Law Enforcement Simulation Project Roadmap

Sedar Olmez

November 2018

1 Introduction

In our model, we will be focusing on a specific domain, that being the law enforcement agencies. Therefore, our ABM (Agent Based Model) will be tailored to acknowledge all the characteristics of this domain. It is crucial that the model behaves in the utmost realistic manner as possible, although this will always be an estimate and not definitive as we are focusing on a non-deterministic environment. The goals of this project are:

- Create a simulation of law enforcement agent's behaviour when it comes to responding to crime.
- Ensure the optimum number of resources are distributed and exhausted according to crime severity to minimise the number of crimes.
- Categorise the severity level of crimes and produce a mechanism to delegate response to more fitting entities i.e. local crimes to neighbourhood watch, terrorist attacks to armed patrol and so forth.

The initial model at a low calibre will be developed on Netlogo, the literature followed for this task will be that of [1] which covers all areas of agent-based modelling in Netlogo. Markov Decision Processes (MDP) can be embedded in the simulation to ensure the dynamic changes in the environment are always accounted for and that the correct resource aims to reach the highest reward which will be tackling a specific crime. An example scenario is presented below:

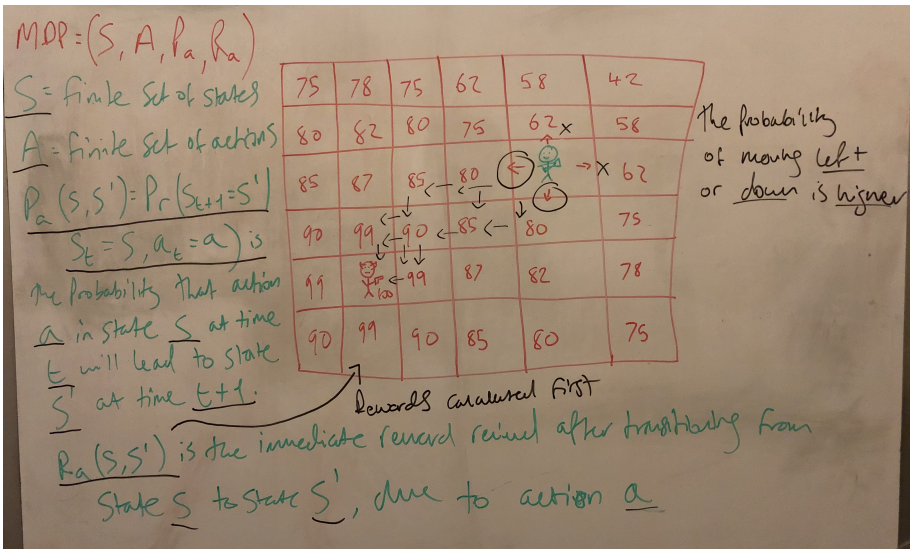

Figure 1: An example of Markov Decision Process in an ABM

In Figure 1, the armed police officer is delegated the task of getting to the criminal. The mechanism for task delegation is not clear at this early stage but will be worked on.

In Figure 1, we wanted to present how useful Markov Decision Processes can be to this project. The reason for why this is necessary is that the agents can not just move randomly, this will lead to anomalies and not answer the second bullet point above, we need structure and efficiency if randomness is involved then agents will behave in a non-optimal fashion.

Everything we came up with above is prone to change on demand, now we will focus on understanding the:

- Environment which the simulation will be based on
- The variables which will be tweaked to present new phenomena like increasing the number of officers, increasing the number of crime and so forth.
- The different types of resources represented as agents and their characteristics i.e. abilities, the maximum speed of police vehicles and so forth. The more detail the more accurate the simulation becomes.
- How Markov Decision Processes will be embedded into the simulation.

2 Agents

For this model to be simulated with accuracy, we need to identify the agents (turtles according to NetLogo). The model will start off simple but could be

extended into a far more complex model with the addition of new agents and capabilities. As of now, we have the following:

- People (pedestrians)
- Constable
- Sergeant
- Inspector
- Criminal
- Police truck
- Police car
- Police Helicopter

We produced a graph which matches each agent to applicable actions:

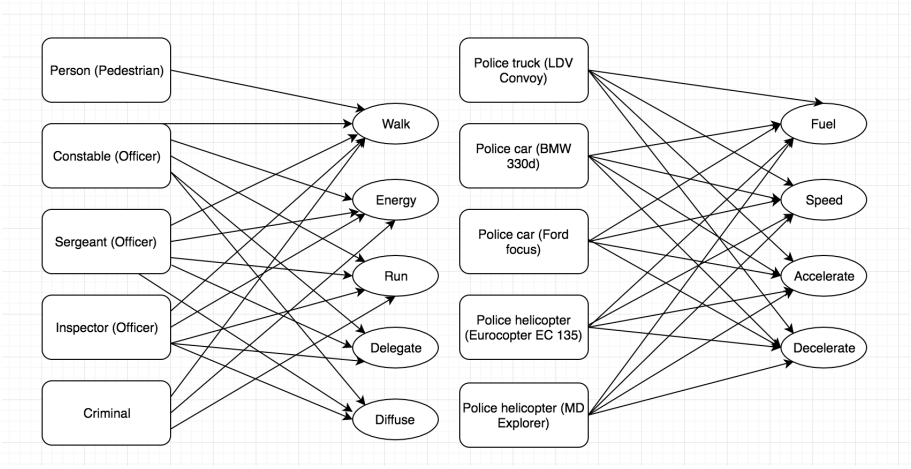

Figure 2: A representation of actions assigned to agents

3 Severity classification

The notion of crime severity will be used to delegate tasks to the most equipped resource to handle that task. We identified three severity levels:

- Green - Crimes committed towards individuals
- Amber - Crimes committed to property
- Red - Terror related crimes

Now we present a graph which maps out the type of crime assigned to a severity level and the agent that can diffuse that crime. The crime-severity score data tool by the ONS (Office for National Statistics) was used here to identify the types of crimes:

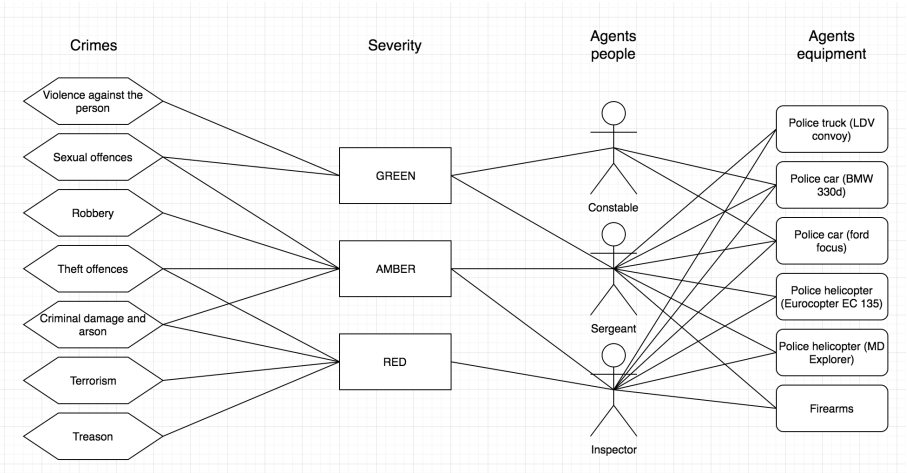

Figure 3: Crime, severity and agents classification

Crime severity weight calculation: *Weight = (custody rate * Average custodial sentence length) + (community order rate * community order equivalence) + (fine rate * fine equivalence)*

Where:

- Custody, community order, fine rate is the proportion of sentences for an offence that resulted in the outcome of a custodial sentence, community order, fine being given as punishment.
- Average custodial sentences length is the average length of time that all offenders were sentenced to serve in custody.

4 Task delegation and its importance

In a dynamic environment, where the goal is for agents to get to a specific coordinate, then task delegation is necessary. In our simulation it is possible to end up with a scenario such as this:

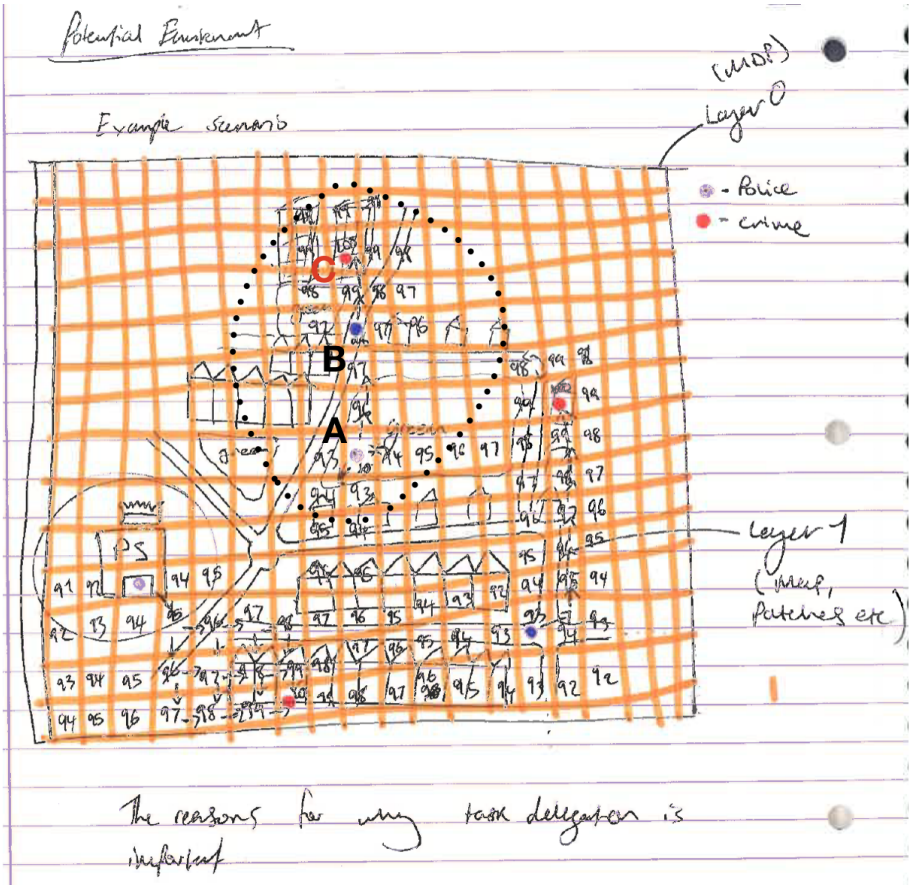

Figure 4: A scenario where task delegation is required

If we take a look at agents A and B and crime C. It is clear that B will move towards C before A, however, due to MDP both A and B see C as the closest reward thus they will both move to it. This is one reason for why a *task delegation* mechanism is required to ensure agents can delegate tasks to one another, this is a phenomena that MDP lacks the ability to solve.

5 Development

In this section, we present the prototype implementation of the model starting with its early stage. As we progress with the model, more will be added with the improvements made.

5.1 Core backend

The back-end logic of MDP has been implemented. The hardest part was to get the officers to move to the closest crime scene. The value iteration algorithm was implemented which is categorised as a reinforcement learning technique:

1: **Procedure** Value_Iteration(S,A,P,R,θ)
2: **Inputs**
3: S is the set of all states
4: A is the set of all actions
5: P is state transition function specifying $P(s'|s,a)$
6: R is a reward function $R(s,a,s')$
7: θ a threshold, $\theta>0$
8: **Output**
9: $\pi[S]$ approximately optimal policy
10: $V[S]$ value function
11: **Local**
12: real array $V_k[S]$ is a sequence of value functions
13: action array $\pi[S]$
14: assign $V_0[S]$ arbitrarily
15: $k \leftarrow 0$
16: **repeat**
17: $k \leftarrow k+1$
18: **for each** state s **do**
19: $V_k[s] = max_a \sum_{s'} P(s'|s,a) (R(s,a,s')+ \gamma V_{k-1}[s'])$
20: **until** $\forall s \ |V_k[s]-V_{k-1}[s]| < \theta$
21: **for each** state s **do**
22: $\pi[s] = argmax_a \sum_{s'} P(s'|s,a) (R(s,a,s')+ \gamma V_k[s'])$
23: **return** π, V_k

Figure 5: Value iteration for MDPs, storing V

I implemented a small model which demonstrates how value iteration works:

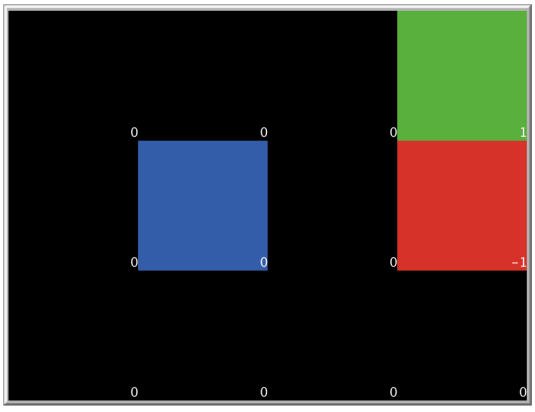

Figure 6: Value iteration one

In Figure 6, the first iteration sets a goal patch as 1 reward, a bad patch as

-1 and the rest 0. When we perform the calculation for the next iteration, the patch left of the green patch should become 0.74 using the following formula:

$$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)\,(R(s,a,s') + \gamma V_k(s')) \quad \text{for } k \geq 0$$
$$V_k(s) = \max_a Q_k(s,a) \quad \text{for } k > 0.$$

Figure 7: Value iteration formula

Let V_k be the value function assuming there are k stages to go, and let Q_k be the Q-function assuming there are k stages to go. These can be defined recursively. Value iteration starts with an arbitrary function V_0 and uses the above equation to get the functions for $k+1$ stages to go from the functions for k stages to go.

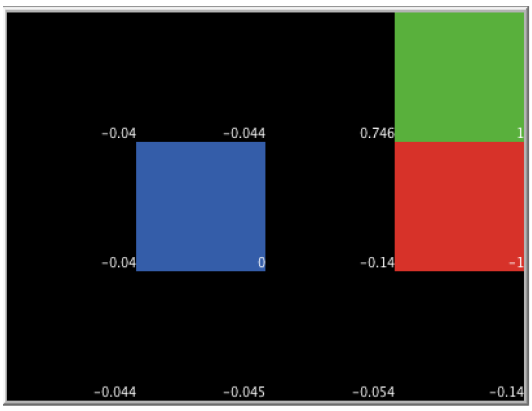

Figure 8: Value iteration two

And if we want equilibrium, we keep iterating until the values converge:

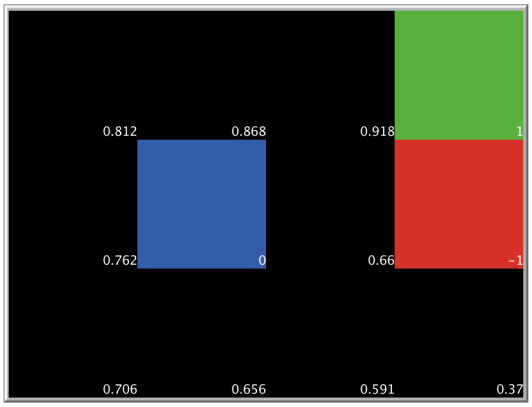

Figure 9: Value iteration equilibrium reached

Now that the policy is set (Figure 9) the agent can move around the map and use the most optimal path to the green patch avoiding the red one at all costs.

Our model uses this same principle but with moving agents.

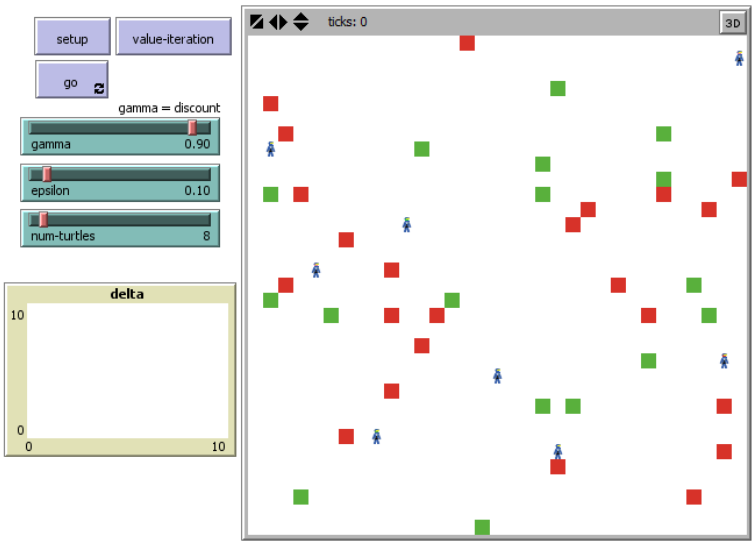

Figure 10: Value iteration model version 1, 8 officers

The first step is to calculate the optimal policy for the current state by clicking the *value_ iteration* button (this will change in the future). Once equilibrium is reached, we run the model.

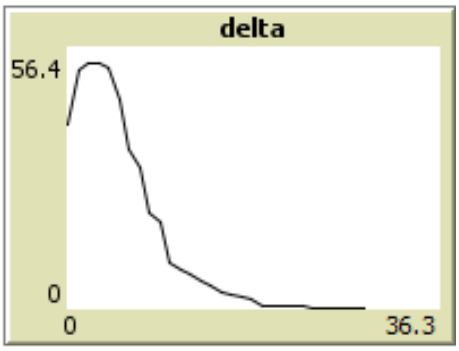

Figure 11: Equilibrium plot

After several iterations, the delta plotting stops, thus we can run the model now:

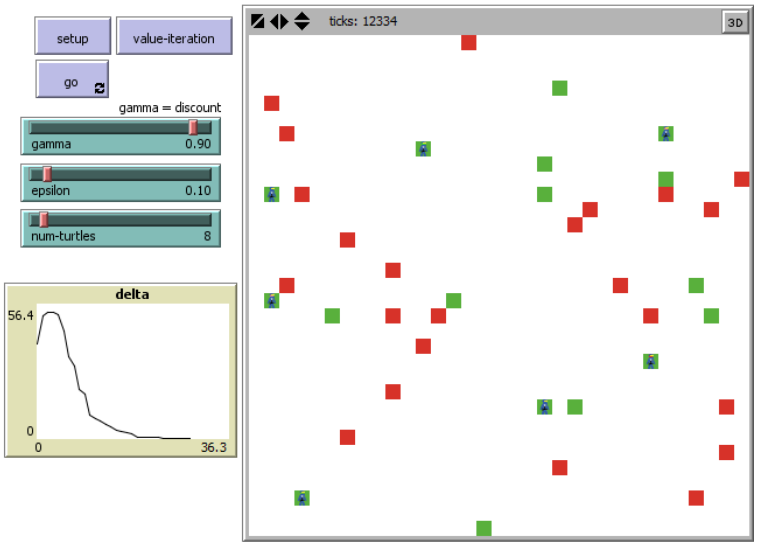

Figure 12: Optimal policy implemented

If we look at both Figures 10 and 12, we can see all agents in Figure 10 have moved to their closest crime scene in Figure 12. So the logic clearly works.

Now we demonstrate with a single officer:

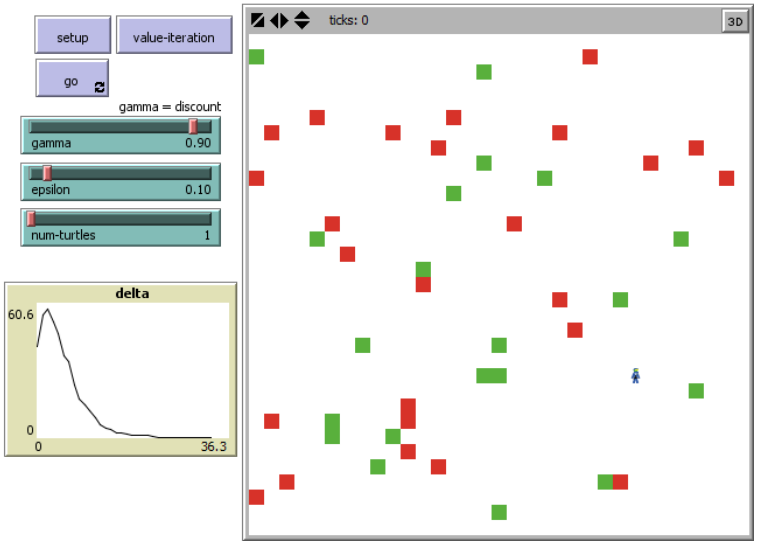

Figure 13: Single Officer example

In Figure 13, we depict a single Officer with two crimes occurring in close proximity, according to the policy, let us see which crime it moves to:

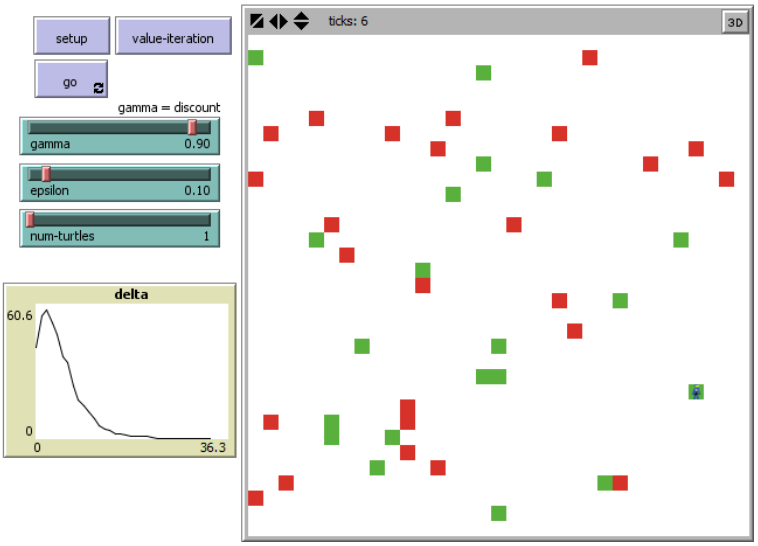

Figure 14: Single Officer crime diffusion

According to Figure 14, it diffuses the crime that was a patch closer to it on the right.

References

[1] Uri Wilensky and William Rand. *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT Press, 2015.