Dynamic Resource Allocation for Fire Incidents

CS561 Final Report

Rishav Chourasia(140101086)
Anish V Monsy(140101081)
Soumik Mukhopadhay(140101085)
Uppinder Chugh(140101084)

***Abstract*— Efficient allocation of resources are paramount to a well functioning society. Dynamic allotment of resources for fire incidents can provide a solution that reduces damage and loss of life while conserving taxpayer dollars in the process. Reinforcement Learning(RL) field has seen interesting advances and improvement in par with human level performances in single as well as multi-agent systems. We apply three different techinques — Simple Feedforward DQN, DQN with convolution and recurrent layers and Proximal Policy Optimization(PPO) — on a grid world simulation of San Diego region's fire incidents based on geolocation of area zipcode.**

I. Motivation

There are several cases of critical emergency events across the country each year, which affect millions of people. The size of an emergency response by responders may scale from a single paramedic to multi-state rescue agents fighting gigantic fires. The time it takes for first responders to act quickly at the scene is highly critical during emergency events and can often make the difference between life and death. To make sure enough responders are available at any given time, standard staffing and resource usage is very high which makes such allocation an important potential application for optimizations. Thus, a static model that executes allocation of resources dynamically would be highly useful to government agencies.

II. Approach

A. Objective

In a given region, emergency incidents may occur at any given time and dynamic resource allocation is essential for handling these incidents. We have chosen the city of San Diego as our region and isolated only the incidents that were forwarded to the fire department for testing our approach. We use a reinforcement learning approach for tackling the problem.We give as input to our model the current condition of the city at a particular time step and the model is expected to allocate the resources and solve the incidents with minimum total response time. We have assumed only one fire truck for this project but this can be easily extended to a multi-agent firetruck system. Our approach can be extended to solve any type of emergency incidents requiring instantaneous response.

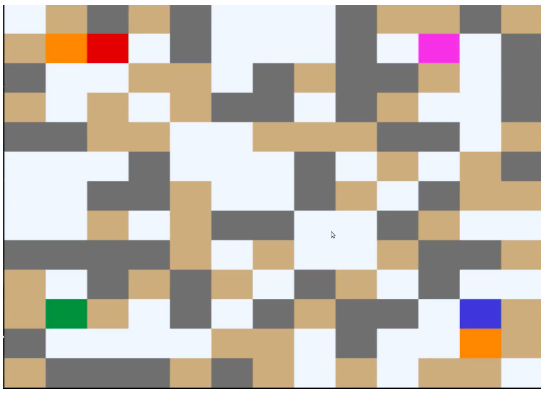
Fig. 1. Pommerman Game Environment

B. Data and preprocessing

We have collected our data from Open Data Portal of the city of San Diego [2]. The data we have chosen is of the year 2016 and format is as follows.

{agency_type:'Fire', call_category:'Emergency Medical Response', city:'SAN DIEGO', incident_number:'CV16013448', jurisdiction:'San Diego', problem:'Medical Aid', reporting_date :'2016-07-29T18:56:51', state: 'CA', zip:91911}

We have converted the zipcode of the incidents into latitudes and longitudes by using a direct mapping given in [1]. These latitudes and longitudes were scaled and discretised to obtain coordinates in the X,Y axis for usage in our simplified environment. A similar preprocessing was done on the time field of the incident since our environment expects discrete time frames. We have divided the emergency incidents in the ratio 0.8:0.2 for training and testing. For training and testing the model, we sample episodes having duration 1000 time steps.

C. Environment

We have simplified the real world situation of reporting and response of emergency incidents into a game-like environment borrowed from Pommerman [3] as can be seen in 1. We have modified the Pommerman environment to fit into our specification.

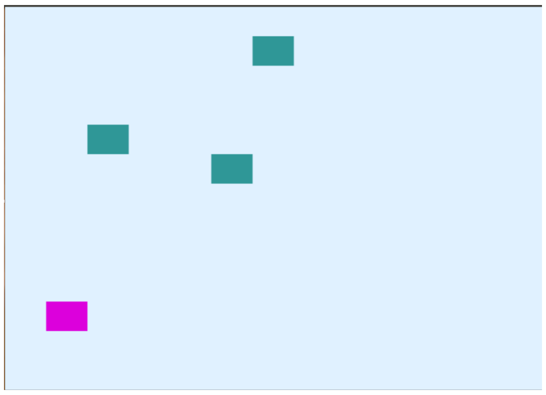

Fig. 2. Our Game Environment (Pink block represents Firetruck and Green block represents Fire locations)

It contains a 13*13 grid with a player,obstacle or powerup object occupying each grid position. We modified this by making it single player,removing obstacles and having only one type of reward object which in our case is the location of the emergency incident. We dynamically add this reward object on the grid based on the contents of the emergency incident data file. Each player has five actions:- Left,Right,Up,Down or Stop. We have simplified the process of the fire engine dealing with the incident as merely reaching the incident location on the grid. An example snapshot of the environment can be seen in 2

D. Model

Reward Function: We have chosen the reward function of an agent on reaching the location of the incident as the following.

$$Reward(t) = max(0.01, 1 - 0.01t)$$

where t is the number of steps between the time at which the incident appears on the grid to the time at which an agent reaches the location of the incident.

State of the environment: state of the environment is described by a 507 sized vector representing grid positional information and a 9-sized vector representing agent specific information. The entire state of the environment can hence be described in a 516-sized vector.

We have tried 3 different algorithms.The first two models are deep Q learning models.For these two models, we additionally used experience replay to speed up training.

1) DQN with convolution and recurrent layers: The first model as shown in 3 involves processing the 507 sized grid positional information through a two layer convolutional neural network.The first layer has 6 feature maps and the second layer has 12 feature maps with a kernel size of 5*5. In parallel, the 9 sized agent specific information is passed through a single hidden layer neural network. The combined output after this stage is a 364 sized vector which is passed to two recurrent networks, one for calculating the value stream and another for the advantage stream which is based on the dueling network architecture. The recurrent networks consist of two LSTM layers with 64 hidden units in each layer. The recurrent networks help the model to understand change of the environment over time in a better manner. The advantage and value streams are then combined to obtain the final required Q value for each of the 6 actions.

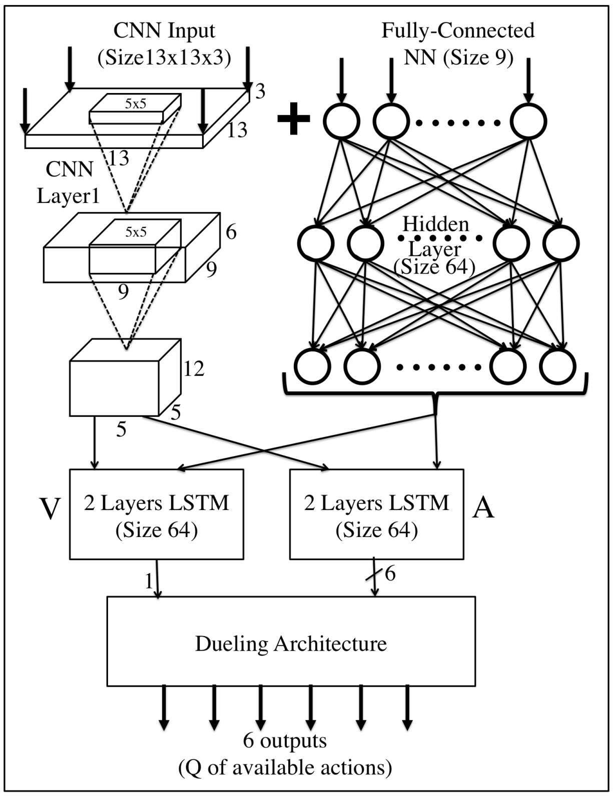

Fig. 3. DQN with convolution and recurrent layers

2) Feedforward DQN: The second model(4) is relatively simpler being a feedforward network with 2 fully connected hidden layers with 64 units each. At the output layer , we obtain the Q-value for each of the 6 possible actions of the agent.

3) Proximal Policy Optimization: Proximal policy opitimization (or PPO), is a reinforcement learning algorithm based on policy gradient methods which has shown to provide state-of-the-art results on the standard benchmark - *Atari*. In a supervised setting of training the agents, we can easily use a cost function which will further be used to update the policy parameters of the model for achieving good results. The loss function employed by PPO is shown below:

$$\mathbf{L^{CLIP}}(\theta) = \mathbf{\hat{E}_t[min(r_t}(\theta)\mathbf{\hat{A}_t, clip(r_t, 1} - \epsilon, \mathbf{1} + \epsilon)\mathbf{\hat{A}_t]}$$

Here, we use a neural architecture similar to the Feedforward DQN except for the last layer which now has a softmax function.The output of this model is fed to PPO.

- θ is the policy parameter, which are the set of weights of the neural network
- $\hat{E}_t$ represents the experimental expectation over timesteps

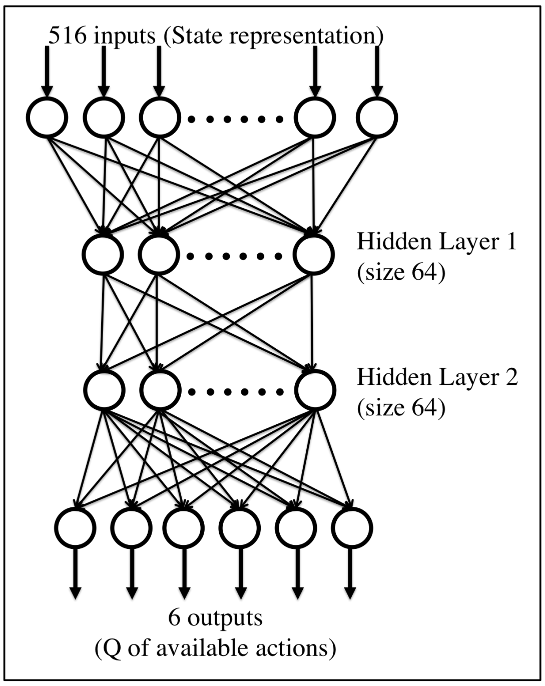

Fig. 4. Feedforward DQN

- r_t is the ratio of the probability under new and old policies, respectively
- $\hat{A}_t$ is the estimated advantage at time t
- ϵ is a hyperparamater which is usually 0.1 or 0.2

III. Results

During the training of the three approaches,we found that the training of the DQN with convolutional and recurrent layers was quite slow. Since we did not have powerful computational resources,we avoid this approach. For the simple feedforward DQN, we observed the relation between episode rewards and the number of episodes it has been trained on given in 5.Each episode is of 1000 steps.Each episode may not have the same number of incidents but if the model is training well, an overall increasing curve with noise can be expected.

This indicates that the models is not training well on our incident data. However for the PPO based approach, we find the relation in 6. Hence we have decided to use the third approach for our task. We have trained the model on 500 episodes and then the model is evaluated on the test split of the data.

We compare our model's result against a deterministic agent that was used in the Pommerman environment. This goes to the nearest incident if it is within two reachable moves or does random moves otherwise.We call this deterministic agent as SimpleAgent.

On an episode randomly chosen from the test set, SimpleAgent gives a reward of around 9.4 roughly, whereas our agent on being trained on 500 episodes gives around 8. With

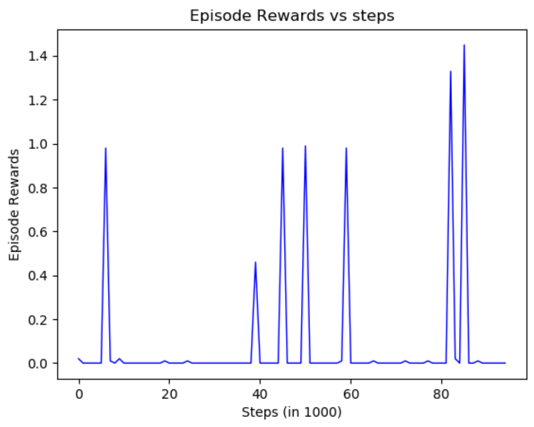

Fig. 5.

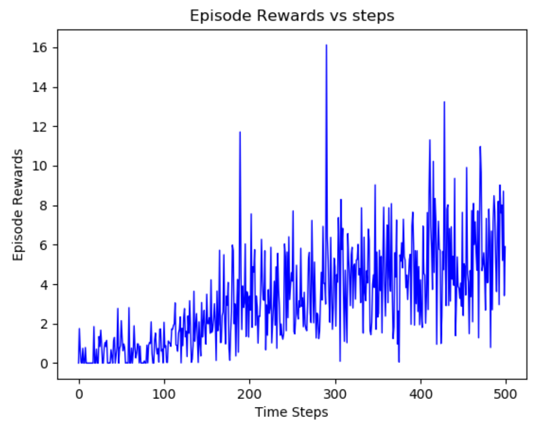

Fig. 6.

enough training we believe it will surpass the performance of SimpleAgent.

IV. Conclusions

We created an environment for simulating fire truck scheduling in the San Diego region and analysed performance of RL algorithms like deep Q learning network and its variants: dueling architecture , recurrent architecture with prioritised experience replay. We chose the PPO approach and we believe that with sufficient training it will surpass the performance of a deterministic algorithm on the same task.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

[1] Gaslamp Media, San Diego zipcode to Geo-location mapping.
[2] DataSD, Government of San Diego, Government Public Data Portal.
[3] Pommerman, PlayGround: AI Research into Multi-Agent Learning