University of Sydney

COMP5703

Capstone Project Individual

Decision Making over Street Networks

Author:
Handuo Mei

Student Number:
470471467

June 21, 2018

DECLARATION

SIGNED: ..Handuo Mei..................... DATE: .....June 21, 2018...................

Contents

Abstract

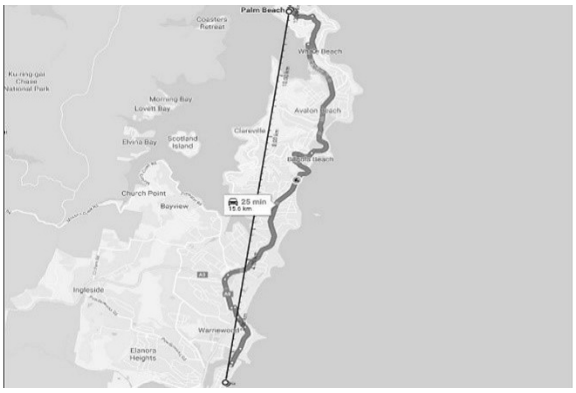
In order to provide better decision making of the street networks, the Euclidean distance is not sufficient, the actual car driving on the road is following street network driving route not Euclidean distance. An example has shown the difference between the Euclidean distance and street network distance can be significant. (In introduction) The existing model is using Euclidean distance as input for decision making which can be improved by replacing with Street Network distance. This project is to find out a solution to apply street networks distance over existing model to improve the decision making. Monte Carlo Tree Search (MCTS) is one algorithm that can be applied on the above scene. This solution can be used by the police cars to make effective and efficient decisions over street networks, specifically, a car equipped with MCTS algorithm will have such a patrolling plan that the car will frequently visit the higher crime density areas and explore more areas on the map. The results of this report can contributes to the police car discovering the crime earlier and thus prevent the crime in advance.

1 Introduction

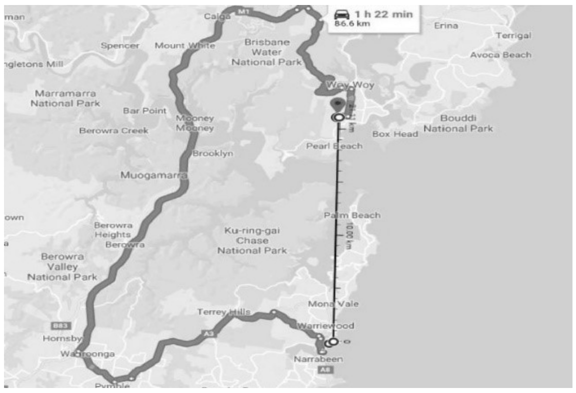
Crime is a major threat to the society. Some hot spots or areas have much higher chance to have a new crime happened than other areas. The crime is highly related to the local demographical statistics such as population, education and income level, and many other factors. Fortunately, the police generally have the historical data of crime and demographical data in most of the cases. Therefore, we can apply the statistical models to estimate the crime for future, based on the observed data. Once we have the estimations of crimes for future, we then can reallocate the police patrolling to increase the chance that discovering the crime earlier and thus prevent the crime in advance.

This report intend to investigate how to effectively and efficiently patrol on the map within a specified time period, given the estimations of crimes. Specifically, we defined an effective patrolling plan for a police car is the one can maximizing the number of different areas visited, while the efficiency means maximizing the number of visits for areas which have higher possibilities of crime occurrence, within a specified time period. For instance, if a police car visited many different areas but have less number of visits on high crime density areas, we will say it is effective but not efficient. On the other hand, if another police car that stayed on a high crime density area and never move, then we will say it is efficient but not effective.

This report formulated the above case as a reinforcement learning problem, since



(a) Palm Beach to Narrabeen Beach



(b) Palm Beach to Umina Beach

it involved the interaction between the police car and the street network, as well as the study about the trade-off between exploration and exploitation. Reinforcement learning explicitly focused on goal-directed learning from interaction[1], and this report primarily used terminologies and methods from reinforcement learning (see background section for more details on reinforcement learning). Formally, we referred the police car as the agent, estimated crime density as the environment, nodes as the states, edges as the actions, and the simulated crime density at each nodes is the reward giving to the agent. These terminologies used interchangeably in this report. Every time the police car arrived at a node on the street network, it will receive a numerical reward which depends on the magnitude of estimated crime density at that node; the larger the crime density , the higher reward the car will receive, and vice versa. Therefore, the goal of this report can be described as, to find out a patrolling plan, which can maximize the total reward received during the patrolling. In the methodology section, we will present and compare several different algorithms which addressing the above circumstance. Additionally, this report simulated the estimated crime density by using the superposition function which will be introduced later in the methodology section, as we do not have the sophisticated estimation outcomes of crime density.

There is an existing model approximating decision making under uncertainty in Euclidean space. The Euclidean distance give OK approximation in most of cases, however there are a few cases which will not provide accurate approximation.

For example, the Euclidean distance and the real-world street driving distance between Umina beach, Palm Beach and Mona vale are shown in a table 1.

This example shows in terms of similarity if apply Euclidean distance Umina beach is closer to Palm beach compared to Narrabeen Beach. The reality there are 8 times difference between them from Figure 1a and Figure 1b. This mismatch between

Table 1: Compare Euclidean and network street driving distance

From - To	Euclidean Distance	Street driving distance
Palm Beach - Narrabeen Beach	12.1 km	15.6 km
Palm Beach - Umina beach	9.3 km	97.5 km

Euclidean distance and street network drive distance will lead to incorrect decision making. The project is to address this issue by applying street network distance on the model to improve the decision making.

2 Aim

The aim of this project is to investigate and develop the feasible algorithms to effectively and efficiently allocate a police car, given the mapping of estimated crime density, to increase the chance of discovering the crimes earlier and thus preventing crimes in advance.

This report used two statistics to measure the effectiveness and efficiency of the algorithms; the exploration rate indicates the proportion of map which have been explored by the car and it measures the effectiveness, and the total rewards collected by the car indicates the efficiency of the algorithm. The project will be completed by comparing exploration rate and total reward collected between five police car agents, that equipped with $random^{[1]}$, $random^{[2]}$, and three MCTS agents but parameterized by different value of C. We defined the best algorithm as, the one that can maximize the total rewards and explore most areas on the street network.

3 Background/Related literature

3.1 Reinforcement learning

Reinforcement learning is learning what to do, how to map situations to actions,so as to maximize a numerical reward signal[1]. An ordinary reinforcement learning system involves these sub-elements, which are, agent, environment, state, action, policy, reward signal, value function, and optionally a model of environment. The **Agent** in reinforcement learning is an object who must have a goal and will learn the way of how to behave and make better decisions to achieve its goal. The agent interacts with the **environment**. Environment is defined as, anything that outside the agent but has interactions with it. Most importantly, the agent can only sense the environment but not be able to modify it. The environment responding to the actions taken by the agent and assign it a numerical rewards. **State** is the representation of environment, and we denoted it as S. The **Policy** defines the way of behaving of an agent, and it denoted as π in this report. The policy is a function that maps the states to **actions**, and tells the agent which action to be taken next. Moreover, it is the core of a reinforcement learning agent [1]. In most of the reinforcement learning system, the goal for an agent is to learn a policy. At every time the agent made an action, it will receive a immediately numerical **Reward** which tells it how good or bad the action taken before. **Value function** determines what is good for an agent over the long run, and indicates the value of the states, which is the expected total reward can be collected by the agent over the following steps, starting from that state. **A model of environment** defines the behavior of environment, and predict the probabilities of the resultant next state after taking different actions. This report only considered the case where the next states are deterministic, which means the agent knows the resultant following states for each action. The goal of a police car agent is said to be, to maximize the total reward collected over the long run.

In our case, the police car (agent) will interact with the street network (environment) which comes with the estimated crime density. The car will receive the corresponding information at each node (state), which includes the spatial location, the estimated crime density, and the neighbor nodes. Since this report treat the street nodes as the states, and in order to distinguish the different between the node index i and the time step t, we used the s_i^t to denotes the node i at the time step t. Suppose the police car is at the node s_i^t at time t, the car requires to select a neighbor node (action) from the neighbor nodes set, $\mathcal{A}(s_i)$, to visit at time $t+1$. Once it made the decision and traveled to the node s_j^{t+1}, it will then receives a reward r_j^{t+1} which indicates how good was that action. The Monte-Carlo Tree Search(MCTS), which will be briefly introduced in the methodology section, is one of the efficient and effective method for solve a reinforcement learning system. This report used MCTS to find the optimal patrolling plan for the police car.

Current model [2] is a fully probabilistic model consider spatial correlation to predict the crime count. Currently the model is using Euclidean distance. There are also existing Machine learning algorithms for predicting crime. Predpol, for example, is a Crime Prediction Algorithm, which use self-exciting point process [3] models to replicate the crime behavior. Self-exciting point process models the crime because certain crime behavior is highly clustered and sequential which is similar as earthquakes clustering pattern, near the location of initial earthquake, the risk of subsequent earthquakes, or aftershocks will increase. HunchLab[4] used both self-

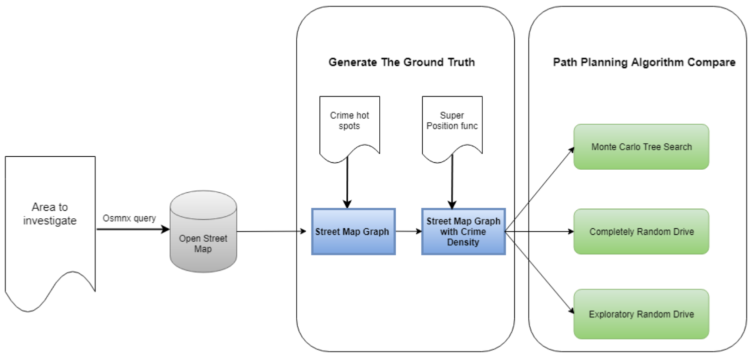


Figure 2: Project Architecture

exciting point process and Risk Terrain Modelling algorithm, the primary model used is stochastic gradient boosting machine(GBM) Both Predpol and HunchLab will forecast will occur in a given space-time The Street Networks in the project will be generated from OpenStreetMap [5], OpenStreetMap is free of charge and provide topological data structure, the Nodes and Ways in OpenStreetMap will be as input for navigation OSMnx [6] is a Python package for StreetNetworks, it can query the OpenStreetMap data and construct the route required in this project.

4 Methodology

4.1 Overview

The architecture of this project primarily involved three stages, and the Figure 2 illustrated the relationships between these three stages. Stage one will be querying the OpenStreetMap [5] using Osmnx package in python[6]. The OpenStreetMap takes input such as an area name or a set of coordinate system, and it outputs a directed graph class with edges and nodes. Figure 3 is an example of the Street Networks in Strathfield. Stage two is to generate the ground truth crime density through a superposition function, which takes the hots pots locations as well as the magnitude and the influential radial of each hots pots as the inputs, and outputs a python pandas data frame. The output data frame records the information of each nodes, including the generated ground truth crime density for each node. Moreover, this will be used as the environment for our reinforcement learning agent. At stage three, several path planning algorithm have been investigated and developed, and their performance will be compared later in the result section.

Figure 3: 5km area in Strathfield

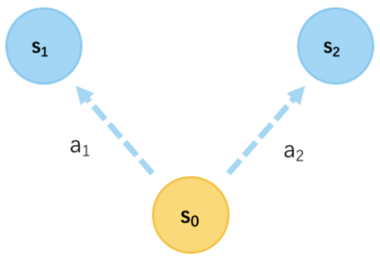


Figure 4: Illustration of states and actions

4.2 Solutions for Patrolling Plan

This section will present several algorithms which addressing to the patrolling plan, each of them will be briefly explained in the following sub sections. The figure 6 shows an toy example of states (node) and actions (edges). The node s_0 has two edges connected, a_1 and a_2, and these edges lead to the next state s_1 and s_2 correspondingly. The edge a_2 indicates the actual street is one way, while a_1 indicates a two way road. A full description of the notations used in this report can be found in the appendix section.

4.2.1 Random Algorithm

The random algorithm is the simplest algorithm for a police car to patrol on the street. Suppose the car is at the state s_i, it just uniformly selects the next node to visit from the its neighbor nodes set $\mathcal{A}(s_i)$.

$$a_i \sim uniform(\mathcal{A}(s_i))$$

We denoted this algorithm as $random^{[1]}$, and set it as the baseline patrolling algorithm. This report also developed a slightly improved random algorithm, denoted as $random^{[2]}$, which has the mechanism that encourages the police car to explore more areas, rather than drives completely random. Specifically, $random^{[2]}$ algorithm records the total number of visit for each node, and each time it selects the next node according to the probabilities calculated from the following function.

$$p(a_j(s_i)) = \frac{\frac{1}{N_j}}{\sum_{l=1}^{n} \frac{1}{N_l}}$$

This function indicates the probability of select the action j at state i. The N_j is the total number of visit for the node j, while the summation term is the total number of visit for all neighbor nodes. For instance, consider the diagram in figure 6, the probability of go to state s_1 is:

$$p(a_1(s_0)) = \frac{\frac{1}{N_1}}{\frac{1}{N_1} + \frac{1}{N_2}}$$

Suppose the number of visit for node s_1 and s_2 are 1 and 5 correspondingly, then the probability of go the node s_1 is $p(a_1(s_0)) \approx 91\%$. As the result, a car followed the $random^{[2]}$ algorithm will has higher chance to select a less visited neighbor node to visit than other frequent visited neighbors. Therefore, the car is likely to explore more areas on the map.

4.2.2 Monte Carlo Tree Search(MCTS)

Monte Carlo Tree Search (MCTS) is a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results. There are 4 steps involved in one iteration of MCTS, which are: Selection, Expansion, Simulation and Back-propagation. MCTS has been widely used in the domains that can be represented as trees of sequential decisions, such as board games and planning problems. MCTS is computational feasible and can be used to estimate the state value and learn the way of behavior for any problem that can be organized as a tree structure. The game of Go is a traditionally Chinese board game which has approximately 250^150 possible sequence of moves, and the exhaustive search is impossible for such enormous search space. In 2016, the researchers from Google DeepMind [7] proposed a Go program namely, AlphaGo,

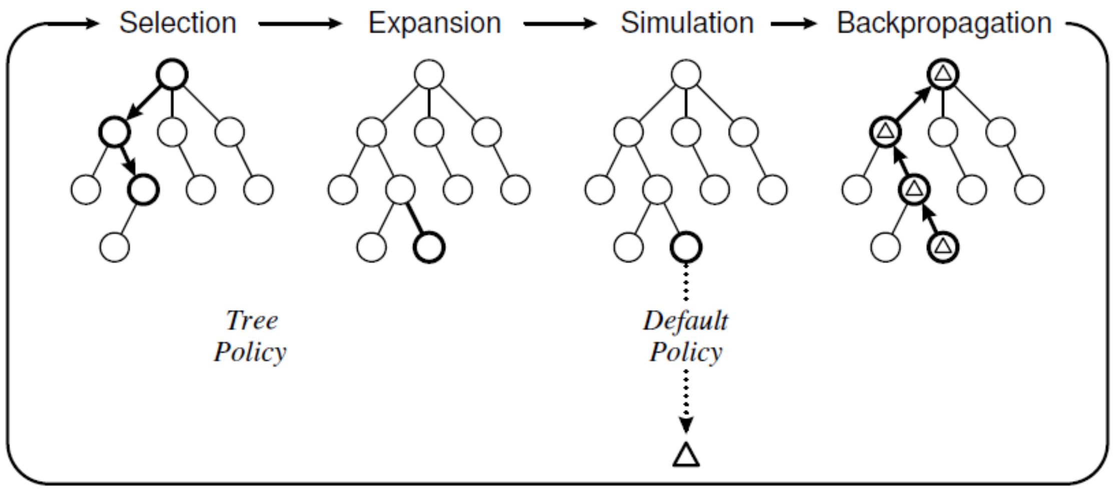
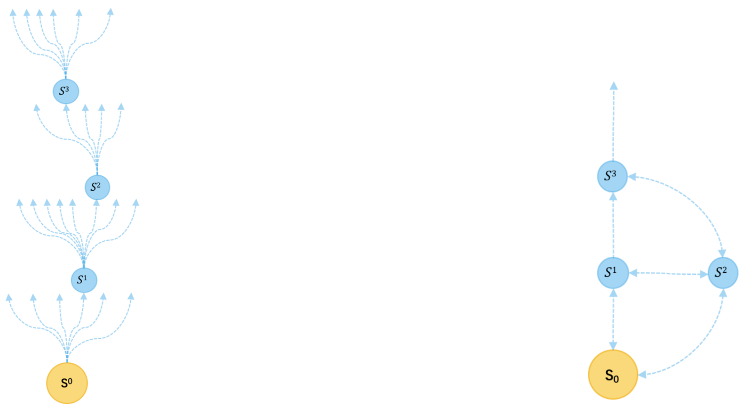


Figure 5: The image previewing how Monte-Carlo Tree Search Algorithm work, paper [10]

which primarily combined the MCTS and deep Convolutional Neural Networks, and defeated the human European Go champion by 5 to 0. One year later, the same team proposed another version of Go program named AlphaGo Zero, which not surprisingly defeated the human world Go champion, and winning 100-0 against the previously published, champion-defeating AlphaGo[8]. These examples indicates that the MCTS is a statistical anytime algorithm for which more computing power generally leads to better performance[9].

MCTS generally grows based on two policy, a tree policy and a default policy. A tree policy conduct the selection and expansion process with the consideration of exploration and exploitation, and it intends to select the most urgent and valuable node to discover next. The tree will then runs a sequence of simulation from the selected urgent node, and the moves in the simulation process were guided by the default policy. In our project, we used a random rollouts policy which randomly moves d steps ahead. Once the simulation over and arrived at the terminal state s_T, the agent will received rewards r. The back propagation process involves back propagate the reward to all the nodes which have been visited and update the corresponding number of visit for each node. The figure 7 illustrates a general case of Monte Carlo Tree Search.

The sadness is that a general MCTS algorithm is not suitable for the case that the police car patrolling on the street network. Unlike most of the case where MCTS succeed, that can have a clear tree built by the algorithm from top to bottom to indicates the transition of states, there are lots of circles exist in a street network



(a) A typical state tree (b) State tree for the street network

Figure 6: The difficulties of applying MCTS on street work

graph. Figure 8 shows the structure of the state trees for a general sequential decision case and for a street network.

The figure 8.a shows a state tree for a general sequential decision making case, the agent moved to a new state after taking an action, and most importantly, the possible states left are not identical to the states that have been visited in the past. Therefore, such cases generally has a terminal state. Figure 8.b shows the state tree for the case of decision making over street networks. We can see that there exists circles in the tree structure. If started at s_0 and moved to s_1, it then can have the choice that moves backward to the s_0. According to this property, we can conclude that the future states space can be infinite for the agent and there is no terminate state exist in the tree.

In order to fix the above issues, we firstly defined d to be the number of time steps that rollouts from the state s_i, while the last node that visited in this simulation is defined to be the terminal state. Secondly, we restricted the car and not allowed it to travel back to the node that has been visited at one time step in the past. For example, if a car at the node s_0 at time t, and moves to s_1 at time $t+1$, then it is not allowed to move back to the state s_0 at time $t+2$. This constraint ensures the car to not travel between two nodes. Lastly and most importantly, we used the Upper Confidence Bound for Tree (UCT) to regularized the car to not getting stuck at some small areas, even there are plenty of rewards. The UCT also balanced the trade-off between exploitation and exploration in reinforcement learning system. The UCT score of state s_i is defined as:

$$UCT(s_i) = v(s_i) + C\sqrt{\frac{\log \sum_{j=1}^{n} N_j}{N_i}} \tag{1}$$

The N_i is the total number of visit for state s_i, and the $v(s_i)$ is the state value of s_i. The state values $v(s_i)$ indicates the expected total return can be received by the agent over the next d time steps started from state s_i. The state values were estimated by MCTS, and it is defined as:

$$v(s_i) = \sum_{d=1}^{d} \gamma^d r^{[d]}$$

γ is a parameter that controls the importance of the reward received at each time step t. This report explicitly used the gamma of 1.1 for MCTS algorithm, to increase the importance of rewards received in future. The reward given to the car at the node s_i was sample from a normal distribution:

$$r_i^{[t]} \sim \mathcal{N}(\mu_i, \epsilon)$$

where the μ_i is the crime density at node i and the epsilon is the noisy added to the reward artificially. This report used epsilon of 0.2 when simulate the reward. We then back propagate the state value according to the averaged reward over the sequence of simulations. Moreover, this report used the incremental implementation for updating the state value $v(s_i^t)$ at time t, to save the memory and to have constant per-time-step computation cost[1]. The incremental implementation updating rule stated as follow:

$$v(s_i^t) = v(s_i^t) + \frac{1}{N_i}\left[v(s_i^{t+1}) - v(s_i^t)\right]$$

Every time the police car needs to make the decisions about which nodes to move at next time step, it firstly calculates the UCT score for each of the neighbor, then selects and moves to the neighbor node according to the probabilities, that calculated as follow:

$$p(s_i) = Softmax(UCT(s_i)) = \frac{\exp^{UCT(s_i)}}{\sum_{j=1}^{l} \exp^{UCT(s_j)}}$$

A softmax function was applied to the UCT scores, in order to be probabilistic when selecting actions. Suppose the state s_i has l neighbors, and there is one neighbor has UCT score larger than others, then it will has higher chance to be selected due to the higher probability output from softmax function.

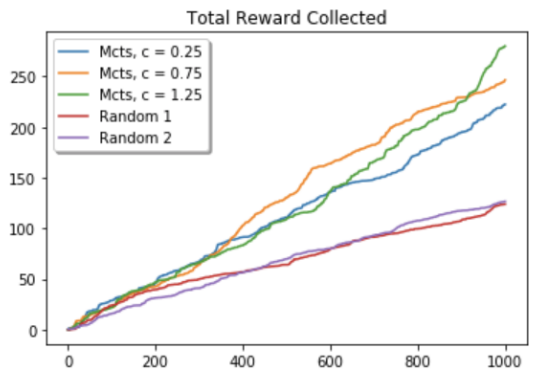
The UCT score is the summation of two terms, while the left term $v(s_i)$ encourages the exploitation of higher-return choices, and the right term is to encourages the exploration of less visited choices. C is a tunable exploration parameter, and the default choice suggested by the researchers is $1/\sqrt{2}$, however, in practice the C usually chosen empirically. MCTS will randomly sample the search space and growing the search tree based on the returned randomly simulation result. The decision over Street Networks can be viewed as a planning problem, the trees growing with sequential decisions. Moreover, computational budget is set to constrain the how many times the search tree should be iteratively built.

5 Results

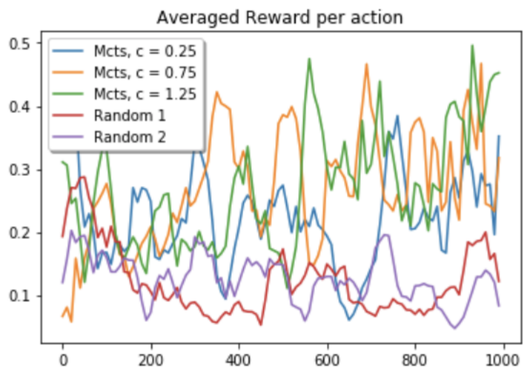
The output from stage three including the statistics of total reward collected, averaged reward received per 50 iteration, and the exploration rate during 1000 iterations for different path planning algorithms. The object is that within the same number of iteration, the agents followed MCTS algorithm are expected to have more total reward compared to the random walked agents.

Graph 7a indicates that the Monte Carlo Tree Search algorithms parameterized by different values of C have outperformed the algorithms *random*[1] and the explorative *random*[2] in total reward collected. The best agent that has the most reward collected within 1000 time steps, is the MCTS with the exploration parameter C of 1.25. Recall that the parameter C controls the exploration ad exploitation of MCTS algorithm, and higher value the C generally leads to more explorations. Our results also indicates that the more exploration leads to have higher chance to discover the higher reward area, in our case, the high crime density area. Therefore, higher exploration parameter leads to higher total rewards collected. Moreover, the best MCTS agent has more than two times of total reward collected compare to the random drive agents. Graph 7b on the right shows the averaged reward of each action per 50 iterations. As we can see that, the averaged rewards for MCTS agents steadily growth over the time, while the random agents always have lower reward collected at each iteration. The MCTS with the exploration parameter of 1.25 seems to have significantly higher averaged reward after about 500 iterations.

Figure 8 shows exploration rate achieved by the MCTS agents and random drive



(a) Total Reward received



(b) Average Reward per 50 iterations

Figure 7: Reward comparisons between different algorithms

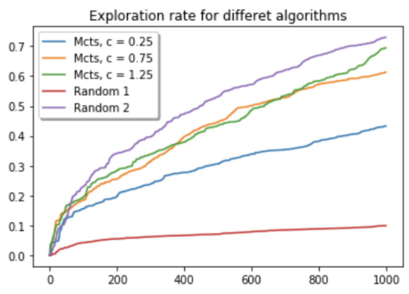


Figure 8: Exploration rate at each iteration

agents at each of the 1000 iterations. Not surprisingly, the MCTS agent with higher value of C achieved the second highest exploration rate over the 1000 iterations, while the exploration algorithm *random*[2] achieved the highest, since the natural of its exploration property.

6 Conclusion and Discussion

In this report, we developed several methods to simulate how a police car should patrol over the street networks to discover the most urgent areas that are likely to have a new crime, given the estimated crime density surface.

To achieve our goal, firstly we specified an interested area by creating a street networks map from existing open street map API. With extracted street networks map the crime density surface map being created. The surface map is constructed by combining the crime hot spots and a superposition function. It provides an overall accumulated crime density surface map. The impact from the hot spots is decayed over the distance. The surface map including digraph of nodes and edges, the nodes represent the actual street intersections, while the edges represent the roads between intersection. A pandas Data frame is used to store the crime density for each node.

Then overall 5 different police patrolling algorithms were fitted over 5 different cars, and tested in a same simulated environment. 2 cars were equipped with modified random algorithms, and these random walk algorithm collect much less reward compared to the other three cars that equipped with Monte Carlo Tree Search(MCTS) algorithm. The reasons are, instead of randomly selecting next move, the decision for MCTS is based on many simulations. As stated in [9] MCTS will pick a promising sequence of moves and run simulations of how the reward will be collected. This has been the true value of an action is approximated using random simulation, and these values may be used efficiently to adjust the policy towards a best-first strategy. The algorithm progressively builds a partial tree, guided by the results of previous exploration of that tree. The tree is used to estimate the state value for each node, with these estimates becoming more accurate as the tree is built.

In conclusion, the MCTS agent with parameter C of 1.25 can be considered as our best agent, which achieved the highest total rewards while having the second highest exploration rate. Moreover, we can conclude that, the MCTS agents have learned the way of behavior in order to collect more rewards in the long run, and we conclude such algorithms can discover the most urgent areas on the map, given the estimation of crime density.

7 Limitations and future work

There are few parts that can be improved from current version of implementations.

The estimated crime density surface in current version was simulated from randomly generated hot spots, however in real world the distributions of crime density are much more complex. For example, from Bower's [10] finding, 80% of theft related crime happened in 20% of the facilities, to improve the model accuracy need to incorporate the real crime data to current model.

In terms of crime hot spots, current model only considered the spatial, for case like the higher crime rate at night near the bar to be addressed the temporal constraint theory is required. This theory [11] provides a better understand of the crime pattern shit. Future model can be further improved by incorporating time parameters.

The current version only considered the situations that only one police car patrolling on the street, however in real world, the number of police vehicles are usually more than one. Therefore, how do the cars collaborate with each other to better discover and prevent the crimes is more complex than only one police car, and this can be considered as a multi-agent reinforcement learning task.

Last but not the least, due to the complex of street network structure, even we increased the depth of search d, the car equipped by current algorithms can also be described as myopic, since it is very unlikely that the car can sense the crime density on the other side of the map. In order to fix the myopic problem, we are currently working on the a Higher Level Path Planing method. We also referred the algorithms developed in this report as Lower Level Path Planing method, since it is more efficient to traverse within a smaller region but not farsighted enough to look beyond the local area. The Higher Level Path Planing method includes the spatial distance into consideration, and calculated another UCT score for bigger squares areas.

$$UCT_a = \bar{V}_a + C\sqrt{\frac{2\log(\sum_{j=1}^{n} N_j)}{N_a}} + \delta\mathcal{D}(\ell_c, \ell_a)$$

Unlike the UCT score stated in the previous section which calculates the state value for each neighbor node, the function above is actually calculates the UCT score for much bigger spatial areas, figure 9 illustrated the difference between the street network nodes and bigger areas. The subscript a indicates one of the bigger square area shown on the right of figure 9, and $\bar{V}_a$ is the averaged state value of all nodes within the square area a. The mid term of this function is identical to the term in a general UCT function, and the term on the right is a new introduced term which controls the attraction of farther areas. $\mathcal{D}$ is a function to calculate the distance between the current location ℓ_c and the location of area ℓ_a. Either Euclidean distance or street network distance would be suitable for such calculation. At each time step, the car not only consider which neighbor node to visit at next time step, but also which area to be visited at next. The area can also defined as the SA1, or SA2 area suppose the car is in Australia. We referred the above method as one possible solution to make the higher level planning, and let the car become farsighted.

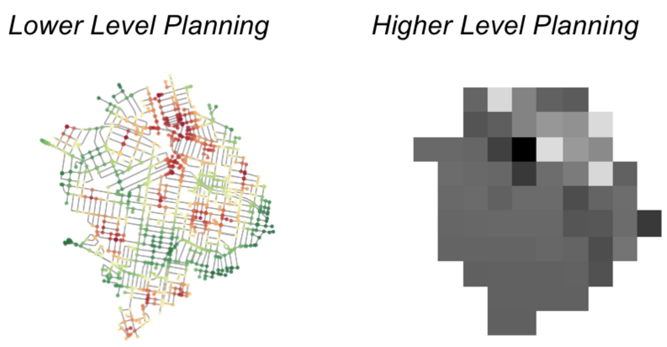


Figure 9: Exploration rate at each iteration

Nomenclature

$\mathcal{A}(s_i)$	set of all possible actions in state s_i (connected neighbor nodes)
$\mathcal{G}$	a street network obtained from Open Street Map site
a	action (edges on the street)
$a(s_i^t)$	action selected at the node i at the time step t
$a_*(s_i^t)$	optimal action at state i at time step t, which leads to the optimal next state
mu_i	the ground truth crime density at node i
N_i	total number of visit at state i
r	reward (simulated crime density)
$r_i^{[t]}$	reward to the agent at time t at the node i
s	state(nodes on the street network)
s_i	state i
s_i^t	state i at time step t. E.g s_9^5 indicates the car is at the node 9 at the time step 5
t	discrete time step
v	value, which is the expected return
$v(s_i)$	value of state i, which is the expected return could be received if starts from s_i

$v_*(s_i)$	the optimal value of state i, which is the highest expected return could be received if starts from s_i

References

[1] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[2] Marchant Matus R, Haan S, Clancey G, and Cripps S. Applying machine learning to criminology: Semi-parametric spatial-demographic bayesian regression. ready for submission, 2017.

[3] Matthew S Gerber. Predicting crime using twitter and kernel density estimation. *Decision Support Systems*, 61:115–125, 2014.

[4] Azavea. Hunchlab under the hood. Available at `https://cdn.azavea.com/pdfs/hunchlab/HunchLab-Under-the-Hood.pdf`(2015).

[5] OpenStreetMap. openstreetmap.org. Available at `https://openstreetmap.org`(2018).

[6] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017.

[7] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[8] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[9] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[10] Kate Bowers. Risky facilities: crime radiators or crime absorbers? a comparison of internal and external levels of theft. *Journal of Quantitative Criminology*, 30(3):389–414, 2014.

[11] Jerry H Ratcliffe. A temporal constraint theory to explain opportunity-based spatial offending patterns. *Journal of Research in Crime and Delinquency*, 43(3):261–291, 2006.