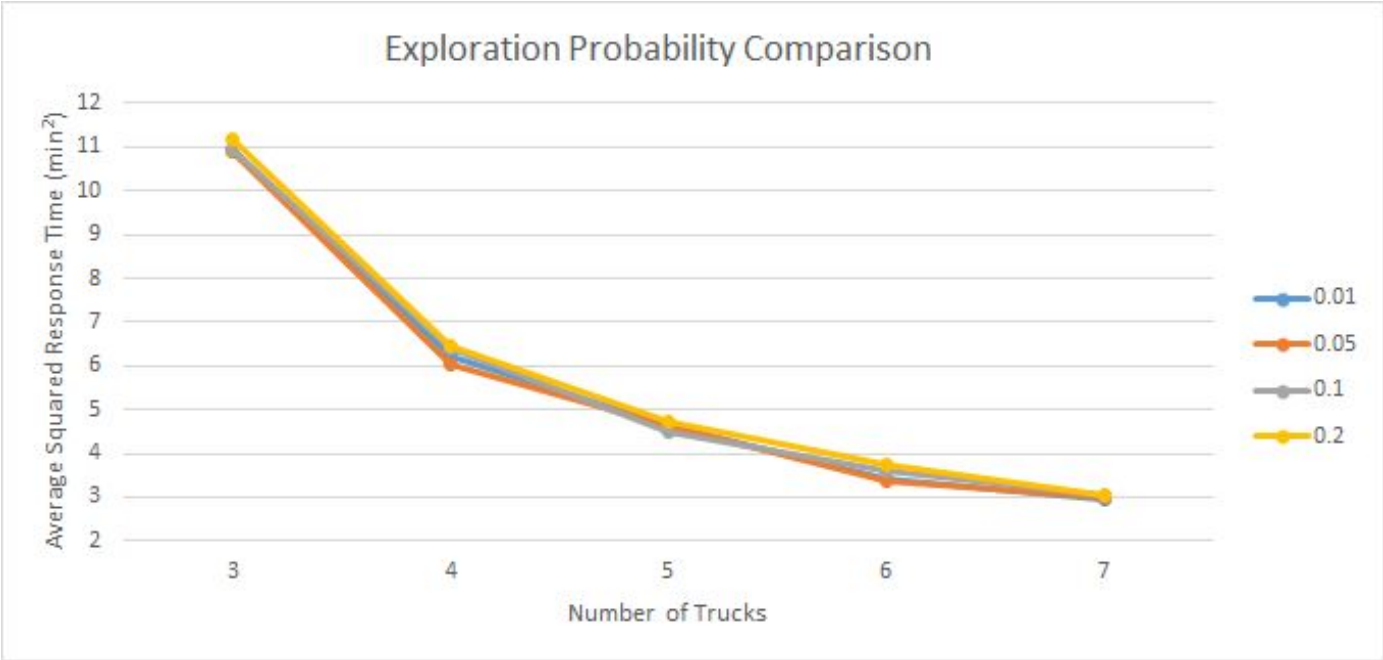
[4] *SanGIS*. San Diego Geographic Information Source, n.d. Web. 21 Oct. 2016.

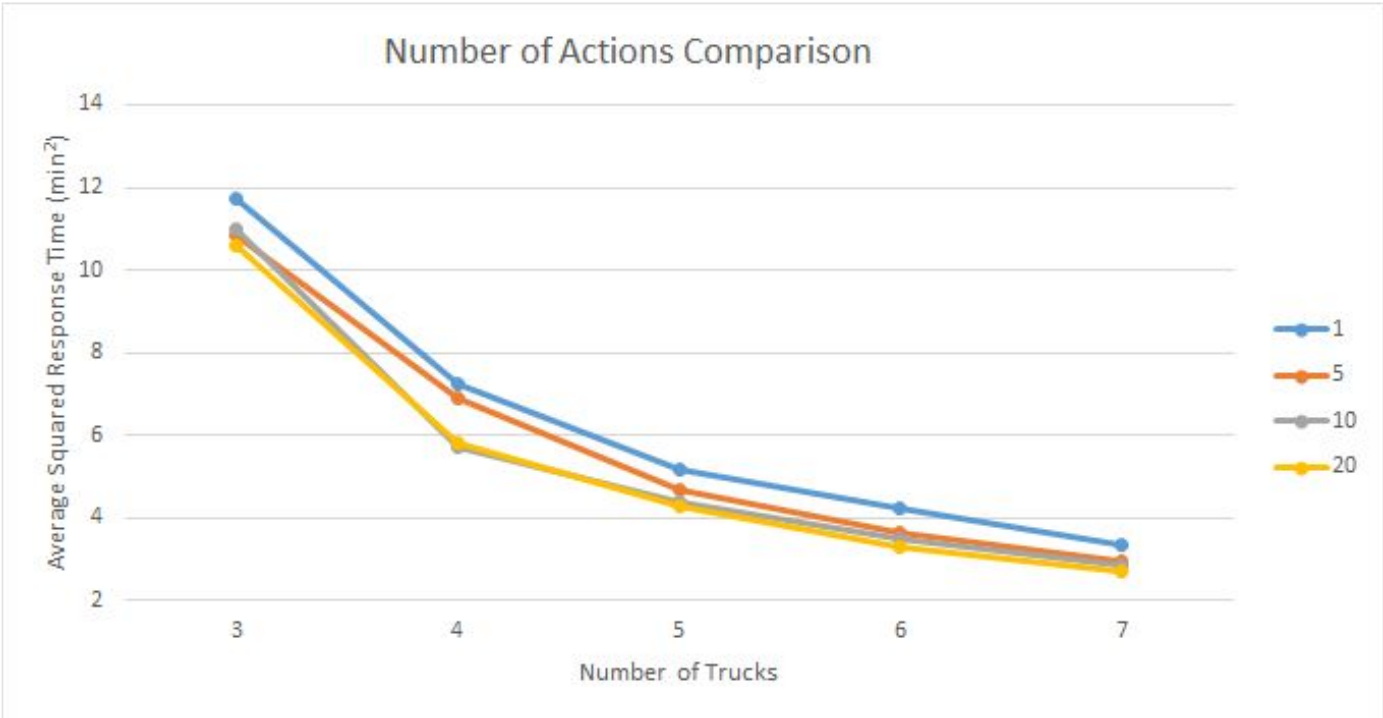
[5] "San Diego Demographic Data." *Census.gov*. N.p., n.d. Web. 21 Oct. 2016.

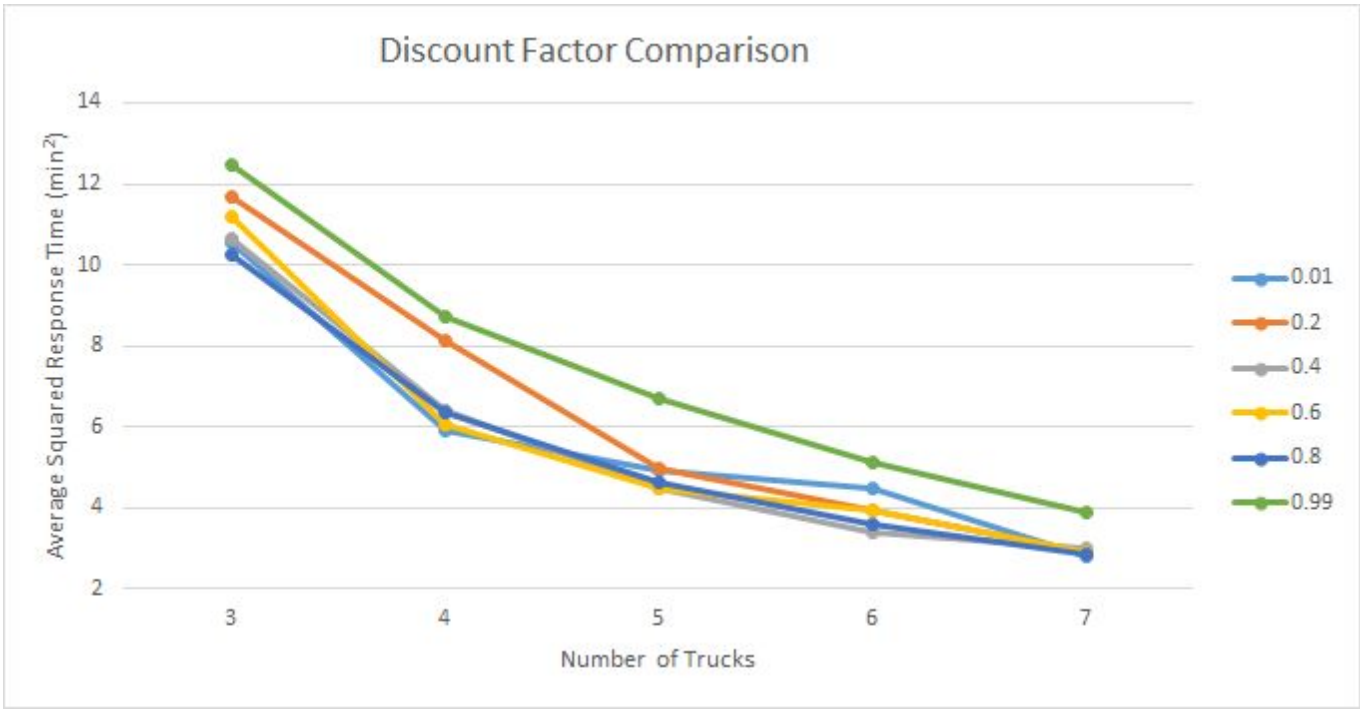
[6] Wong, Jeff. "Walking the Beat: Mining Seattle’s Police Report Data." *Bayes Impact*. N.p., n.d. Web. 21 Oct. 2016.

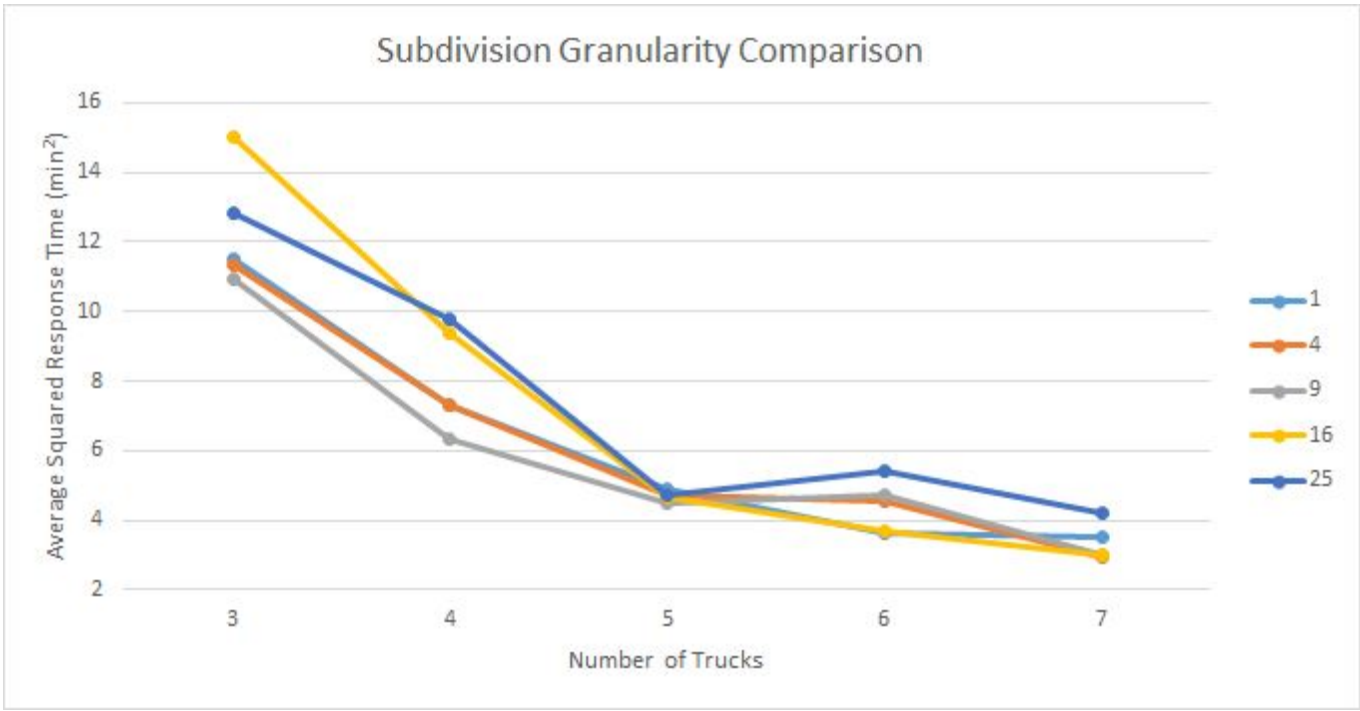
IV. APPENDIX

A. Hyperparameter Selection

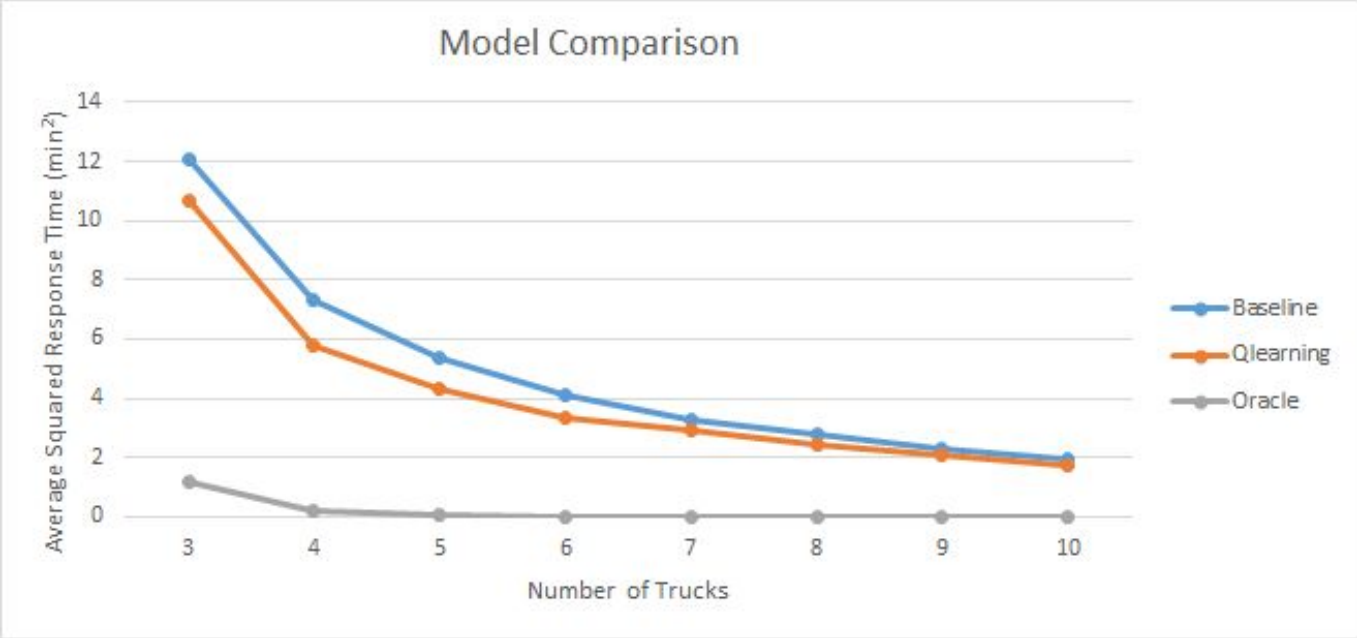





Discount Factor Comparison
Average Squared Response Time (min²)
14
12
10
8
6
4
2
3
4
5
6
7
Number of Trucks
0.01
0.2
0.4
0.6
0.8
0.99


Subdivision Granularity Comparison
Average Squared Response Time (min²)
16
14
12
10
8
6
4
2
3
4
5
6
7
Number of Trucks
1
4
9
16
25

B. Final Results



Overall Model Comparison (Average Squared Response Time (min^2)			
Number of Trucks	Oracle	Baseline	Qlearning
3	1.172	12.10311	10.65123
4	0.201	7.301133	5.768354
5	0.045	5.360975	4.309559
6	0.0125	4.090545	3.348983
7	0.00433	3.255942	2.893347
8	0.0025	2.751449	2.457537
9	0.00162	2.282803	2.047218
10	0.00153	1.961615	1.760112

C. Demonstration of Learning