Strategic Ambulance Deployment with AI

CS 182 - Final Project - Fall 2020

M. Elaine Cunha, Amanda Durham, and Hardik Gupta

1 Introduction

Car accidents are the world's leading cause of death for ages 5-29 and the eighth leading cause overall. There are about 1.35 million deaths per year around the world due to car accidents.[1] Emergency vehicle response time is critical in reducing the mortality rate from car accidents. On average, it takes emergency response vehicles 7 minutes from the time a 911 call is received to the time of arrival at the accident site. This response time can change dramatically depending on the area. For example, rural areas can have response times closer to 15 minutes and low-income neighborhoods are often subject to response times that are 10% higher than a higher income neighborhood.[2][3] Cities need to strategically organize and allocate emergency response resources in order to get to an accident in the fastest time possible.

Our goal is to assist city planners in optimizing both ambulance placement and dispatch throughout a city using AI principles learned in this class. This project will have two parts. The first part will explore whether we can optimize placement of a fleet of ambulances throughout a city so that we can meet a guaranteed response time for any accident in a given area. In the second part of this project we will explore how we can find the fastest route to an accident for each ambulance and dispatch the ambulance that can get to the accident the quickest. We believe that the combination of these two problems can be used to assist city planners when making decisions around planning and managing their emergency response teams.

2 Problem Specification

For both problems outlined above a square city grid will be simulated consisting of edges to represent the streets that can be traveled on, vertices as the intersections, black dots to represent the ambulances, and a red dot to represent the accident location. Each city block, represented by the edge between two grid points, will be assigned a time to travel to demonstrate traffic that may be present on different city blocks. A visual representation for the city grid is presented in Figure 1 below.

Finding a strategic placement of a fleet of ambulances throughout a city will involve modeling a Constraint Satisfaction Problem (CSP) to figure out where individual ambulances should be placed, in order to guarantee a maximum response time no matter where an accident occurs in the city. The scope of this problem will include finding the maximum distance that each ambulance within a fleet of n number of ambulances can travel to within a given time, T (guaranteed response time), such that each vertex within the city grid can be reached by an ambulance within time T. If the city grid cannot be covered in the given guaranteed time, an ambulance will be added to the fleet,

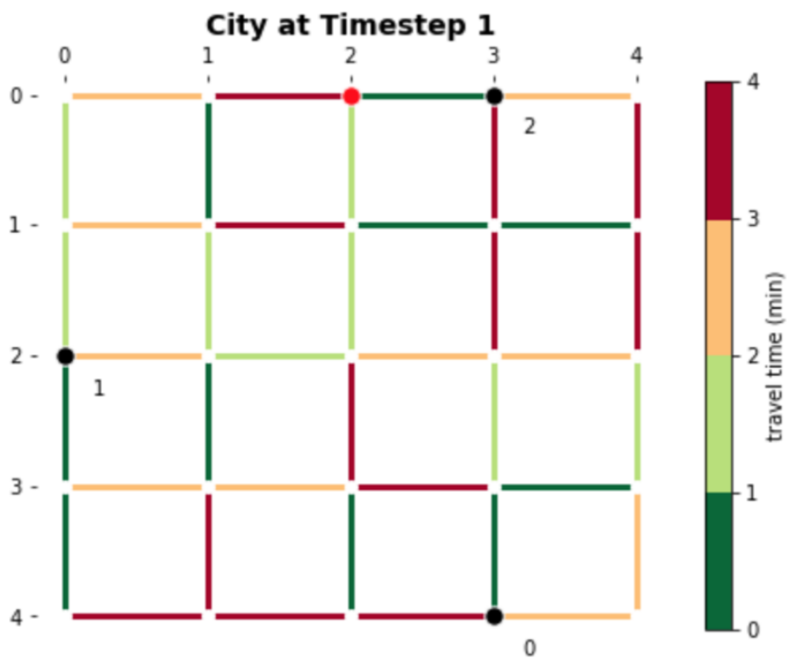


Figure 1: Visualization of the city grid

and the fleet size becomes $n + 1$.

Secondly, we will use a Markov Decision Process (MDP) to find the optimal response path to a given accident for each ambulance. Once this optimal policy is found, we will choose to send the single ambulance that can get to the accident the quickest. In this MDP, the agents are the individual ambulances that make up the fleet that resulted from the CSP in part one. The states that the agents can be in are the vertices that make up the grid of the city. The actions each ambulance can take includes moving in the north, south, east, or west direction. We have assigned a probability to moving in the intended or unintended direction, from which the transition function is based on. Lastly, the varying time to travel each edge of the grid resembles a negative reward for each move taken. A positive reward is granted for the agents that reach the accident location.

3 Approach

3.1 Constraint Satisfaction Problem for Ambulance Placement

We first tackled the problem of assigning locations for ambulances before an accident happens. As described earlier, each vertex in the city grid is a location and the traffic patterns in the city are represented by numerical weights assigned to the edges between the vertices. Given all possible locations and the traffic patterns, how should we place the ambulances such that any accident location can be reached within a guaranteed response time?

Ideally, we would solve a set of MDP problems for all possible accident locations. Given any accident location, if we have an optimal policy for the route an ambulance could take starting from anywhere, we would be able to find the global optimum for the placement of ambulances in the city. However, solving MDPs for all possible accident locations is computationally expensive for large city grids.

To make this problem computationally feasible, we decided to pursue a CSP framework. We implemented a CSP algorithm similar to the asynchronized weak commitment search and backtracking search algorithms discussed in class.

Instead of relying on an optimal policy given any accident location, we started with a heuristic for the route an ambulance may take. we defined the heuristic as the straight-line path with a single turn (L-shaped) between an ambulance's current location and the accident location. For example, if the ambulance is at the bottom-left corner and the accident occurs at the top-right corner, the ambulance would simply go north until reaching the top and then go east until reaching the accident location.

Given this heuristic, we defined the tradeoff between 1) the guaranteed upper limit to the time an ambulance may take to reach any accident location (guaranteed response time) and 2) the number of ambulances required in the city to satisfy the guarantee time constraint.

In our implementation, we define the maximum number of blocks around the ambulance's current location to search for a better placement. We prioritize searching for other locations along the direction (vertical or horizontal) that is the longest traversal between the ambulance and location that is farthest away. We define a maximum number of iterations to make with the current set of ambulances to ensure any location can be reached within the guaranteed response time. If the constraint is not met after the given number of iterations, we add another ambulance to the city and start the iterations again.

Summary of the CSP algorithm:

- Start with randomly placing n ambulances in the city grid
- Calculate the time taken by each ambulance to reach all vertices in the city grid by following an L-shaped route
- For each vertex, find the ambulance that can reach in the minimum amount of time
- Now find the vertex that is farthest away
- If the time taken to reach that vertex is greater than T (guaranteed response time), move the ambulance that is closest to that vertex along a defined number of blocks
- After a defined number of iterations, if the farthest vertex still cannot be reached, introduce another ambulance to the city grid and start over with $n + 1$ ambulances

Via this approach, we are able to find the minimum number of ambulances required such that any location can be reached following an L-shaped route within a guaranteed response time.

3.2 Markov Decision Process for Ambulance Dispatch

Next, we worked on solving the problem of selecting and dispatching an ambulance after an accident happens. To solve this problem, we employed a Markov Decision Process approach.

At this point, we already have ambulances placed across the city grid based on the output of the CSP problem defined earlier. The ambulances can move in the four cardinal directions (north, south, east, and west). These movements can be deterministic or probabilistic depending on the scenario we want to run. When an accident happens, we use the MDP framework to generate an optimal policy for an ambulance to follow from any location in the city grid to reach the accident location in the shortest amount of time. Note that we now allow for movements in any of the four cardinal directions at each vertex rather than just following an L-shaped path. If an action would move the ambulance off of the city grid, it instead remains in its current location. The travel time counts as a negative reward while solving the MDP. Reaching the accident location results in a big positive reward.

Finding the optimal policy is computationally expensive and therefore we only perform this step for a given accident location (rather than for all possible accident locations). Once we have converged on the optimal policy, we can simulate the route of all ambulances to the accident location. The ambulance that would reach the accident location in the shortest amount of time is selected as the responding ambulance. We further track step-by-step the movement of this ambulance to the accident location.

Summary of the MDP algorithm:

- At time step 0, we have ambulances placed within the city grid such that all possible accident locations can be reached by following an L-shaped route within the guaranteed response time (this is the output from the CSP algorithm)
- At time step 1, an accident location is randomly generated
- At this step, we solve the MDP to find the optimal policy to reach the accident location in the minimum amount of time from each vertex in the city grid
    - Note that we are no longer constrained to follow an L-shaped path
- Once we have a converged optimal policy, we can simulate the expected time taken by each ambulance to reach the accident
- We pick the ambulance that can reach the accident location in the shortest amount of time
- At time step 2 and onwards, we track the movement of the responding ambulance until it reaches the accident location

3.3 Code Structure

There are three key components in our code: 1) defining the architecture of the city as a grid and visualizing the ambulances and accident locations, 2) solving the CSP to place the ambulances in the city, and 3) solving the MDP to pick the responding ambulance and it's route to the accident location.

We wrote our code from scratch using only core Python functions and the Numpy library. We decided not to use any other libraries to allow for maximum flexibility and the opportunity to learn the under-the-hood mechanics for solving CSPs and MDPs in a simulation environment such as the one we defined.

We structured our code around a top-level class called Simulation. We first create a rectangular city (using Numpy matrices) where each vertex is a location (an intersection) in the city and the edges are the roads. The edges can have numerical weights that represent the travel time across the edge, which allows us to simulate varying traffic conditions. The travel time counts as a negative reward while solving the MDP.

Our code is written for flexible testing. It allows for altering a number of variables to represent different scenarios:

- City size
- Minimum and maximum weights for each edge
- Number of ambulances
- Probabilistic movement of ambulances in the four cardinal directions
- CSP parameters
  - The guarantee time constraint to meet while solving the CSP
  - Number of iterations allowed before another ambulance is introduced
  - Number of neighboring locations to test
- MDP parameters
  - Reward for reaching the accident location
  - Discount rate
  - Tolerance for convergence

Lastly, we used the Matplotlib library to create a visual representation of our simulated city. The visual representation allows us to track the locations of the ambulances over time as well as the accident location.

4 Experiments

4.1 Scenarios Evaluated

Our analysis evaluates the model by altering the parameters described above in three comparison scenarios:

1. **Increased Uncertainty in Movement**
   This scenario evaluates the impact on an MDP policy's "best route" if the likelihood of moving in an unintended direction (i.e., accidentally taking a wrong turn) is considered. We run two simulations for the same five-by-five city with three ambulances. The maximum traversal time for a city block is four minutes, and the CSP solves for ambulance placement to guarantee a six-minute response to any accident location. In the first simulation, we assume a 100 percent certainty in movement, thus ensuring an ambulance moves from an intersection in its intended direction 100 percent of the time. In the second simulation, we assume an ambulance moves in its intended direction 95 percent of the time and in either perpendicular

direction 2.5 percent of the time each. If an ambulance attempts to move off the defined city grid, it remains in its location for the time step. We compare the policy solution, the designated responding ambulance, and the time to respond in both simulations.

2. **Ambulance Placement**
   We test the effectiveness of our CSP implementation using a Monte Carlo analysis for both random ambulance placement and CSP placement. Each method is simulated 1000 times with a different random seed for each iteration. Again, we use a five-by-five city with a maximum traversal time of four minutes for any city block. Ambulances move in the intended direction 100 percent of the time. A guarantee of six minutes is specified for the CSP. As described in the Approach section, the CSP algorithm may add ambulances if it cannot solve the problem in a prespecified number of iterations. To maintain consistency, we initialize the corresponding random placement simulation with the same number of ambulances resulting from the CSP solution. In this scenario, we compare the distribution of ambulance response times of the two placement strategies.

3. **Complexity Analyses**
   This scenario measures the run time of our model by increasing the complexity of the problem with a larger city size. We measure on-the-wall clock time of solving both the CSP and MDP components of simulations for a five-by-five, ten-by-ten, fifteen-by-fifteen, twenty-by-twenty, and thirty-by-thirty city size. All simulations use the same random seed and a 100 percent certainty in ambulance movement. For this analysis, we scale up the constraint for the CSP proportionately as we increase city size (higher guaranteed response time for larger cities). As part of future work, we would solve for minimal guaranteed response times possible with a given number of ambulances while varying city sizes.

4.2 Results

1. **Increased Uncertainty in Movement**
   In this experiment, we ran two example simulations that were identical except in the certainty of movement assumption. In the first simulation, ambulances move in the intended direction 100 percent of the time. In the other, ambulances move in the intended direction 95 percent of the time and in either perpendicular direction the remaining five percent of the time. Notably, the second scenario leaves a zero percent chance of moving in the opposite direction of what is intended.

   Figure 2 shows for both simulations the layout of the city and the ambulance locations (reflecting the CSP solution) just after a randomly generated accident occurs at (0, 2). For each movement assumption, the model solves the MDP and generates a policy. Figure 3 includes a side-by-side comparison of the policies for both simulations.

   The resulting policy for movement under complete certainty (Fig. 3a) differs from the policy for uncertain movement (Fig. 3b). Under certain movement, the policy always points in the direction of the shortest path by travel time. With uncertain movement, the policy balances the shortest travel time with the likelihood of unintentionally moving away from the accident. Intersections of the three southernmost horizontal streets have a policy to always move northward, thereby minimizing the number of actions that could result in movement directly away (southward) from the accident, because the ambulance will never move in the opposite

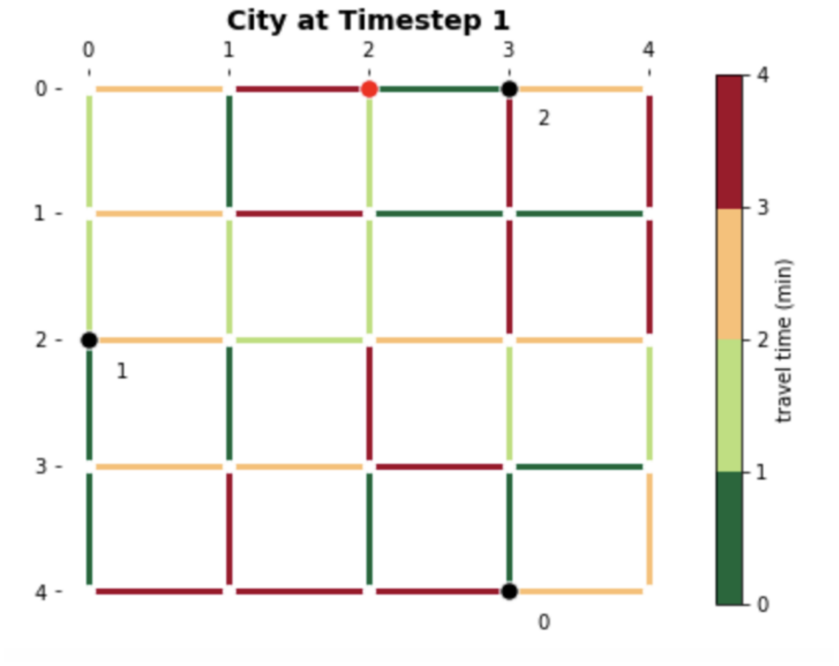
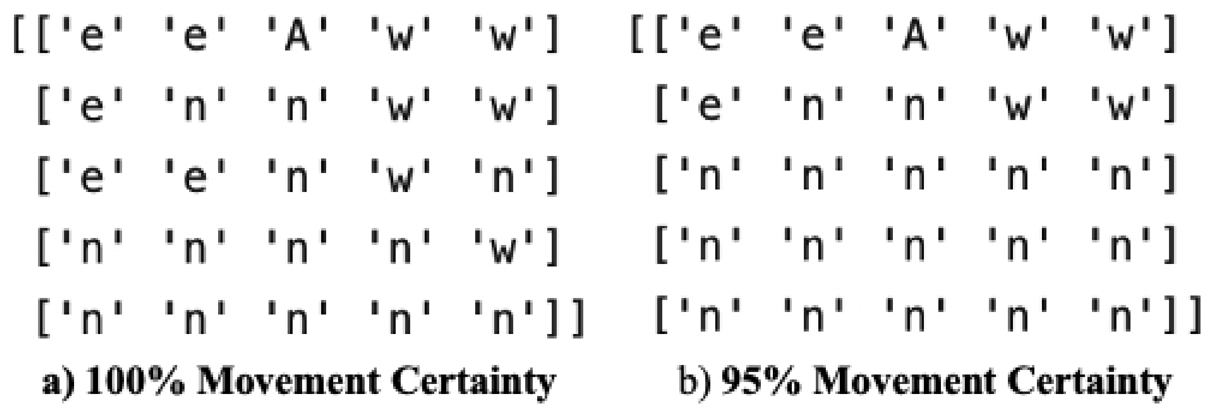


Figure 2: City initialization for movement uncertainty simulations with CSP ambulance placement (black circles) and a randomly generated accident (red circle)



```
[['e' 'e' 'A' 'w' 'w']        [['e' 'e' 'A' 'w' 'w']
 ['e' 'n' 'n' 'w' 'w']         ['e' 'n' 'n' 'w' 'w']
 ['e' 'e' 'n' 'w' 'n']         ['n' 'n' 'n' 'n' 'n']
 ['n' 'n' 'n' 'n' 'w']         ['n' 'n' 'n' 'n' 'n']
 ['n' 'n' 'n' 'n' 'n']]        ['n' 'n' 'n' 'n' 'n']]
```

a) 100% Movement Certainty **b) 95% Movement Certainty**

Figure 3: MDP policy solution for a) 100 percent certainty in movement and b) 95 percent certainty in movement

direction of what is intended. As the ambulance gets closer to the accident, minimal travel time is prioritized more, resulting in the same policy for the two northernmost horizontal streets as in the simulation with complete certainty of movement.

Both simulations result in the same responding ambulance and total response time, primarily due to the CSP placement that ensured close proximity to any potential accident. Therefore, in the case of uncertain movement, the CSP's placement also reduced the number of opportunities (intersections) in which an ambulance could accidentally move in the wrong direction.

2. **Ambulance Placement**
This experiment evaluates the effectiveness of placing ambulances according to a CSP. Figure 4 contains the distribution of ambulance response times when placed randomly and by a CSP. We see that CSP placement reduces both the average response time for a five-by-five city by almost 20 percent (over half a minute) and eliminates any response time longer than six minutes (reflecting the input specified for the response time guarantee). Note that approximately 18 percent of the simulations resulted in a randomly generated accident occurring at the location (we assume a uniform distribution for accident locations) of one of the ambulances and thus a response time of zero.

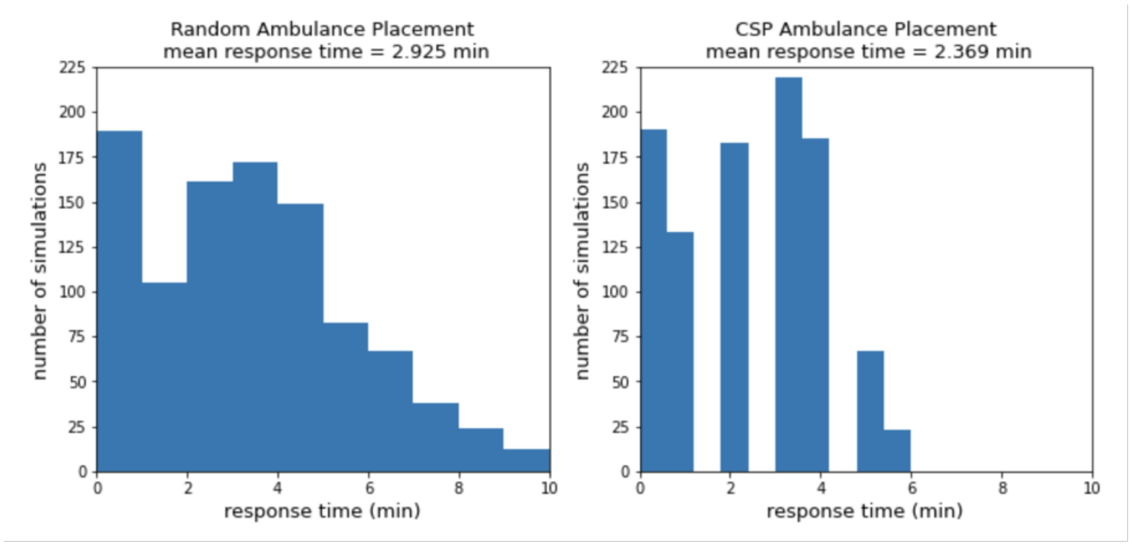


Figure 4: Histogram of ambulance response times for random ambulance placement (left) and CSP ambulance placement (right)

Figure 5 illustrates the difference in ambulance placement between the two approaches for a selection of simulations. We see that solving the CSP results in a more dispersed distribution of ambulances throughout the city, while random placement often lets ambulances become clustered.

3. **Complexity Analyses**
We assessed the scalability of the model by clocking on-the-wall run time for increasingly large city sizes. Figure 6 compares the run time for solving the CSP alone and the entire simulation (CSP and MDP). Grid sizes of less than fifteen intersections by fifteen intersections solved in their entirety within a minute. A twenty-by-twenty city solved in approximately 90 seconds, and a thirty-by-thirty gridsize did not converge on a solution after several hours of running. The MDP component of the model is largely responsible for the nonlinear increase in run time. However, we note that our parameters for the CSP scaled with the city size (e.g., the time guarantee increased), while the parameters for the MDP remained the same.

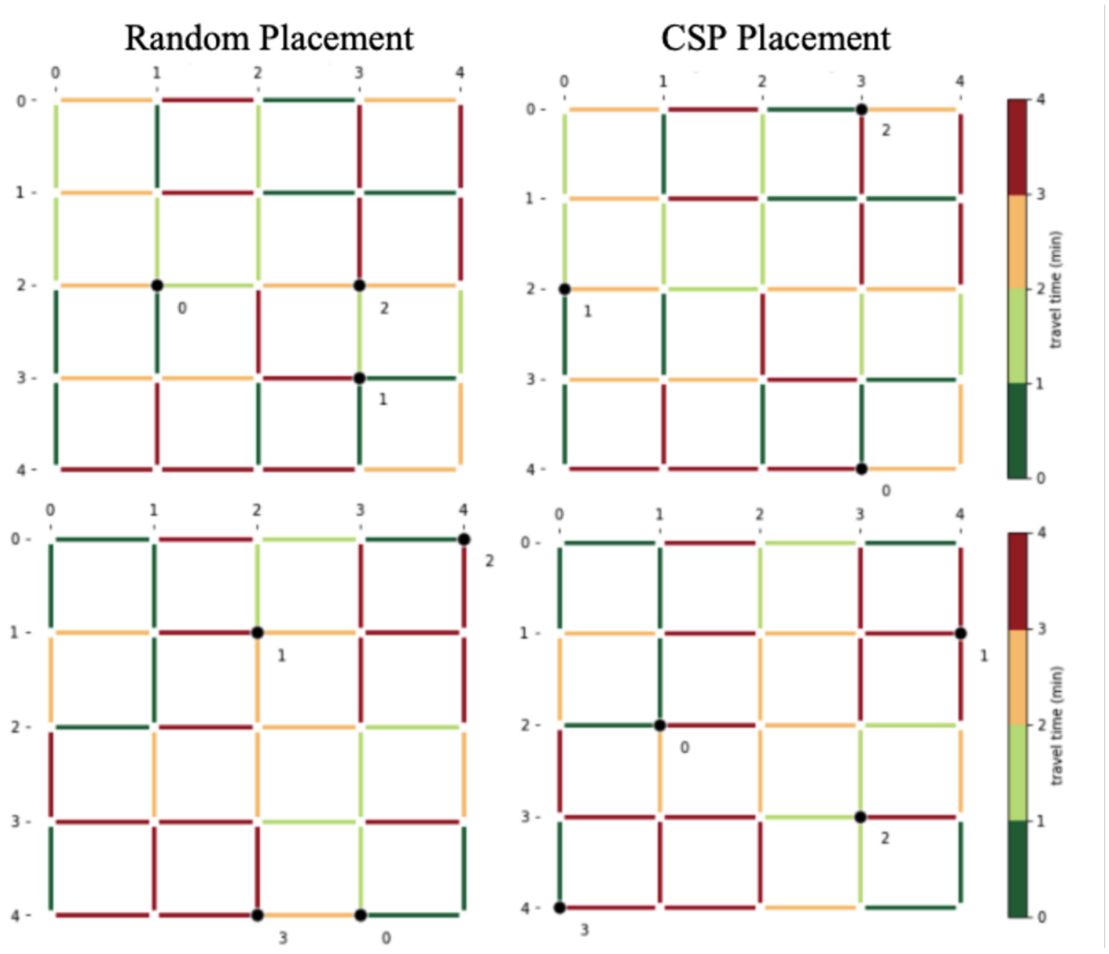


Figure 5: Example ambulance locations for random (left column) vs. CSP (right column) placement

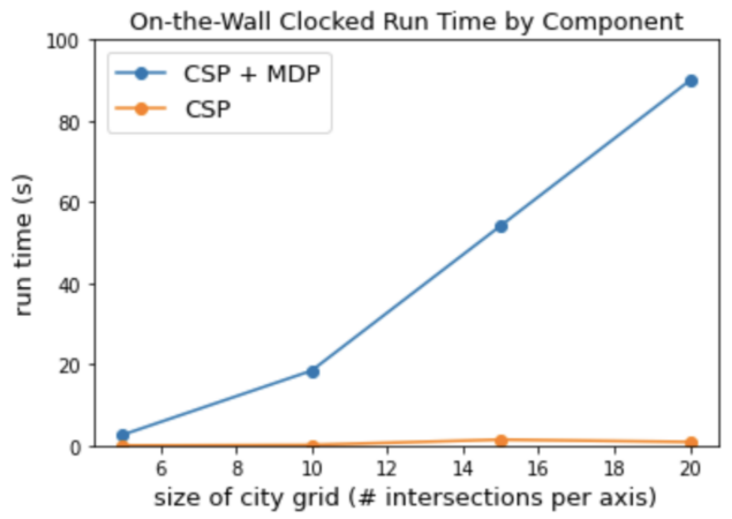


Figure 6: Run time of model components by city size

5 Discussion

5.1 Summary

Around the world, mortality rates from car accidents remains one of the leading causes of death. Emergency response time to accidents is critical for reducing mortality rates from car accidents. We believe strategic placement and deployment of emergency response teams is a critical exercise for city planners.

In this project we addressed both placement and dispatch optimization of a fleet of ambulances across a simulated city grid. A constraint satisfaction problem was used to compute the location that each individual ambulance should be placed while at rest (i.e., before responding to an accident), in order to meet the constraint of a guaranteed response time to an accident that may occur anywhere in the city grid. In the CSP, we also were able to determine if a new ambulance should be added to the fleet in order to meet the constraint. Once optimal placement of each ambulance was determined, an MDP was developed and executed to find the policy denoting which direction an ambulance should move at any vertex in the city in order to reach the accident as fast as possible. The MDP then assigns the ambulance that is able to respond to the accident the fastest based on the policy.

5.2 Takeaways

We were able to model an urban ambulance dispatch response using an MDP with variable parameters in order to examine how certain parameters affected the optimal response policy. Our MDP model is able to run on a varying number of ambulances in the city, because it has to consume the number of ambulances that are required to meet the constraint from part one. It is also executable across different sized city grids, with different time travel times discretization across each city street. We ran a few scenarios to find strengths and weaknesses in our model and were ultimately able to discover the following themes.

The first main theme we discovered was the importance in finding a balance in certain parameters of the MDP inorder to solve your intended goal. Our intended goal of the MDP was to get an ambulance to respond to an accident using the fastest route possible. We tested the effect that the unpredictability of each move an agent takes has on the response policy. This scenario demonstrated the importance in finding a balance in movement unpredictability. We found that when increasing the uncertainty in movement, i.e., making it more likely that an ambulance moves in the unintended direction with each step, our agents were more inclined to move in the direction that ensured they wouldn't be moving in the opposite direction of the accident, instead of traveling on the shortest route of the accident. Thus, in MDP implementation you can utilize the uncertainty probability in your transfer function as a way to optimize for certain behavior. In our case, a smaller uncertainty was more realistic and also allowed for us to optimize for the shortest path, rather than simply always arriving at an accident no matter the path. Another parameter that had to be balanced in order to solve for the intended goal was the reward function. We originally ran the MDP using a reward of +500 for making it to an accident. This high reward ended up making the cost of each move negligible, and thus it was difficult to distinguish which ambulance had the fastest route because the policy did not account for travel times.

A second theme we came up against in our model was scalability. We found that our model performed well on a 5 by 5 city grid, executing the MDP in about 10 seconds. For a 20x20 city grid, our model was executed in about 1.5 min. However, at a 30x30 sized grid our model took several hours to run, thus showing our model is limited in its scalability. Scalability is an important factor in our model, as will be discussed below, since we ultimately would want this system to be used by a wide range of cities, be it small, condensed urban areas or spread out, rural towns.

Lastly, our first iteration of this project did not include utilizing a CSP in order to place the ambulances throughout the city, but instead used random ambulance placements to run the MDP model from. For the second iteration of the project (post-presentation), we were able to implement the CSP to place ambulances in locations which satisfied the constraint of making it to an accident within a certain amount of time. This ended up improving the results of our MDP by both decreasing the average response time of each ambulance, as well as ensuring that there was a maximum response time for each potential accident location.

5.3 Future Implementations

There are a number of future improvements that we would want to make to this model in order to make it a more realistic simulation of a city and the many obstacles that can affect both placement of ambulances and their response policy. As mentioned above, the first improvement would be around making the model scalable across larger city sizes. This will be important in order to be useful to more rural and remote areas of the world. This would be one of our first priorities since it is often these more remote towns that need to be more resourceful with their emergency response teams.

Secondly, we would want to add advanced scenarios to our city simulation to more accurately represent what occurs in a city. A perfect square grid is not the best representation of a city, and many cities/towns do not follow a grid structure. We would want to extend our city simulation to take on a variety of city architectures. To do this we could potentially integrate with Google Earth or Google Map APIs to map out city limits and streets. Additionally, traffic is not always consistent on each block at all points of the day, as our model shows. To improve this, we would want to add dynamic traffic patterns that represent on and off peak times of travel in a city. Another interesting advancement we would look to implement in the future is varying the parameters we discussed above at each intersection. This would represent differing uncertainties (i.e., the likelihood of making a wrong turn) depending on where you are within a city.

Lastly, an advancement we would want to make to the model is the ability to respond to multiple accidents that either happen simultaneously or one after the other. Simultaneous accidents would require us to run a model similar to ours but that can decide between travel time ties if necessary. Accidents that happen in sequence, would require us to understand how the positions of each ambulance has changed based on the response to the previous accident, re-organizing the placement of the fleet in order to satisfy the constraint of our CSP model, and the solving of the MDP for the next accident.

All of these improvements would allow us to better simulate a city and the factors that can affect both parts to our problem. These implementations would ultimately allow for our model to be more helpful to city planners during emergency response resource planning.

A Appendix 1 - System Description

All of our source code is saved in the Python file ‘simulation.py‘. The two classes we defined can be imported as follows along with standard libraries:

- `from simulation import Simulation`
- `from simulation import Ambulance`
- `import numpy as np`
- `import matplotlib.pyplot as plt`

After importing these classes, a new simulation can be initiated:

- `demo_simulation = Simulation(city_size=5, min_ambulance=3, max_time=4, p_int=1)`
  - Use city_size to define the size of the square grid.
  - Use min_ambulance to define how ambulances to start with. Note that the CSP algorithm will iteratively add ambulances if the guaranteed response time constraint is not met.
  - Use max_time to define the upper limit of the time taken to traverse an edge. Note that the lower limit is always 1 (minute).
  - Use p_int to define the probability of moving in the intended direction. (1-p_int)/2 will be the probability of moving each of the perpendicular directions.

To solve the CSP for placement of ambulances across the city grid, use the following command:

- `demo_simulation.csp_ambulance(guarantee_time=6, verbose=False)`
  - Use guarantee_time to define the guaranteed response time constraint for reaching any vertex in the city by following an L-shaped route.
  - Use verbose to turn on/off the visualization of the city grid for each iteration in the CSP algorithm.

Up until this point, we are at time step 0. To step through the simulation, use the following command:

- `demo_simulation.step()`
  - An accident location is randomly picked when going from time step 0 to 1. The MDP is solved and the responding ambulance is picked.
  - Using this command again steps from 1 to 2 and onwards, which tracks the movement of the responding ambulance until the accident location is reached.

To visualize the the city grid at any step, use the following command:

- ``` `demo_simulation.show_city() ```

To visualize the optimal policy after an accident location is generated, use the following command:

- `demo_simulation.show_policy((demo_simulation.accident_location[0],`

```
    demo_simulation.accident_location[1]))
```

We have attached a Jupyter notebook that demonstrates the use of our codebase. We have also attached another Jupyter notebook that runs the scenarios we have discussed in this report.

Our codebase and the Jupyter notebooks can also be found at this link: `https://github.com/hgupta18/ambulance-dispatch.git`.

B Appendix 2 - Group Makeup

M. Elaine Cunha

1. Project discovery Proposal - All individuals discussed our various interests and what type of problem we wanted to work on. We all ultimately landed on a project proposal that allowed for all of our strengths and weaknesses to be developed. All group members met frequently together to discuss progress of project.
2. MDP Coding Lead - Lead implementer of MDP Portion of the code.
3. CSP Coding Partner - Worked in partnership with Hardik to develop CSP portion of model in between presentation and final project deliverable.
4. Experimenter - Ran the experiments mentioned above to test the various strengths and limitations of our model.
5. Deliverables - Created a documented and presentable version of deepnote simulation. p Worked with group on presentation deliverable and final project write-up.

Amanda Durham

1. Project discovery Proposal - All individuals discussed our various interests and what type of problem we wanted to work on. We all ultimately landed on a project proposal that allowed for all of our strengths and weaknesses to be developed. All group members met frequently together to discuss progress of project.
2. CSP and MDP outline by hand - Outlined the parts of the problem by hand using definitions learned in class.
3. Presentation Lead - Lead on presentation content.
4. Project Write-Up Lead - Lead on project write-up content.

Hardik Gupta

1. Project discovery Proposal - All individuals discussed our various interests and what type of problem we wanted to work on. We all ultimately landed on a project proposal that allowed for all of our strengths and weaknesses to be developed. All group members met frequently together to discuss progress of project.
2. City Simulation Lead - Lead developer of City Simulation portion of the code.
3. CSP Coding Partner - Worked in partnership with Elaine to develop CSP portion of model in between presentation and final project deliverable.
4. Deliverables - Worked with group on presentation deliverable and final project write-up.

References

[1] "Road Safety." World Health Data Platform, World Health Organization, https://www.who.int/data/gho/data/themes/road-safety

[2] Mell, Howard K et al. "Emergency Medical Services Response Times in Rural, Suburban, and Urban Areas." JAMA surgery vol. 152,10. 2017.

[3] Hsia RY, Huang D, Mann NC, et al. "A US National Study of the Association Between Income and Ambulance Response Time in Cardiac Arrest." JAMA Netw Open. 2018

[4] "Uber Nairobi Ambulance Perambulation Challenge". Zindi, https://zindi.africa/competitions/uber-nairobi-ambulance-perambulation-challenge