

## Singapore Management University Institutional Knowledge at Singapore Management University

---

Research Collection School Of Information Systems

School of Information Systems

---

11-2011

# A POMDP model for guiding taxi cruising in a congested urban city

Lucas AGUSSURJA


*National University of Singapore*

Hoong Chuin LAU

*Singapore Management University, [hclau@smu.edu.sg](mailto:hclau@smu.edu.sg)*

**DOI:** [https://doi.org/10.1007/978-3-642-25324-9\\_36](https://doi.org/10.1007/978-3-642-25324-9_36)

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Artificial Intelligence and Robotics Commons](#), [Operations Research, Systems Engineering and Industrial Engineering Commons](#), and the [Transportation Commons](#)

---

### Citation

AGUSSURJA, Lucas and LAU, Hoong Chuin. A POMDP model for guiding taxi cruising in a congested urban city. (2011). *Advances in Artificial Intelligence: 10th Mexican International Conference on Artificial Intelligence, MICAI 2011, Puebla, Mexico, November 26 - December 4: Proceedings*. 7094, 415-428. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/1385](https://ink.library.smu.edu.sg/sis_research/1385)

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

# A POMDP Model for Guiding Taxi Cruising in a Congested Urban City

Lucas Agussurja<sup>1</sup> and Hoong Chuin Lau<sup>2</sup>

<sup>1</sup> The Logistics Institute Asia Pacific, National University of Singapore  
21 Heng Mui Keng Terrace #04-01, Singapore 119613

[tlila@nus.edu.sg](mailto:tlila@nus.edu.sg)

<sup>2</sup> School of Information Systems, Singapore Management University  
80 Stamford Road, Singapore 178902

[hclau@smu.edu.sg](mailto:hclau@smu.edu.sg)

**Abstract.** We consider a partially observable Markov decision process (POMDP) model for improving a taxi agent cruising decision in a congested urban city. Using real-world data provided by a large taxi company in Singapore as a guide, we derive the state transition function of the POMDP. Specifically, we model the cruising behavior of the drivers as continuous-time Markov chains. We then apply dynamic programming algorithm for finding the optimal policy of the driver agent. Using a simulation, we show that this policy is significantly better than a greedy policy in congested road network.

**Keywords:** agent application, intelligent transportation, POMDP, taxi service.

## 1 Introduction

Taxis are a major mode of transport in every urban city in the world. There were about 24,000 taxis in Singapore, and 87,000 licensed taxi drivers, and taxis provide about 850,000 trips daily, as of April 2009. In Singapore, taxis are operated by a small number of taxi companies. The largest taxi company, ComfortDelgro, operates over 15,000 taxis, and captures about 63% of the market share. Like many congested cities in the world, people in Singapore often view taxis as a more efficient mode of transportation compared to private cars. Statistics released by the Singapore Land Transport Authority shows that a taxi travels some 120,000 km a year, and more than a third of such travel is empty cruising, which represents a wastage of resources.

Although research related to taxi services can be found in the economics, transportation, and operation research literature, studies from the AI point of view, however, has been lacking to our knowledge. This is an opportunity for applied AI research, since the system resulting from the collective behavior of the drivers can be readily modeled as a multi-agent system, with drivers acting as rational agents towards maximizing their individual utilities. We believe that the public transport arena offers a rich domain for application of AI concepts and methodologies.

In this paper, we consider the specific problem of modeling and guiding taxi driver cruising behavior in a congested city consisting of a large number of taxis and passengers. This work is made possible by a large dataset obtained from a large taxi operator. We model, for simplicity, the network as a graph where nodes represent zones while arcs represent adjacency. Passengers “arrive” in each zone as a Poisson process, while taxis are either in the state of **occupied** or **cruising** (when no passenger is onboard and the driver is actively seeking passengers, whether moving about or resting at a location). We are concerned with improving the overall utilization rate by offering intelligent guidance on its cruising behavior (i.e. deciding when and where to move from zone to zone considering the passenger arrival processes as well as in response to the behavior of the other drivers).

We propose a POMDP to model decision process of a taxi driver during cruising. Our purpose in this paper is not to invent a new algorithm for solving POMDP. Rather, motivated by the vast recent literature in AI on solving large-scale POMDPs (see Related Works below), we choose POMDP as the modeling framework for our problem, and present in this paper how such model is built, and experimental results obtained. The proposed model can be embedded into a vehicular device that guides the taxi driver to move intelligently within a highly dynamic environment (such as in a congested city) that will maximize occupancy and hence revenue.

## 2 Preliminaries

Our system consists of:

1. A directed graph  $G = (V, E)$ , representing the road network on which the taxis operate. Each node in the graph is associated with a zone. A taxi may move from one zone to the other following the edge linking the zones. We assume that the graph is connected.
2. A set of  $n$  drivers indexed by  $i \in \{1, \dots, n\}$ . The cruising behavior of a driver governs its preference in choosing which zone to cruise in. In this paper, we assume that each driver cruising behavior is independent of the others.
3. The agent driver under consideration.

The objective in this paper is to compute the best cruising policy (response) of the agent, one that maximizes the time that the agent is occupied, given the cruising behavior of the other drivers and a finite period (the planning horizon). We make the assumption that all the drivers and the agent are always in operation during the planning horizon. The drivers and agent constantly transit from the cruising (the term is used interchangeably with “unoccupied”) state to occupied state and back to cruising state.

A partially observable Markov decision process for a single agent can be described as a tuple  $\langle S, A, T, R, \Omega, O \rangle$ , where:

- $S$  is a finite set of states of the system,
- $A$  is a finite set of actions of the agent,

- $T : S \times A \rightarrow \Delta(S)$  is the *state transition function*, where we write  $T(s, a, s')$  for the probability of ending in state  $s'$ , given that the agent is in state  $s$  and takes action  $a$ ,
- $R : S \times A \rightarrow \mathbb{R}$  is the *reward function*, where  $R(s, a)$  is the expected reward for taking action  $a$  while in state  $s$ ,
- $\Omega$  is a finite set of possible observations the agent can receive,
- $O : S \times A \rightarrow \Delta(\Omega)$  is the *observation function*, where we write  $O(s, a, o)$  for the probability of making observation  $o$  after the agent took action  $a$  in the current state  $s$ .

The agent maintains an internal *belief state*  $b \in \Delta(S)$  that described its belief of the current state of the system based on its previous observations, actions and the initial belief state  $b^0$ . A  $t$ -step policy tree is a perfect  $|\Omega|$ -ary tree of depth  $t$  describing completely the agent’s possible actions for the next  $t$  steps. Each node in the tree specifies an action with the root as the starting action, and each branch specifies a possible observation and the corresponding child node specifies the next action to be taken.

### 3 Related Works

As an integral part of public transportation system of an urban city, taxi services have been extensively studied from the economics (see [11,14,12]), transportation and operation research (see [13,10]) point of views. In the economics literature, research has been conducted to gain insights into the nature of the demand and supply of taxi services, their interaction and the resulting market equilibrium. The economic consequences of regulatory restraints, such as entry restriction and price control, have been examined as well. In operation research literature, quantitative models have been built to capture driver’s movement behavior, passenger’s searching behavior, and the competitive nature of the drivers.

POMDP is a modeling framework used for agent planning in partially observable stochastic domains [7]. It has been used in AI robotics and planning. The majority of these studies concentrate on finding a scalable algorithm to solve the model (e.g. [3,4,6,8,9]), while some focus on looking at special cases of POMDP for which efficient algorithm might be found. Research into its multi-agent counterpart have become active in recent years, both in competitive setting [5] and non-competitive setting ([1,2]).

### 4 POMDP Model for Single Cruising Agent

In this section, we present our model for a taxi cruising decision process. First and foremost, it should be noted that two models are proposed for cruising behavior. For the agent driver under consideration, we use a POMDP to model its decision process from which we derive its optimal cruising behavior. The behavior of each of the other drivers in the system is modeled by the continuous time Markov chain (see Section 4.2). In this sense, the behavior of these drivers

are "static" in that they do not change in reaction to the behavior of the agent or the other drivers. A more dynamic model would be the partially observable stochastic game (POSG), where each driver is modeled as a POMDP. Given the current state of research [5], POSG is much more complicated and computationally inhibiting in our context where we are interested to model a large number of drivers. Hence in this work, we focus on a single POMDP. Henceforth, to avoid confusion, we will use the term "driver" to refer specifically to a driver who is not the agent under consideration.

#### 4.1 System States

At any point of time, there are  $n$  taxis operating (excluding the agent considered). A state of the system consists of the state of the agent and each of these taxis. Specifically, a state of the system is described by a tuple  $s = \langle \omega, L, D, (\delta_i), (l_i), (h_i), (d_i) \rangle$ , with  $1 \leq i \leq n$ , where:

- $\omega = 1$  if the agent's taxi is occupied, and  $\omega = 0$  otherwise (i.e. cruising),
- $L \in V$  is the current location zone of the agent,
- $D \in V \cup \{0\}$  is the destination zone of the passenger if the agent's taxi is occupied,  $D = 0$  otherwise,

and for each driver  $i$ :

- $\delta_i = 1$  if the taxi is occupied, and  $\delta_i = 0$  if it is currently cruising,
- $l_i \in V$  is the current location zone of the driver,
- $h_i$  contains the path history of the driver if it is occupied, that is, the sequence of zones starting with the zone from which the passenger is found to the current location zone of the driver. If the taxi is unoccupied,  $h_i$  is empty.
- $d_i \in V \cup \{0\}$  if the destination zone of the passenger if the taxi is occupied,  $d_i = 0$  otherwise.

To simplify notations, we will omit the subscripts of a variable when we consider the vector value of the corresponding variable. For example,  $l = (l_1, \dots, l_n)$ , and  $d = (d_1, \dots, d_n)$ .

#### 4.2 A Model for Driver Cruising Behavior

We model the driver cruising behavior as a continuous time Markov chain, which consists of a set of states where the transition time from one state to another is exponentially distributed. The legitimacy of exponential distribution has been verified from a real trace of taxi data of a large Singapore taxi operator for one month on the Singapore road network (see Section 6.1).

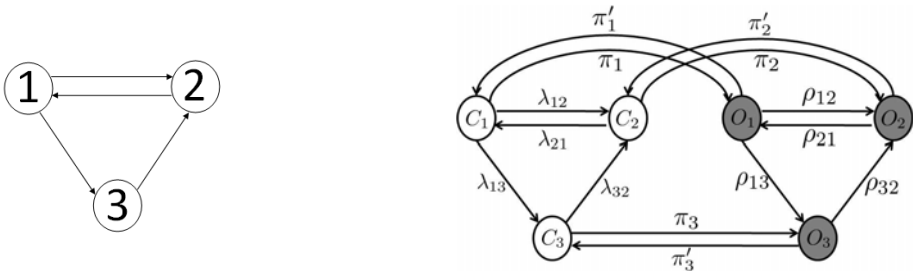
The set of all possible states of a driver is given by  $\{C_1, \dots, C_{|V|}, O_1, \dots, O_{|V|}\}$ , where state  $C_j$  ( $j \in V$ ) corresponds to the driver being in zone  $j$  and cruising for passengers; while state  $O_j$  ( $j \in V$ ) corresponds to the driver being in zone  $j$  and currently delivering a passenger to its destination. The following are the sets of parameters which govern the transitions between the states:

- $\{\lambda_{jk}\}_{j,k \in V}$ , where  $\lambda_{jk}$  is the rate of transition from state  $C_j$  to state  $C_k$ . This set of parameters describes the cruising behavior of a driver. In our model, we assume that the drivers are uniform, i.e., have the same cruising behavior.
- $\{\pi_j\}_{j \in V}$ , where  $\pi_j$  is the rate of transition from state  $C_j$  to state  $O_j$ . This set of parameters describes the rate of finding a passenger in a zone (see equation 1).
- $\{\pi'_j\}_{j \in V}$ , where  $\pi'_j$  is the rate of transition from state  $O_j$  to state  $C_j$ . This set of parameters describes the rate of finding the dropoff point of a passenger in its destination zone.
- $\{\rho_{jk}\}_{j,k \in V}$ , where  $\rho_{jk}$  is the rate of transition from state  $O_j$  to state  $O_k$ . This set of parameters describes the congestion rate of the road network.

Together, these sets of parameters constitute the  $Q$ -matrix of the Markov chain, which describes the chain completely. As mentioned previously, we assume that the drivers operate independently of each other. Each zone  $j \in V$  has a constant passenger arrival rate  $\mu_j$ . When a passenger arrives in a zone, it will be randomly picked up by one of the drivers cruising in the zone. In other words, for every  $j \in V$ , we have:

$$\pi_j = \frac{\mu_j}{(1 - \omega) \cdot \alpha(L, j) + \sum_{i=1}^n [(1 - \delta_i) \cdot \alpha(l_i, j)]}, \quad (1)$$

as the rate of finding a passenger in zone  $j$ , where  $\alpha(j, k) = 1$  if  $j = k$ , and  $\alpha(j, k) = 0$  otherwise.



**Fig. 1.** Example: The figure on the left shows a network of 3 nodes, while the figure on the right shows the corresponding Markov chain describing the behavior of a driver

An example of the Markov chain on a simple 3-node network is given in Figure 1. Two useful quantities for our modeling that can be derived from the Markov chain are: (1) The probability of leaving a state after  $t$  time has passed which is given by  $1 - \exp\{-(\sum \text{outgoing rate}) \cdot t\}$ , and (2) The probability of moving from one state to the other with transition rate  $\lambda$  which is given by:  $\lambda / \sum \text{outgoing rate}$ . In the example of Figure 1, the probability of leaving the state  $C_1$  after one unit of time has passed is  $1 - \exp\{-(\lambda_{12} + \lambda_{13} + \pi_1)\}$ , with the probability of going to state  $O_1$  given by  $\pi_1 / (\lambda_{12} + \lambda_{13} + \pi_1)$ .

### 4.3 Actions and State Transition Function

Next, we define the actions of the agent and derive the state transition function  $T$  for the POMDP. We do this by discretizing the continuous Markov chain into periods of one minutes, where each period corresponds to one step of the POMDP. The state transition function  $T$  can be separated into two independent components as follows:

$$\begin{aligned} T(s, a, s') & \tag{2} \\ &= \Pr[\langle \omega', L', D', \delta', l', h', d' \rangle | \langle \omega, L, D, \delta, l, h, d \rangle, a] \\ &= \Pr[\omega', L', D' | \omega, L, D, \delta, l, a] \cdot \Pr[\delta', l', d' | \omega, L, \delta, l, d]. \end{aligned}$$

Consider the first term of the right hand side of Equation 2. The actions available to the agent depend on the state of the agent.

**Case 1:**  $\omega = 0$ . In the unoccupied state, the agent may take one of the following actions in a step: ( $a_1$ ) continue cruising in the current zone, or ( $a_2$ ) make an attempt to move to an adjacent zone. In the unoccupied state, the agent may have the chance of getting a passenger as well. Given that  $\omega = 0$  and the agent takes action  $a_1$ , the following may happen in one step: (1) The agent doesn't find any passenger, or (2) The agent manages to find a passenger with  $k$  as the destination zone. In both cases, the location of the agent doesn't change. The following gives the probability of each of these cases respectively:

$$\Pr[0, j, 0 | 0, j, 0, \delta, l, a_1] = e^{-\pi_j},$$

$$\Pr[1, j, k | 0, j, 0, \delta, l, a_1] = (1 - e^{-\pi_j}) \cdot \gamma_{jk},$$

where  $\gamma_{jk}$  is the probability of a passenger's destination zone being in  $k$  given that its starting zone is  $j$ . On the other hand, if the agent chooses  $a_2$  and make an attempt to move to an adjacent zone  $z$ , one of the following may occur: (1) A passenger is found, and the agent stays in the current location, (2) No passenger is found, and the agent manages to move to zone  $z$ , or (3) No passenger is found and the agent stays in the current location. The following gives the probability of each of these cases respectively:

$$\Pr[1, j, k | 0, j, 0, \delta, l, a_2 \rightarrow z] = \frac{\pi_j \cdot \gamma_{jk}}{\pi_j + \rho_{jz}} \left( 1 - e^{-(\pi_j + \rho_{jz})} \right),$$

$$\Pr[0, z, 0 | 0, j, 0, \delta, l, a_2 \rightarrow z] = \frac{\rho_{jz}}{\pi_j + \rho_{jz}} \left( 1 - e^{-(\pi_j + \rho_{jz})} \right),$$

$$\Pr[0, j, 0 | 0, j, 0, \delta, l, a_2 \rightarrow z] = e^{-(\pi_j + \rho_{jz})}.$$

**Case 2:**  $\omega = 1$ . In the occupied state, the action of the agent is determined (denoted by  $a_3$ ). When it is not in the destination zone of the passenger, it will make an attempt to move to an adjacent zone along the shortest path from the current zone to the destination zone. And when it is in the destination zone of

the passenger, it will try to find the dropoff point within the zone. For the former case, the following are its success and failure probabilities respectively:

$$\Pr[1, z, k | 1, j, k, \delta, l, a_3 \rightarrow z] = 1 - e^{-\rho_{jz}} ,$$

where  $z$  is the next zone in the shortest path from  $j$  to  $k$ , consequently:

$$\Pr[1, j, k | 1, j, k, \delta, l, a_3 \rightarrow z] = e^{-\rho_{jz}} .$$

And for the later case, its success and failure probabilities are given by:

$$\Pr[0, k, 0 | 1, k, k, \delta, l, a_3 \rightarrow k] = 1 - e^{-\pi'_k} ,$$

and:

$$\Pr[1, k, k | 1, k, k, \delta, l, a_3 \rightarrow k] = e^{-\pi'_k}$$

respectively.

Consider the second term of the right hand side of Equation 2. Assuming independence among the drivers, we have:

$$\begin{aligned} & \Pr[\delta', l', d' | \omega, L, \delta, l, d] \\ &= \prod_{i=1}^n \Pr[\delta'_i, l'_i, d'_i | \delta_i, l_i, d_i, \omega, L, \delta_{-i}, l_{-i}, d_{-i}] , \end{aligned}$$

where the  $-i$  subscript denotes a vector without the  $i$ -th element. Each driver behavior is modeled as an independent Markov chain as described in Section 4.2. In the unoccupied state, in one step of the POMDP, the following may occur: (1) No passenger is found, and the driver continue to cruise in the same zone, (2) No passenger is found, and the driver moves to an adjacent zone, or (3) A passenger is found in the current zone. The following are the probability of each of these events respectively:

$$\Pr[0, j, 0 | 0, j, 0, \dots] = \exp \left\{ - \left( \sum_{z \neq j} \lambda_{jz} + \pi_j \right) \right\} ,$$

$$\Pr[0, k, 0 | 0, j, 0, \dots] = \frac{\lambda_{jk}}{\sum_{z \neq j} \lambda_{jz} + \pi_j} \left[ 1 - \exp \left\{ - \left( \sum_{z \neq j} \lambda_{jz} + \pi_j \right) \right\} \right] ,$$

$$\Pr[1, j, k | 0, j, 0, \dots] = \frac{\pi_j \cdot \gamma_{jk}}{\sum_{z \neq j} \lambda_{jz} + \pi_j} \left[ 1 - \exp \left\{ - \left( \sum_{z \neq j} \lambda_{jz} + \pi_j \right) \right\} \right] .$$

In the occupied state, the behavior of a driver is similar to that of the agent. The driver will attempt to move the the next adjacent zone in the shortest path to the destination zone. Once in the destination zone, the driver will try to find



the exact dropoff point of the passenger. The following probabilities are derived similarly to those of the agent:

$$\Pr[1, j, k | 1, j, k, \dots] = e^{-\rho_{jz}} ,$$

where  $z$  is the next zone in the shortest path from  $j$  to  $k$ , and:

$$\Pr[1, z, k | 1, j, k, \dots] = 1 - e^{-\rho_{jz}} .$$

Similarly:

$$\Pr[1, k, k | 1, k, k, \dots] = e^{-\pi'_k} ,$$

$$\Pr[0, k, 0 | 1, k, k, \dots] = 1 - e^{-\pi'_k} .$$

This completes the description for the state transition function of the POMDP. It should be noted that all other transitions that are not defined in this section have zero probability of occurring.

#### 4.4 The Complete Agent POMDP Model

We define the rest of the components of the POMDP here. The agent's reward function  $R$  is simply a binary function that return 1 if the agent is in occupied state and 0 in the unoccupied state, i.e,  $R(s, a) = 1$  if  $\omega = 1$  and  $R(s, a) = 0$  otherwise. This models the objective of the agent to be in the occupied state as long as possible. Following the data that we have for the taxi movement in Singapore, the agent may observe the state  $s$  of the system in each step of the POMDP, except for the destination zone of the passenger of the occupied drivers, that is, the agent may observe the variables  $\omega, L, D, \delta, l, h$  completely, but not the variable  $d$ . The observation function, therefore, is given by:

$$O(\langle \omega, L, D, \delta, l, h, d \rangle, a, o) = \sum_{d'} T(\langle \omega, L, D, \delta, l, h, d \rangle, a, \langle o, d' \rangle) .$$

This completes the definition of the POMDP model.

## 5 Solving the Finite Horizon POMDP Model

Next, we present our solution method to compute the optimal  $T$ -horizon policy for the POMDP model. We apply the standard dynamic programming algorithm [7] for solving general POMDP. The main difference is the representation of the belief state. Given the specific nature of the agent's observation function, we are able to represent its belief state as a set of trees (see below).

### 5.1 Agent Belief State

A belief state  $b$  is a probability distribution over  $S$ . In our case, since the agent observes the state of the system completely except for the destination zone of occupied drivers, its belief state is reduced to a probability distribution over

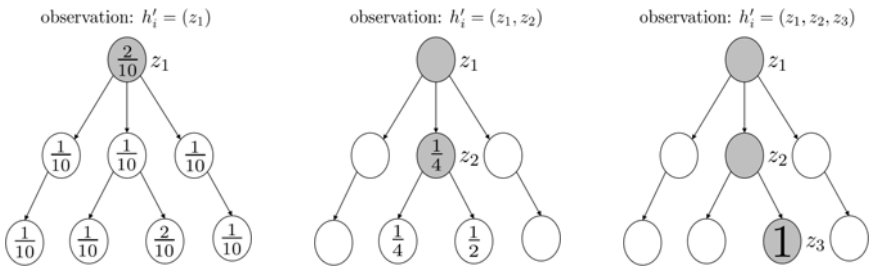
$\{V \cup \{0\}\}^n$ , which is all the possible value for vector  $d$ . We let  $b(d)$  denote the probability assigned to the destination vector  $d$  by belief state  $b$ . The agent starts with the initial belief state  $b^0$ , and in each step of the POMDP, it computes a new belief state  $b'$ , based on the old belief state  $b$ , an action  $a$ , and an observation  $o$ . The new degree of belief in a destination vector  $d'$ , can be computed as follows:

$$\begin{aligned} b'(d') &= \Pr[d'|b, a, o] \\ &= \prod_{i=1}^n \Pr[d'_i|b, a, \langle \omega', L', D', \delta', l', h' \rangle] \\ &= \prod_{i=1}^n \Pr[d'_i|b, \delta'_i, h'_i]. \end{aligned}$$

It is obvious that if  $\delta'_i = 0$ , then  $d'_i = 0$  with probability 1. For the case when  $\delta'_i = 1$ , the value of  $\Pr[d'_i|b, \delta'_i, h'_i]$  for each  $i$  is computed as follows. When there is only one element in the path history  $h'_i$ , i.e.  $h'_i = (j)$ , then for all  $k \in V$ ,  $\Pr[k|b, 1, j] = \gamma_{jk}$ . When there is more than one element in  $h'_i$ , where  $h'_i = (j, \dots, z)$ , we first construct a shortest path tree with the zone  $j$  as its root. This tree would contain all the nodes in  $V$  (assuming that the network is connected). Then, consider the subtree with  $z$  as its root, and let  $V_{j,z}$  be the set of all the zones in the subtree. For every zone  $k \in V_{j,z}$ , we have:

$$\Pr[k|b, 1, (j, \dots, z)] = \frac{b(k)}{\sum_{k' \in V_{j,z}} b(k')},$$

while the probability is zero for the rest of the zones outside the subtree. This computation process is illustrated in Figure 2.



**Fig. 2.** Example of the agent’s belief regarding the destination zone of an occupied driver, which is represented by a shortest path tree. The number in the node represents the probability of having that node be the destination zone, with empty node denoting zero probability. The tree is updated as new observation is received.

## 5.2 Dynamic Programming for POMDP

The first step in solving a POMDP by dynamic programming is to convert it into a completely observable Markov decision process (MDP) with the set of states  $\mathbb{S} = \Delta(S)$ , i.e. the set of all possible beliefs about the state. In this paper, we focus on computing the optimal policy for  $\mathcal{T}$ -horizon planning given an initial belief state of the system  $b^0$ . When  $\mathcal{T} = 1$ , the agent can take only a single action. When  $\mathcal{T} = 2$ , it can take an action, make an observation, then take another action based on the observation made. In general, the agent's  $t$ -step policy can be represented by a tree. The root of the tree specifies the first action. The resulting observation determines the edge to be followed from the root, which determines the next action to be taken. A  $t$ -step policy tree, therefore, is a perfect  $|\Omega|$ -ary tree of depth  $t$ . The next step, is to find the value function, that gives the value of a policy, given an initial belief state. We can then use the value function to directly determine the optimal policy. Let  $\mathcal{V}_{p^t}(b)$  be the value function denoting the value of executing the  $t$ -step policy  $p^t$ , given the belief state  $b$ . Since:

$$\mathcal{V}_{p^t}(b) = \sum_{s \in S} b(s) V_{p^t}(s), \quad (3)$$

where  $V_{p^t}(s)$  is the value of executing the policy  $p^t$  in the state  $s$ , the next step is to find the value of a policy given a state of the system. This is given by:

$$V_{p^t}(s) = R(s, a(p^t)) + \sum_o O(s, a(p^t), o) \sum_{s'} T(s, a(p^t), s') V_{o(p^t)}(s'), \quad (4)$$

where  $a(p^t)$  denotes the action associated with the root of the policy tree  $p^t$ , and  $o(p^t)$  denotes the subtree (a  $(t - 1)$ -step policy tree) of  $p^t$ , whose root can be obtained by following the edge associated with observation  $o$ , from the root of  $p^t$ . Given an initial belief state  $b^0$ , the optimal  $\mathcal{T}$ -step policy is then given by:

$$\Pi^*(b^0) = \arg \max_{p^\mathcal{T}} \mathcal{V}_{p^\mathcal{T}}(b^0).$$

Notice from equation 3 that, associated with each  $t$ -step policy  $p^t$  is a vector of  $|S|$ -dimension. We call this the *value vector* of the policy and denote it by  $V_{p^t}$ . A single value vector  $V_{p^t}$  is enough to compute  $\mathcal{V}_{p^t}(b)$  for different values of  $b$ . The general dynamic programming algorithm for POMDP proceeds in  $\mathcal{T}$  iterations, where each iteration consists of two steps.

In the first step of iteration  $t$ , the algorithm is given a set  $P^{t-1}$  of  $(t - 1)$ -step policy trees computed from the previous iteration, and the corresponding set of value vectors  $V^{t-1}$ . It then computes the new sets  $P^t$  and the corresponding  $V^t$  as follows. First, a set of  $t$ -step policy trees,  $P_+^t$  is created by generating every possible  $t$ -step policy tree that makes a transition, after the initial action and observation, to the root of some  $t$ -step policy tree in  $P^t$ . Note that

$|P^t| = |A||P^{t-1}|^{|\Omega|}$ . The value vector for each of the new  $t$ -step policy tree can then be computed from equation 4, and stored in the set  $V_+^t$ .

In the second step, the set  $P_+^t$  is pruned by removing policy trees that need not be considered because they are dominated by some other policy trees in the set. Specifically, a policy tree  $p^t$  with the corresponding value vector  $V_{p^t}$  is dominated if for all  $b \in \mathbb{S}$  there exists another policy  $q^t \in P^t \setminus p^t$  such that  $\mathcal{V}_{q^t}(b) \geq \mathcal{V}_{p^t}(b)$ . This test of dominance is implemented using linear programming. If  $p^t$  is removed from  $P_+^t$ , then the corresponding value vector  $V_{p^t}$  is removed from  $V_+^t$  as well. The resulting set,  $P^t$  and  $V^t$ , are then sent to the next iteration of the algorithm.

## 6 Experimental Results

### 6.1 Extracting Driver Cruising Behavior from Historical Data

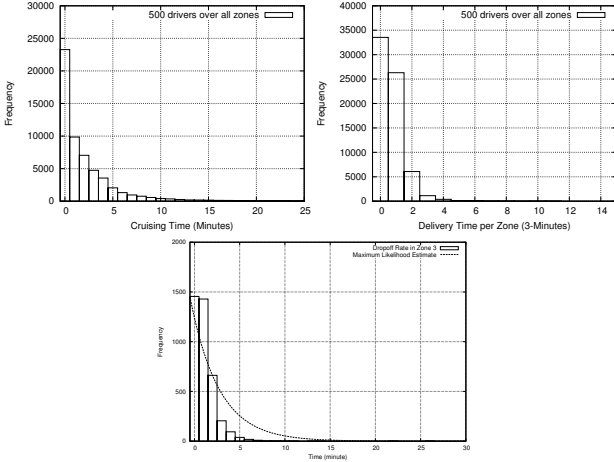
In order to derive the state transition function, we first analyze a real trace of historical data of a large Singapore taxi operator for one week on the Singapore road network. For this purpose, we obtained a dataset from a local taxi operator, which consists of a detailed information of each taxi in each second interval - its speed, its status (passenger on board, free, offline, etc), current position (given in latitude-longitude coordinate).

We first divide the Singapore’s road network into 88 logical zones (see for instance <http://www.onemap.sg/index.html>) and derive a mapping function from lat-long coordinates to the respective zone number. The information we need to extract from the dataset is the distribution patterns of the cruising. From the information extracted, we observe that for a randomly chosen 500 drivers and over all the zones, the following random variables are close to being exponentially distributed: (1) cruising time of an unoccupied taxi in a zone before moving to adjacent zone without finding any passenger, (2) time taken by the taxi moving from zone to zone while occupied, and (3) time taken to find the exact place in the destination zone to drop the passenger off. The frequency histograms for these random variables can be seen in Figure 3. This provides the motivation for us to model the driver cruising behavior as a continuous time Markov chain.

### 6.2 Simulation

To evaluate the quality of the POMDP policy, we compared it with two other policies, namely random and greedy. The random policy is to choose a random zone within its neighborhood to move to. The greedy policy, when unoccupied, is to move to a zone (among its current and adjacent zones) that has the least number of cruising taxis. Note that the random policy provides the lower bound benchmark. Any method that seeks to improve the agent cruising policy should at least perform better than a random policy. The question is whether the POMDP is more intelligent compared to the myopic greedy policy.

We simulated both passengers "arrivals" and drivers movement on the network. The passenger arrival in each zone is implemented as a Poisson process. As



**Fig. 3.** Observations from Real Dataset

a passenger appears in a zone, it joins a FIFO queue associated with the zone. If there are some taxis cruising in the zone, the passenger at the head of the queue is removed and assigned to a randomly chosen taxi. The average time spent in the queue models the average time spent by passengers waiting for a cruising taxi. We are interested in the utilization rate of the single agent driver in the network. The movement of all other drivers follows the continuous-time Markov chain model. When entering a zone in the cruising state, a passenger might be assigned to the driver, and it enters the occupied state. Otherwise, a set of exponentially distributed random variables is instantiated, one for each adjacent zone according to the corresponding rate in the Markov chain. The driver will wait for the smallest value of the instantiated random variables before moving to a zone following the edge associated with the random variable. While waiting, a passenger may still be assigned to the driver, in which case, it enters the occupied state instead. For initialization, we allow the system to run without the agent for a small time interval to reach a stable state. After a stable state has been reached, we freeze the system and save its configuration. Then, the agent driver to be tested is inserted to the system in a randomly chosen zone before resuming the simulation. The same agent is inserted to the same configuration several times before the average result is collected.

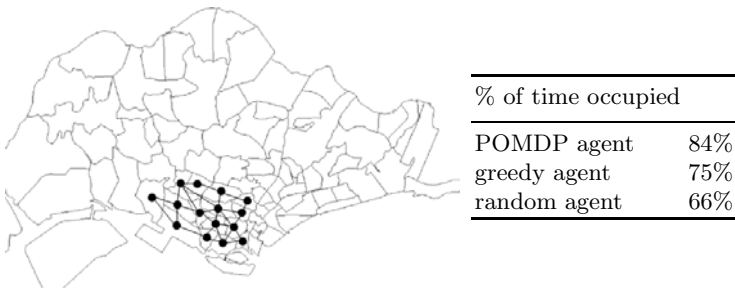
We performed our experiments on two sets of different network topologies and settings. First, on small grid networks. The parameters of the simulation are as follows. The average time between two passenger appearances in each zone is set at 0.5 minutes. The destination zone of a passenger are uniformly distributed over all the zones. During simulation, the agent accesses the state of the system every minute and decides on an action to take. The metric for comparison between the random, greedy and POMDP agents is the percentage of the 20-minute period when it was in the occupied state. Table 1 shows the results on different sizes of network, while the number of drivers was kept constant at 20. As shown in

the table, the improvements are more significant on a congested network (larger taxi-to-zone ratio) than on a sparse network. This is intuitive, as competition among the taxi drivers is fiercer on a congested network, and the agent with better cruising policy will generally perform better.

**Table 1.** Results of small grid networks

	$2 \times 2$	$3 \times 3$	$4 \times 4$	$5 \times 5$
POMDP agent	41%	72%	81%	85%
greedy agent	22%	61%	75%	80%
random agent	17%	51%	66%	71%

The second set of experiments is performed on a real 15-node network that models the congested central business district of Singapore and its surrounding zones (Figure 4). The arrival rate of passengers in each zone is derived from the real dataset discussed above. We consider only passengers with origins and destinations within these 15 zones, and restrict the cruising area of the drivers to be within these zones as well. Similar to the first set, the agent, in the unoccupied state, accesses the state of the system every minute and decides on the action to take. The number of drivers operating in the area is set to be 250. We simulate 2 hours of real time operations. As shown from the result, the POMDP agent performs significantly better than the greedy agent.



**Fig. 4.** Results of restricted area of Singapore

## 7 Future Works

We are aware of several limitations of the current model and its solution method. First is the issue of scalability. This is an inherent problem in practically all POMDP models for planning, and ours is no exception. It is thus interesting to see if the problem structure can be exploited for a more aggressive pruning strategy. It is also worthwhile to explore approximate solutions (for example by eliminating policy trees that are approximately dominated in the pruning step), to understand the trade-off between the quality of sub-optimal policies and the

gain in computation time. The second is the extension to multi-agent system. A possible extension is to consider a scenario where every taxi driver is equipped by a tool, telling them the optimal cruising policy, given the current state of the system and the policy of the other drivers. Information might be shared publicly, and each driver observe the state of the system completely, including the policy of the other drivers, or it might be kept private, and each agent has to guess the policy of the other drivers. The interesting questions are: What is the optimal policy of a driver in this case? What is the resulting state of the system if every agent execute its optimal policy, and how do we characterize equilibrium?

## References

1. Becker, R., Zilberstein, S., Lesser, V., Goldman, C.V.: Transition-independent decentralized markov decision processes. In: Proc. 2nd Conf. on Autonomous Agents and Multi-agent Systems, pp. 41–48 (2003)
2. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of markov decision processes. *Mathematics of Operation Research* 27, 819–840 (2002)
3. Cassandra, A., Littman, M.L., Zhang, N.L.: Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In: Proc. 13th Conf. on Uncertainty in Art. Intell., pp. 54–61 (1997)
4. Hansen, E.A.: Solving pomdps by searching in policy space. In: Proc. 14th Conf. on Uncertainty in Art. Intell., pp. 211–219 (1998)
5. Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: Proc. 19th AAAI Conf. on Art. Intell., pp. 709–715 (2004)
6. Hansen, E.A., Feng, Z.: Dynamic programming for pomdps using a factored state representation. In: Proc. 5th Conf. on AI Planning Systems, pp. 130–139 (2000)
7. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. *Art. Intell.* 101, 99–134 (1998)
8. Nair, R., Tambe, M., Yokoo, M., Pynadath, D., Marsella, S.: Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In: Proc. 18th Int'l Joint Conf. on Art. Intell., pp. 705–711 (2003)
9. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable markov processes over a finite horizon. *Operations Research* 21, 1071–1088 (1973)
10. Wong, K.I., Wong, S.C., Bell, M.G.H., Yang, H.: Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing markov chain approach. *Advanced Transportation* 39, 81–104 (2005)
11. Wong, K.I., Wong, S.C., Yang, H.: Calibration and validation of network equilibrium taxi model for hong kong. In: Proc. 4th Conf. of the Hong Kong Society for Transportation Studies, pp. 249–258 (1999)
12. Wong, K.I., Wong, S.C., Yang, H., Wu, J.H.: Modeling urban taxi services with multiple user classes and vehicle modes. *Transportation Research Part B: Methodological* 42, 985–1007 (2008)
13. Yang, H., Wong, S.C.: A network model of urban taxi services. *Transportation Research Part B: Methodological* 32, 235–246 (1998)
14. Yang, H., Wong, S.C., Wong, K.I.: Demand-supply equilibrium of taxi services in a network under competition and regulation. *Transportation Research Part B: Methodological* 36, 799–819 (2002)