Wireless PCS Systems Modeling and Optimization Using DEDS Approaches

Zikuan Liu

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements for the Degree of

Master in Computer Science at

Concordia University

Montreal, Quebec, Canada

August 2002

0-612-77716-2

Abstract

Discrete event dynamical systems (DEDS) are a class of man-made systems, which are driven by a set of discrete events. Typical examples include queuing systems, manufacturing systems, telecommunication networks, and so on. In the past two decades, the modeling, simulation, and optimization of DEDS have received considerable attention. The objective of this thesis is to apply the DEDS approaches to study wireless personal communication systems. Two applications are studied: the call request buffering problem and the mobile terminal location tracking problem.

In a personal communication service network, a set of channels is assigned to every cell. When a phone call arrives and is assigned a channel, it consumes the channel until the end of the conversation. If no channel is available, the call is dropped/blocked. In many cases, some channels may become available shortly after a call is dropped. Thus, if some buffering mechanism is used, a cell may accommodate more phone calls and thus the channel utilization is increased. This is referred to as call request buffering. The first part of this thesis develops an algorithm to estimate the sensitivity of a Markov process using a single sample path of its uniformization Markov chain, and uses a gradient algorithm to find a locally optimal parameter. After that, a random call request buffering scheme is introduced, which is optimized using the proposed algorithm.

In a wireless personal communication service system, in order to deliver a phone call, the network has to track the mobile terminals' locations from time to time. The second part of the thesis proposes a mobile terminal location update model, whose states consist of two components—the time elapsed since last call arrived and the distance the mobile terminal has traveled since last registration. This model is characterized as a Markov decision process. Using a Markov dynamical programming algorithm, the optimal update strategy is computed. To implement the optimal strategy, a new scheme is introduced to calculate the distance a mobile terminal has traveled since its last registration. This part also develops a simulation- based optimization algorithm for mobile location update, which can be implemented on-line.

Key Words: Personal communication service system, call request buffering, location update/registration, Markov decision process, online optimization.

Acknowledgment

I would like to express my deepest gratitude to my supervisor, Dr. T.D. Bui, for his invaluable assistance, constant guidance, and great encouragement. Especially, he initiated my research of applying discrete event dynamical systems theory and methods to telecommunication systems. I am very grateful to Dr. L. Narayanan for her constructive suggestions and insightful comments. I thank the graduate program secretary, Ms H. Monkiewicz, for her kindly advising and help. I also want to express my thanks to Ms. Y. Luo and Mr. X. Jiang, with whom I have discussed many problems in detail.

Finally, but not least, I want to thank my family for their great patience and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Discrete Event Dynamical Systems

Discrete Event Dynamical Systems (DEDS) are a class of man-made systems, which are driven by a set of discrete events that occur at discrete random instances. Typical examples of DEDSs are queuing systems. manufacturing systems. computer networks, personal communication service (PCS) networks, and so on. A DEDS is governed by a set of artificial rules such as CPU scheduling rules, rather than natural laws. Generally speaking, the state transitions of a DEDS are triggered by a set of discrete event processes. such as interrupts in computer operating systems, customer arrival and departure in queuing systems etc. These event processes may be asynchronous and concurrent, and moreover, they may not be independent of each other. Consequently. DEDSs can not be characterized by conventional differential equations or difference equations; and thus new models have to be developed. In the past two decades, a lot of DEDS models have been reported in the literature. Roughly speaking, these models can be classified into three categories according to their abstraction levels: untimed(logical). timed, and stochastic. The choice of the appropriate level of abstraction depends on the objectives of the analysis.

- Untimed models: Petri nets and automata. In some contexts, we are interested in the "logical behavior" of the system, that is, in ensuring that a precise ordering of events takes place which satisfies a given set of specifications; or we are interested in checking whether a particular state of the system can be reached or not, for example, "at a crossover, two sides have green traffic lights at the same time". In these cases, it suffices to consider only the logical behaviors of the systems.

- Timed Models: Timed automata, timed petri nets, and Dioid algebras. Use these models to deal with event timing. Typical questions to be answered include: "how much time does the system spend at a particular state?" or "how soon can a particular state be reached?" and so on. The most popular performance indices for timed models are throughput and response time.

- Stochastic timed model—Generalized semi-Markov processes (GSMP). GSMP is the most general probabilistic model. When the life times of all the events are exponentially distributed or geometrically distributed, a GSMP becomes a Markov process. Here we provide an informal definition of GSMP given by [11]. A more formal definition of GSMP can be found in [15], [6] and references therein.

Let S be the state space which is a subset of nonnegative integers. Let E be the (finite) set of all possible events in the GSMP. For each state $x \in \mathrm{S}, \Gamma(x)$ will be used to denote the feasible event subset associated with a particular state x. Also define

- $t_n \in R^+$— the epoch of the nth state transition (or the nth event time).

- $\alpha_n \in E$— the nth event.

- $x_n \in \mathrm{S}$—the nth state visited by the process $x_n = x(t_n+)$.

- $c_n(\alpha)$—at t_n, the time remaining until event α occurs, provided $\alpha \in \Gamma(x_n)$. $c_n(\alpha)$ is called the clock reading or lifetime of event α.

With the above notations, $\{x_n\}$ can be recursively defined by the following equations:

$$t_{n+1} = t_n + \min\{c_n(\alpha) : \alpha \in \Gamma(x_n)\} \tag{1.1}$$

$$\alpha_{n+1} = \{\alpha \in \Gamma(x_n) : c_n(\alpha) = \min\{c_n(\alpha') : \alpha' \in \Gamma(x_n)\}\} \tag{1.2}$$

$$x_{n+1} = \psi_n(x_n, \alpha_{n+1}, \eta_{n+1}) \tag{1.3}$$

where $\psi_n : \mathbf{S} \times E \times R \rightarrow \mathbf{S}$ specifies the state transition mechanism and η_{n+1} is used to denote possible stochastic state transition from x_n to x_{n+1} according to the following probability

$$P(x_{n+1} = j | x_n = i, \alpha_{n+1} = \alpha) \triangleq p_{ij,n+1}(\alpha)$$

The event clock reading is updated according to

$$c_{n+1}(\alpha) = \begin{cases} c(\alpha) - (t_{n+1} - t_n), & \text{if } \alpha \in \Gamma(x_{n+1}) \cap (\Gamma(x_n) \backslash \{\alpha_{n+1}\}) \\ Y_{n+1}(\alpha), & \text{if } \alpha \in (\Gamma(x_{n+1}) \backslash \Gamma(x_n)) \cup (\Gamma(x_{n+1}) \cap \{\alpha_{n+1}\}) \\ 0, & \text{if } \alpha \notin \Gamma(x_{n+1}). \end{cases} \tag{1.4}$$

where $Y_{n+1}(\alpha)$ denotes the life span of a newly generated event α. The above equations can be interpreted as follows: (1.1) and (1.2) state that the event with the shortest lifetime happens first and thus is the next event. Equation (1.3) represents the state transition caused by this triggering event. Once an event α_n occurs and the current state is x_n, then the state switching is governed by ψ_n given by (1.3). Equation (1.4) is used to update the event lifetime. A GSMP evolves in the following way: Given initial state x_0 (which sets initial clock Y_0), (1.1) and (1.2) determine the next event time and the next event. After that, (1.3) gives the next state. This is followed by clock reading updating by (1.4). Then go back to (1.1), and so on.

To illustrate the concepts of GSMP and how it can be used to characterize a queuing system, let us consider a $G/G/1$ system, in which customers arrive at a queue, wait for service, and leave upon finishing service. In this system, there are two events, customer arrival denoted by α, and customer departure denoted by β. Let x be the system state designating the number of customers in the system. Then, we identify this queuing system as a GSMP, with

$$E = \{\alpha, \beta\}$$

$$\mathbf{S} = \{0, 1, 2, \cdots\}$$

$$\Gamma(x) = \begin{cases} \{\alpha, \beta\}, & \text{if } x \geq 1, \\ \{\alpha\}, & \text{if } x = 0 \end{cases}$$

$$p_{ij,n}(\alpha) = \begin{cases} 1, & \text{if } j = i + 1 \\ 0, & \text{otherwise} \end{cases}$$

$$p_{ij,n}(\beta) = \begin{cases} 1, & \text{if } i \geq 1 \text{ and } j = i - 1 \\ 0, & \text{otherwise} \end{cases}$$

When all the life spans $Y_n(\alpha), n \geq 0, \alpha \in E$ are exponentially distributed or geometrically distributed, the above defined GSMP becomes a Markov process.

1.2 Thesis Goals

This thesis studies the problems of modeling and optimization of wireless networks using DEDS approaches. Two applications are studied: call request buffering and mobile terminal location tracking.

In a personal communication service network, channels are the most precious resources, which are assigned among cells. When a phone call arrives and is admitted, it occupies a channel until the termination of the conversation. If no channel is available, the call is dropped. In many cases, channels may return shortly after a call has been dropped. Thus, if some buffering mechanism is applied, a cell may accommodate more phone calls and thus the channel utilization can be improved. This thesis models the call request using a queuing system, and develops a random call buffering mechanism. An online optimization algorithm is introduced to maximize the channel utilization and guarantee the quality of service (QoS), which is represented by restricting the call admission delay under a given constant with a large enough probability.

In a wireless personal communication service system, in order to route a phone to its destination, the network has to trace the mobile terminal's location from time to time, which is a very expensive task. The second part of the thesis addresses the mobile terminal location update modeling and optimization problem.

The rest of the thesis is organized as follows: Chapter 2 develops a simulation-based optimization algorithm for continuous-time Markov processes using uniformization. As an application of the algorithm developed in Chapter 2, Chapter 3 studies the call request buffering problem. Chapter 4 proposes a mobile terminal location update model. A Markov dynamical programming algorithm is used to find the optimal strategy. Chapter 5 proposes an online location update algorithm. Chapter 6 is the epilogue of this thesis, which summarizes the thesis, puts forward a new mobility model that is currently under investigation, and gives some further research directions.

Chapter 2

Simulation-Based Optimization of Discrete Event Dynamical System

DEDSs are event driven dynamical systems, whose state transitions are driven by discrete events, e.g., packet arrival, phone call arrival in wireless communication system, etc. Generally, DEDSs are large scale, complex and stochastic systems, which make their design and performance evaluation full of challenges. Traditionally, there are two kinds of powerful tools to handle complex systems—mathematical analysis and computer simulation. Due to the complexity of DEDSs, neither traditional mathematical tools nor computer simulation methods can work very efficiently to deal with their performance evaluation and design optimization problems. The idea of simulation-based optimization is to combine both mathematical tools and simulation techniques, and to implement the optimization by two steps: First, under reasonable assumptions, construct a mathematical model and perform a mathematical derivation as much as possible. When mathematical derivation cannot go further any more, do the remaining work using simulation. Studying results in this field show that a little mathematical derivation can improve the simulation efficiency greatly. In the past two decades, numerous achievements on this topic have been reported in the literature [6].

Perturbation Analysis (PA) is a powerful optimization tool for DEDSs with continuous parameters. The idea of PA is to use **a single sample path** observation to estimate the gradient of the system performance measure with respect to the design parameter and then use a *hill climbing algorithm* to find the optimal design. Compared to the traditional ones, PA has two main advantages: First, it saves a lot of computation time because it needs only to simulate a single sample path. Second and more importantly, PA can be implemented on-line. This optimization approach has been successfully applied in many fields (see [6]).

The Markov process is a very popular model to characterize DEDS. Let $A = (a_{ij})$ be the infinitesimal generator of a Markov process. Since its infinitesimal characteristics, a_{ij}, are not only the system time parameters, but also the system structural parameters, a very small perturbation to a_{ij} will lead to change of the sojourn time on state i and the jump rate from state i to state j. Therefore, a single sample path sensitivity analysis of Markov processes has received considerable attention in the past two decades. This chapter is to implement the single sample path-based sensitivity analysis estimation algorithm and develop a gradient-based algorithm. To this end, let us give the definitions of continuous time Markov process and discrete time Markov chain and their simulation methods.

2.1 Markov Processes and Their Simulation

2.1.1 Continuous-Time Markov Processes

Let $(\Omega, \mathbf{F}, P)$ be a probability space, and let $\{X_t, t \geq 0\}$ be a stochastic process defined on $(\Omega, \mathbf{F}, P)$ and taking values in state space $\mathbf{S} = \{1, 2, \cdots\}$. Then, $\{X_t, t \geq 0\}$ is said to be a Markov process with state space $\mathbf{S}$ if

$$P(X_t = i | X_w : w \leq s) = P(X_t = i | X_s)$$

holds for all $0 \leq s \leq t$ and $i \in \mathbf{S}$. For any $i, j \in \mathbf{S}$, let $P_{ij}(s,t)$ denote the state transition probability, i.e.,

$$P_{ij}(s,t) = P(X_t = j | X_s = i).$$

Matrix $P(s,t) = (P_{ij}(s,t)), i, j \in \mathbf{S}$, is said to be the transition matrix. Under the continuity condition $\lim_{t \to s+} P(s,t) = I$, it follows that, for $0 \leq s \leq u \leq t$, the following hold:

$$P_{ij}(s,t) \geq 0, \ i,j \in \mathbf{S}$$
$$\sum_{j \in \mathbf{S}} P_{ij}(s,t) = 1, i \in \mathbf{S}$$
$$P_{ij}(s,t) = \sum_{k \in \mathbf{S}} P_{ik}(s,u)P_{kj}(u,t), i,j \in \mathbf{S}.$$

The last identity is usually referred to as the Chapman Kolmogorov equation. If the transition probability $P_{ij}(s,t)$ depends only on $t-s$, $\{X_t, t \geq 0\}$ is said to be stationary; otherwise, the process is nonstationary. In the sequel, we refer Markov process as stationary Markov process for simplicity. For a stationary Markov process, it can be shown [13] that the following hold

$$\lim_{t \to 0+} \frac{1 - P_{ii}(t)}{t} = a_i < \infty$$
$$\lim_{t \to 0+} \frac{P_{ij}(t)}{t} = a_{ij} < \infty, \ j \neq i.$$

Obviously, $a_i = \sum_{j \neq i} a_{ij}$. The parameter matrix $A = (a_{ij}), i,j \in S$ with by convention, $a_{ii} = -a_i$, gives the infinitesimal generator of the Markov process $\{X_t, t \geq 0\}$ and $a_{ij}'s$ are called the infinitesimal characteristics. Let $q_0 = (q_0(1), \cdots, q_0(k), \cdots)$ be the initial distribution of $\{X_t, t \geq 0\}$, that is, $q_0(i) = P(X_0 = i), i \in \mathbf{S}$. Then evolution of $\{X_t, t \geq 0\}$ is completely determined by q_0 and its infinitesimal generator matrix A.

For any $i \in \mathbf{S}$, the time of $\{X_t, t \geq 0\}$ staying in state i is exponentially distributed with parameter $-a_{ii}$. Let $\{T_n\}$ denote the sequence of times at which $\{X_t, t \geq 0\}$ changes its states. If $\{X_t\}$ is currently in state i, then at next instance $\{X_t\}$ jumps to state j with probability $a_{ij}/a_i (j \neq i)$. A generic sample path of $\{X_t\}$ can be generated using the following scheme.

Algorithm 2.1.1 *(Markov Process sample path generation algorithm)*

Step 1) Generate random number $u \sim U[0,1]$. *If*

$$u \in \Big[\sum_{i=1}^{j-1} q_0(i), \sum_{i=1}^{j} q_0(i)\Big) (q_0(0) = 0), \ \text{then } X_0 = j \text{ and } T_0 = 0.$$

Step 2) If at clock T_n, $\{X_t\}$ reaches state i, generate random number $u \sim U[0,1]$. Denote

$$A_n = \frac{1}{a_{ii}} \ln(1-u),\ T_{n+1} = T_n + A_n,$$

then, A_n is a sample time of staying on state i, and T_{n+1} is the next jump epoch.

Step 3) Generate random number $u \sim U[0,1]$. If

$$u \in \left[\sum_{k=1}^{j-1} -\frac{a_{ij}}{a_{ii}}, \sum_{k=1}^{j} -\frac{a_{ij}}{a_{ii}}\right) \ (a_{i0} = 0),$$

$X_{T_{n+1}} = j$.

Step 4) Let $n = n+1$ and go to Step 2.

In the above algorithm, step 2 generates the sojourn time on a state, and step 3 generates a state transition; so a generic sample path of a Markov process can be represented as $\{(s_n, A_n), n \geq 0\}$, where $\{s_n, n \geq 0\}$ is the sequence of state observed, and A_n is the time staying in state s_n.

2.1.2 Markov Chain Definition and Simulation

Let $\{X_n, n = 0, 1, \cdots\}$ be a sequence of random variables defined on $(\Omega, \mathbf{F}, P)$ and taking values in $\mathbf{S}$. If for any $i_0, i_1, \cdots, i_{n-1}, i, j \in \mathbf{S}$, and for an arbitrary $n \geq 0$ the following holds:

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \cdots, X_1 = i_1, X_0 = i_0) \triangleq p_{ij}, \tag{2.1}$$

then $\{X_n\}$ is said to be a Markov chain. Matrix $P = (p_{ij}), i, j \in \mathbf{S}$, denotes its one step state transition probabilities, which, obviously, satisfies

$$p_{ij} \geq 0,\ i, j \in \mathbf{S}$$
$$\sum_{j \in \mathbf{S}} p_{ij} = 1, \forall i \in \mathbf{S}.$$

The evolution of $\{X_n, n \geq 1\}$ is completely governed by its initial distribution q_0 and transition probability matrix P.

The following algorithm generates a generic sample path of $\{X_n, n = 0, 1, \cdots\}$.

Algorithm 2.1.2 *(Markov Chain sample path generation algorithm)*

Step 1) Generate random number $u \sim U[0,1]$. *If*

$$u \in \left[\sum_{i=1}^{j-1} q_0(i), \sum_{i=1}^{j} q_0(i)\right) (q_0(0) = 0), X_0 = j.$$

Step 2) Generate random number $u \sim U[0,1]$. *If*

$$u \in \left[\sum_{k=0}^{j-1} p_{ij}, \sum_{k=0}^{j} p_{ij}\right)$$

$X_{n+1} = j$.

Step 3) Let $n = n + 1$ *and go to Step 2).*

The above algorithm shows that a generic sample path of Markov chain can be represented by only a sequence of states. So, generating a sample path of a Markov chain is simpler than generating a sample path of a continuous-time Markov process. The next subsection gives a relationship between continuous-time and discrete-time Markov processes.

2.1.3 Relationship Between Continuous and Discrete Time Markov Processes

Let $\{X_t, t \geq 0\}$ be a continuous time Markov process with infinitesimal generator matrix $A = (a_{ij})$, satisfying $sup\{-a_{ii}, i \in S\} \leq \lambda$, where λ is a positive constant. It is easy to check that $P = I + \frac{1}{\lambda}A$ is a transition probability matrix, where I is the identity matrix. Then, there exists a version of process $\{X_t, t \in [0,\infty)\}$ (denoted by $\{X_t, t \in [0,\infty)\}$ for notational simplicity) having the representation $X_t = Y_{N(t)}$(see [8] Th.(8.4.31)), where $\{N(t), t \in [0,\infty)\}$ is a Poisson process with intensity λ and $\{Y_n, n \geq 0\}$ is a Markov chain, which is governed by $P = I + \frac{1}{\lambda}A$ and independent of $\{N(t), t \in [0,\infty)\}$. Constant λ is called the *uniformization parameter* and $\{Y_n, n \in 0\}$

the *uniformization Markov chain* of $\{X_t, t \in [0,\infty)\}$. Let $\pi = (\pi_1, \pi_2, \cdots)$ be a vector satisfying

$$\sum_{i=1}^{\infty} \pi_i = 1, \text{ and } \pi_i \geq 0, i \in \mathbf{S}.$$

Then, $\pi P = \pi$, i.e., $\pi(I + \frac{1}{\lambda}A) = \pi$, which is equivalent to $\pi A = 0$; so, $\{X_t, t \geq 0\}$ and $\{Y_n, n = 0, 1, \cdots\}$ have the same stationary distribution. Therefore, if consider only stationary distribution involved performance, we can use $\{Y_n, n = 0, 1, \cdots\}$ to take place of $\{X_t, t \geq 0\}$. This is illustrated in the next section in case of computing the sensitivity of the stationary distribution.

2.2 Single Sample Path Based Sensitivity Analysis for Markov Processes

Let $\{X_t, 0 \leq t < \infty\}$ be an irreducible Markov process with state space $\mathbf{S} = \{1, 2, \cdots\}$ and infinitesimal generator $A = (a_{ij}), i, j \in S$, where $a_{ij} \geq 0, i \neq j, a_{ii} < 0$, satisfying $\sup\{-a_{ii}, i \in \mathbf{S}\} < \infty$, and $\sum_{j=1}^{\infty} a_{ij} = 0$, for all $i \in \mathbf{S}$. Let $f(x) : \mathbf{S} \to R$ be a real function and $\pi = (\pi_1, \pi_2, \cdots)$ be the stationary distribution of $\{X_t, t \geq 0\}$. Then, under the assumption $\mathbb{E}_\pi(|f|) = \sum_{i=1}^{\infty} \pi_i |f(i)| < \infty$, the steady state performance measure is defined by

$$\eta = \mathbb{E}_\pi(f) = \sum_{i=1}^{\infty} \pi_i f(i). \tag{2.2}$$

Let $\delta > 0$ be a small enough real number and $Q = (q_{ij}), i, j \in S$ be an infinite matrix with $Qe = 0$, where $e = (1, 1, \cdots)^T$ and ^{T} denotes the transpose operator of a matrix. Then, the sensitivity(derivative) of η, with respect to A, in the direction of Q is defined by

$$\frac{\partial \eta}{\partial Q_A} = \lim_{\delta \to 0} \frac{\eta' - \eta}{\delta}, \tag{2.3}$$

where η' is the steady state performance measure of a Markov process $\{X'_t, 0 \leq t < \infty\}$ with infinitesimal generator $A' = A + \delta Q$. The following lemma gives the expression for $\frac{\partial \eta}{\partial Q_A}$.

Lemma 2.2.1 *([7]) The directional derivative of η with respect to A is given by*

$$\frac{\partial \eta}{\partial Q_A} = \pi Q D^T \pi^T,$$

where $D = (d_{ij}), i, j \in S$, is the matrix of perturbation realization factors with

$$d_{ij} = \mathbb{E}\left(\int_0^{S^j_{(i)}} (f(X^j_t) - \eta)dt\right). \tag{2.4}$$

Here $\{X^j_t, t \in [0, \infty)\}$ is a Markov process with initial state j and infinitesimal generator matrix A, and $S^j_{(i)}$ is the time at which $\{x^j_t\}$ first reaches state i, that is, $S^j_{(i)} = \inf\{t : X^j_t = i\}$.

From the previous section, we can study continuous-time Markov processes using their uniformization Markov chains. The following lemma indicates how to use the uniformization Markov chain to estimate the derivative of a Markov process. In particular, compared with Markov chain, Markov process has one more randomness, i.e. the randomness of the sojourn times. By using the uniformization Markov chain, the inter-event time of the Markov process is replaced by its mean, so using uniformization can reduce the estimation variances greatly (see Liu and Tu [17]).

Lemma 2.2.2 *(Liu and Tu [17]) Let $X = \{X_t, t \in [0, \infty)\}$ be a Markov process with infinitesimal generator A and with stationary distribution π. Let $\lambda \geq \sup\{-a_{ii}, i \in S\}$ be a positive constant, and $\{Y_n, n \geq 0\}$ the uniformization Markov chain of X with uniformization parameter λ. Then, the sensitivity of the performance measure η of $\{X_t, t \in [0, \infty)\}$ in the direction of Q is*

$$\frac{\partial \eta}{\partial Q_A} = \frac{1}{\lambda} \pi Q D^T \pi^T,$$

where $D = (d_{ij}), i, j \in S$ is the matrix of realization factors of the uniformization Markov chain $\{Y_n, n \geq 0\}$, that is,

$$d_{ij} = \mathbb{E}\left(\sum_{n=0}^{L^j_{(i)}} (f(Y^j_n) - \eta)\right), \tag{2.5}$$

where $Y^j = \{Y^j_n, n \geq 0\}$ is a Markov chain with $Y^j_0 = j$ and with transition matrix $P = I + \frac{1}{\lambda}A$, and $L^j_{(i)} = \min\{n; Y^j_n = i, n \geq 1\}$.

From above lemma, we have the following proposition.

Theorem 2.2.1 *Let $\{X_t, 0 \le t < \infty\}$ be an irreducible Markov process with infinitesimal generator matrix $A(\theta)$, where θ is the system design parameter to be optimized. Assume that $A_{ij}(\theta), i, j \in \mathbf{S}$, are differential in interval (a, b) and bounded by positive constant λ, then $\pi(\theta)$ and $\eta(\theta) = \sum_{j\in} f(j)\pi_j(\theta)$ are differential with respect to θ in (a, b) and*

$$\frac{d\eta}{d\theta} = \frac{1}{\lambda}\pi(\theta)\frac{dA(\theta)}{d\theta}D^T(\theta)\pi^T(\theta), \tag{2.6}$$

where $D(\theta) = (r_{ij}(\theta)), i, j \in S$, is the matrix of perturbation realization factors of a Markov Chain with transition matrix $I + \frac{1}{\lambda}A(\theta)$ and $\frac{dA(\theta)}{d\theta} = (\frac{da_{ij}(\theta)}{d\theta})$.

Proof: Since $\{X_t, 0 \le t < \infty\}$ is an ergodic Markov chain with state space $S = \{1, 2, \cdots\}$ and its infinitesimal characteristics $a_{ij}(\theta), i, j \in S$, are differentiable with respect to the controlled parameter $\theta \in (a, b)$. Gene and Meyer [19] proved that $\pi_i(\theta), i \in S$ are differentiable in (a, b). For any $\theta \in (a, b)$, when $\Delta\theta > 0$ is small enough,

$$a(\theta + \Delta\theta) = a(\theta) + \Delta\theta\dot{A}(\theta) + O(\Delta\theta^2),$$

with $\dot{A}(\theta) = (\dot{a}_{ij}(\theta))$. Since $A(\theta)e = 0$, and $\dot{A}(\theta)e = 0$, $A(\theta+\Delta\theta)$ can be approximately obtained by perturbating $A(\theta)$ by $\Delta\theta$ in the direction of $\dot{A}(\theta)$. Thus the proof follows directly from Lemma 2.2.2. Q.E.D.

The above theorem gives an estimator for the directive of η, where $\pi, dA(\theta)/d\theta$ are available, and r_{ij} is an unbiased estimator of the perturbation realization factor. The next section addresses how to optimize η based on the estimation of $d\eta/d\theta$.

2.3 Simulation-Based Optimization Algorithm for Markov processes

The idea of simulation-based optimization algorithm is to use a single sample path of a Markov process to estimate $\frac{\partial\eta}{\partial\theta}$, and then use a gradient algorithm, that is,

$$\theta_{t+1} = \prod_{\Theta}(\theta_t - c_t\tfrac{\partial\eta}{\partial\theta}(\theta_t)) \tag{2.7}$$

to find a local optimal θ, where

- Θ denotes the parameter space, which consists of all the possible parameters.
- $\prod_{\Theta}(\cdot)$ is a projection to Θ, that is

  $$\prod_{\Theta}(x) = \begin{cases} x, & \text{if } x \in \Theta \\ x', & \text{if } x \notin \Theta \end{cases}$$

  where x' is a point on the boundary of Θ that has the minimal distance to x.
- $\{c_t, t \geq 1\}$ is a sequence of positive numbers, which denotes the step sizes of the algorithm. In order for the algorithm to converge well, $\{c_t, t \geq 0\}$ is required to satisfy

  $$\sum_{t=1}^{\infty} c_t = \infty, \quad \sum_{t=1}^{\infty} c_t^2 < \infty \tag{2.8}$$

In the above algorithm 2.7, performance index η has no closed-form expression, so we have to estimate it by simulation. From (2.7), we conclude that the key task of using the above algorithm (2.7) is to estimate $\partial\eta/\partial\theta$. For this purpose, according to (2.6), we have to obtain the stationary distribution π, $\frac{dA(\theta)}{d\theta}$, and $D(\theta)$. Since π can be estimated directly and $\frac{dA(\theta)}{d\theta}$ is available, the difficulty lies in estimating $D(\theta)$. There are two methods to do this. First, simulate and record the sample path for a long enough time, e.g. with 10^6 state transitions. Then, call a computing subroutine to calculate $d_{ij}, i, j \in S$ and π. This method is direct and easy to implement. However, it has two limitations: (i) it needs an array to record the sample path which consumes a lot of memory. (ii) The simulation procedure and the computing procedure work sequentially, which makes this algorithm not easy to be implemented due to the following consideration: when implemented in a real system, the simulation procedure is replaced by a procedure to observe the occurences of the discrete events, such as call arrival and departure. In this case, when observing the system behavior, the computing procedure has nothing to do, but when it starts computing, it has too much work to process so that it may affect the observation of the system behavior.

To overcome these drawbacks, in this thesis we develop an algorithm that distributes the computing time. The algorithm need not use a long array to keep the sample path; instead, it lets the computing procedure and simulation procedure work at the same time in a way that once an event is observed, process it immediately.

2.3.1 Single Sample Path Based Estimators

Now, we proceed to construct the estimators needed to estimate $\frac{\partial\eta}{\partial\theta}$. For this purpose, we need to estimate η, π, D, which are denoted by $\hat{\eta}, \hat{\pi}$ and $\hat{D}$, respectively.

1) $\hat{\eta}_n = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$. It is easy to prove that $\lim_{n\to\infty}\hat{\eta}_n = \eta$, *w.p.*1 (see [12]).

2) $\hat{\pi}_i = \frac{1}{n}S_n(i)$, where $S_n(i)$ denotes the times that state i is visited by $X_0, X_1, \cdots, X_n$. The asymptotical unbiasedness of $\hat{\pi}_i$ follows by letting $f(\cdot) = \delta_i(\cdot)$, where $\delta_i(\cdot)$ is the indicator function.

3) In order to construct an estimator of d_{ij}, let us define two sets of random times $L_{ij}^{(n)}, S_{ij}^{(n)}, n \geq 1$ as follows:

$$L_{ij}^{(1)} = \min\{t, X_t = i\}, \; S_{ij}^{(1)} = \min\{t : t > L_{ij}^{(1)}, \text{ and } X_t = j\}.$$

Inductively, for $n \geq 2$,

$$L_{ij}^{(n)} = \min\{t : t \geq L_{ij}^{(n-1)}, \text{ and } X_t = i\}, \; S_{ij}^{(n)} = \min\{t : t > L_{ij}^{(n)}, \text{ and } X_t = j\}.$$

With $\{L_{ij}^{(n)}, n \geq 1\}$ and $\{S_{ij}^{(n)}, n \geq 1\}$ obtained, we can define $\hat{d}_{ij}^{(n)}$ by

$$\hat{d}_{ij}(n) = \frac{1}{n}\sum_{t=1}^{n} \hat{e}_{ij}^{(t)} \tag{2.9}$$

where $\hat{e}_{ij}^{(t)} = \left[\sum_{L_{ij}^{(t)}}^{S_{ij}^{(t)}} f(X_s) - \hat{\eta}_{\hat{n}}(S_{ij}^{(t)} - L_{ij}^{(t)})\right]$, with $\hat{n} = \max\{S_{ij}^{(n)} : i, j \in \mathbf{S}\}$

4) $\hat{L}_{ij}(n) = \frac{1}{n}\sum_{k=1}^{n}[S_{ij}^{(k)} - L_{ij}^{(k)}]$. Obviously, $\lim_{n\to\infty}\hat{L}_{ij}(n) = \mathbb{E}[L_{ij}]$, *w.p.*1.

2.3.2 The Algorithm Design

A direct method is to simulate a sample of $\{X_t, t \geq 0\}$ long enough and then use the observed sample path to estimate η, π, and further estimate $\frac{d\eta}{d\theta}$. However, in Monte-Carlo simulation, to estimate a parameter accurately, we have to simulate very long time; therefore, we have to record the sample path of $\{X_t, t \geq 0\}$, which is memory consuming. To overcome this problem, let us consider the following fact: Suppose $\{\xi_n, n \geq 1\}$ is a sequence of i.i.d. r.v.s with finite mean. Let $S_n = \frac{1}{n}\sum_{i=1}^{n}\xi_i$. Then we have according to the strong law of large numbers

$$\lim_{n\to\infty} S_n = \mathbb{E}[\xi_1], \quad w.p.1$$

During the observation, instead of keeping a long track of ξ_i, we can use

$$S_n = \frac{1}{n}\xi_n + \frac{n-1}{n}S_{n-1}. \tag{2.10}$$

To design the algorithm, let us introduce several matrices: $\hat{D} = (\hat{d}_{ij}), \hat{E} = (\hat{e}_{ij}), \hat{N} = (\hat{n}_{ij}), \hat{I} = (\hat{i}_{ij})$, and $\hat{L} = (\hat{L}_{ij})$, where

- $\hat{d}_{ij}$ is the estimator of d_{ij}
- $\hat{e}_{ij}$ is the estimator of e_{ij}
- $\hat{n}_{ij}$ denotes the number of terms of $\hat{e}_{ij}$ recorded
- $\hat{i}_{ij}$ denotes the status of counting $\hat{e}_{ij}$; at time t if $\hat{i}_{ij} = 1$, then there exists an integer r satisfying $S_{ij}^{(r)} \leq t \leq L_{ij}^{(r)}$, that is, $f(X_t)$ is counted and is added to $\hat{e}_{ij}$; otherwise, clock t satisfies $S_{ij}^{(w)} \leq t \leq L_{ij}^{(w+1)}$ for some integer w, which means that term $f(X_t)$ is skipped when estimating $\hat{e}_{ij}$.
- $\hat{L}_{ij}$ denotes the difference of $S_{ij}^{(t)}$ and $L_{ij}^{(t)}$.

The main point of the single sample path-based sensitivity analysis is to estimate all the perturbation realization factors $d_{ij}, i, j \in S$, through one sample path. For example, Table 2.1 shows a sample path of length 10

Table 2.1: A Sample Path of A Markov Process

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
0	1	1	2	3	0	1	2	3	4

So, x_0, x_1 can be used to estimate d_{10}; x_0, x_1, x_2, x_3 can be used to estimate d_{20}; x_1, x_2, x_3 used to estimate d_{21}, and so on.

With the above notations given, we can now present the sensitivity estimation algorithm as follows:

Algorithm 2.3.1 *(Sensitivity Estimation Algorithm)*

Step 1) Initialization. For all $i, j \in \mathbf{S}$, *set* $\hat{d}_{ij} = 0, \hat{e}_{ij} = 0, \hat{n}_{ij} = 0, \hat{i}_{ij} = 0$, $\hat{L} = 0$, $\hat{\eta} = 0, \hat{\pi} = 0$, *and* $t = 0$. *Generate initial state* x_0 *with initial distribution* q_0.

Step 2) Denote $x_t = j$.

- $\hat{\eta} = f(j)/t + \frac{t-1}{t}\hat{\eta}$
- $\hat{\pi}_j = \frac{1}{t} + \frac{t-1}{t}\hat{\pi}_j$, *and* $\forall i \neq j$ $\hat{\pi}_i = \frac{t-1}{t}\hat{\pi}_i$;
- *For all* $i, k \in \mathbf{S}$, *if* $\hat{i}_{ik} = 1$, *set* $\hat{e}_{ik} = \hat{e}_{ik} + f(j)$, *and* $\hat{L}_{ij} = \hat{L}_{ij} + 1$;
- *Put* $\hat{i}_{jk} = 1, \forall k \in \mathbf{S}$. *(Change all the zeros of* j*th row to one);*

- *For all* $i \in \mathbf{S}$, *if* $(\hat{i}_{ij} \neq 0)$ *and* $(j \neq i)$, $\hat{n}_{ij} = \hat{n}_{ij} + 1$; $\hat{d}_{ij} = \frac{\hat{e}_{ij}}{\hat{n}_{ij}} + \frac{\hat{n}_{ij}-1}{\hat{n}_{ij}}\hat{d}_{ij}$; $\hat{L}_{ij} = \frac{\hat{L}_{ij}}{\hat{n}_{ij}} + \frac{\hat{n}_{ij}-1}{\hat{n}_{ij}}\hat{L}_{ij}$; $\hat{i}_{ij} = 0$; $\hat{e}_{ij} = 0$.

Step 3) If the stop rule is satisfied, stop; otherwise, $t = t + 1$, *generate a new state for* x_t, *and go to Step 2.*

Example 2.3.1 *To show how the above works, let us implement the above algorithm using Matlab and work out a simple example. In the example, the Markov chain has state space* $\mathbf{S} = \{0, 1, 2, 3\}$ *with initial distribution* $q_0 = \begin{pmatrix} 0.1 & 0.4 & 0.3 & 0.1 \end{pmatrix}$ *and* *state transition probability matrix*

$$P = \begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.3 & 0 & 0.3 & 0.4 \\ 0.2 & 0.4 & 0 & 0.4 \\ 0.4 & 0 & 0.2 & 0.4 \end{pmatrix}.$$

The cost function $f(i) = i. i \in \mathrm{S}$.

```
clear all;
c0=1;  %% functional y=c0*xt;
q0=[.1 .4 .3 .1]; %% initial distribution

%% transition matrix
PP = [0.1  0.2  0.3  0.4;
      0.3  0    0.3  0.4;
      0.2  0.4  0    0.4;
      0.4  0    0.2  0.4];

   len = length(PP);
   nn=size(PP);

   rR = zeros(nn);
   iI = zeros(nn);
   eE = zeros(nn);
   nN = zeros(nn);

   lL = zeros(nn); %% estimate L=(L_{ij})
   elL = zeros(nn); %% estimate E[L]

   %% generate initial state
   x0=getstate(q0);
   eta=0;

   for iter=1:10000
   eta=c0*x0/iter + eta*(iter-1)/iter;
   iI(x0+1,:) = 1; %% set x0-th row to 1

   %% update eE
   for i=1:nn
```

```
        for j=1:nn
           if (iI(i,j)==1 & j~=i & j ~= x0+1)
              eE(i,j) = eE(i,j)+ c0*x0;
              elL(i,j) = elL(i,j)+1;
           end
        end
    end

for i=1:len
if iI(i,x0+1) ~= 0
    iI(i,x0+1) = 0;
    nN(i,x0+1) = nN(i,x0+1) +1;
    rR(i,x0+1) = eE(i,x0+1)/nN(i,x0+1)...
                + (rR(i,x0+1))*(nN(i,x0+1)-1)/nN(i,x0+1);
    lL(i,x0+1) = elL(i,x0+1)/nN(i,x0+1)...
                +(lL(i,x0+1))*(nN(i,x0+1)-1)/nN(i,x0+1);
    eE(i,x0+1)=0;
    elL(i,x0+1)=0;
end
end

x0=getstate(PP(x0+1,:));
end %% end of iter

nN
rR-eta*lL
```

With 10000 *state transitions, the following matrix is the number of times that estimators of d_{ij} are observed*

```
        2645        1091        1370        1888
        1091        1366        1114        1039
        1369        1114        1986        1485
        1888        1039        1486        4003
```

The estimated values of the perturbation realization factors are

```
         0   -0.8056   -1.6796   -2.6573
    0.8035         0   -0.8522   -1.8572
    1.6797    0.8501         0   -1.0117
    2.6558    1.8532    1.0109         0
```

The above algorithm provides a procedure to estimate the derivative $\partial\eta/\partial\theta$. With the derivative estimated, we can optimize the system performance using the gradient algorithm.

Algorithm 2.3.2 *(Simulation-Based optimization algorithm for Markov processes)*

Step 0) Choose initial parameter θ_0 arbitrarily, and set $t = 0$.

Step 1) With parameter θ_t given, simulate $\{X_t, t \geq 0\}$ for a long enough period. Based on the observation, estimate $\frac{\partial\eta}{\partial\theta}(\theta_t)$.

Step 2) Update system parameter $\theta_t = \prod_\Theta \left(\theta_t - c_t\frac{\partial\eta}{\partial\theta}(\theta_t)\right)$.

Step 3) Let $t = t + 1$. If the stop rule is satisfied, stop; otherwise, go to Step 1.

To illustrate the usefulness of the above algorithm, let us work out an example.

Example 2.3.2 *Consider a Markov chain with state space $\mathbf{S} = \{0, 1, 2, \cdots, 7\}$ and state transition probability matrix*

$$P = \begin{pmatrix} 1-\lambda & \lambda/2 & 0 & \lambda/2 & 0 & 0 & 0 & 0 \\ \lambda/3 & 1-\lambda & \lambda/3 & \lambda/3 & 0 & 0 & 0 & 0 \\ \lambda/4 & \lambda/4 & \lambda/4 & \lambda/4 & 0 & 0 & 0 & 1-\lambda \\ 0.1099 & 0.1445 & 0.0789 & 0.0155 & 0.1696 & 0.1548 & 0.1222 & 0.2046 \\ 0.0757 & 0.0665 & 0.1594 & 0.1972 & 0.0871 & 0.1441 & 0.1253 & 0.1447 \\ 0.1373 & 0.0159 & 0.0342 & 0.2316 & 0.2317 & 0.1515 & 0.0039 & 0.1939 \\ 0.0648 & 0.2483 & 0.0029 & 0.0073 & 0.2137 & 0.1326 & 0.2309 & 0.0995 \\ 0.0240 & 0.1857 & 0.1223 & 0.0997 & 0.1298 & 0.0728 & 0.1475 & 0.2182 \end{pmatrix}$$

where $\lambda \in (0, 1)$ is a design parameter. The cost function $f(\cdot) : \mathbf{S} \rightarrow \mathbb{R}$ is defined by

x	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$f(\cdot)$	*4*	*3*	*2*	*5*	*2*	*3*	*1*	*0*

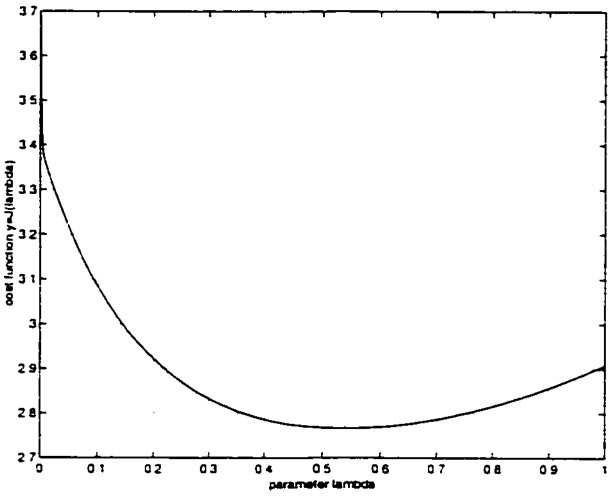

Figure 2.1: Performance function

The performance index considered is as defined by (2.2). The function η *versus design parameter* λ *is plotted in Figure 2.1*

To use the algorithm to find the optimal parameter λ, *we choose step size* $c_t = 1/t$. *Obviously, the direction of the derivative is given by*

$$Q = \begin{pmatrix} -1 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & -1 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For each step, to estimate $d_{ij}, i, j \in \mathrm{S}$, 20000 *state transitions are simulated. The difference between the estimated performance indices of two successive steps less than* 0.00051 *is used as stop rule. With* 0.25 *chosen as the initial value, the procedure stops after* 25 *iterations and generates the computing result:* $\lambda = 0.5400$, *and* $\eta = 2.7664$.

2.4 Conclusions

This chapter developed a simulation-based optimization algorithm for Markov processes. The proposed algorithm works as follows: First, use a single sample path to estimate the gradient of the performance with respect to the design parameters, and then with the gradient estimated, a *hill climbing algorithm* is used to optimize the system performance. A numerical example shows the proposed algorithm works very well.

Chapter 3

Call Request Buffering Strategies and Their On-Line Optimization

In a personal communication services network, a set of channels is dynamically or statistically assigned to every cell by some channel assignment algorithm. When a phone call arrives and there are some channels available, it uses a channel until the end of the conversation. If no channel is available, the call is dropped. In many cases, immediately following a new call drop, a channel becomes available, which is a waste of the network resources. Thus, if some buffering mechanism is introduced to the channel allocation algorithm, a cell may accommodate more phone calls. This is referred as call request buffering. In the literature there are two call request buffering schemes available: the system-control scheme and the user-control scheme [16].

The system-control scheme. As the hardware used for setting up calls becomes more and more powerful, the setup procedure can be made ever faster. However, as long as the calls are being set up for human users, shortening the call-setup delay below a few seconds becomes unobservable and irrelevant to the user. In the system-control scheme, the extra time available for setup is used to allocate channels more efficiently. If no channel is available when a call arrives, the call is buffered in a queue for a short timeout period τ (the value is usually much less than the mean call holding time). The timeout period for the system-control scheme is assumed to be either a constant or a random variable with an exponential distribution.

The user-control scheme. When someone (either a PCS subscriber or a person attempts to call a PCS subscriber) fails to initiate a phone call (assume that the only reason for call failure is that all channels are busy in a cell), the caller may redial a few minutes later. The user-control scheme allows the caller to set up a time out period τ. If a channel is available within τ, the system automatically connects the phone call. Otherwise, the call is dropped. The caller decides in advance whether to buffer the call request in a waiting queue, and specifies a timeout period τ.

In this chapter, a new system-control scheme is proposed: When a phone call arrives and no channel is available, with a probability p buffer a call and with probability $1-p$ drop it. The objective is to maximize the channel utilization and guarantee a certain quality of service that is represented by restricting the setup delay under a given constant with a large enough probability, that is,

$$\begin{aligned} \mathbf{P} : \max \quad & P(\text{no channel is idle}) \\ \text{s.t.} & P(\text{call setup delay} \geq D) \leq p_0 \end{aligned}$$

where D is a constant denoting quality of service, and p_0 is a very small probability, e.g., 10^{-9}.

3.1 Call Request Modeling

Assume that the phone call arrival is Poisson distributed with arrival rate λ, and the base station has m channels to be allocated. Each phone call duration is exponentially distributed with parameter μ. Let $x_t, t \geq 0$, denote the number of customers in the system. Then $\{x_t, t \geq 0\}$ is a birth-death process with birth rate λ_k and death rate μ_k given by

$$\lambda_k = \lambda \quad k = 0, 1, 2, \cdots, \tag{3.1}$$

$$\begin{aligned} \mu_k &= \min\{k\mu, m\mu\} \\ &= \begin{cases} k\mu & 0 \leq k \leq m \\ m\mu & m \leq k \end{cases} \end{aligned} \tag{3.2}$$

The infinitesimal generator of $\{x_t, t \geq 0\}$ is given by

$$A = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & \cdots \\ \mu_1 & -\mu_1 - \lambda & \lambda & 0 & 0 & \cdots \\ 0 & \mu_2 & -\lambda - \mu_2 & \lambda & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mu_k & -\mu_k - \lambda & \lambda \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{3.3}$$

Let $\pi_k, k = 0, 1, 2, \cdots$, be the stationary distribution of $\{x_t, t \geq 0\}$. Then, we have the following(see [21]): when $k \leq m$,

$$\pi_k = \pi_0 \prod_{i=0}^{k-1} \frac{\lambda}{(i+1)\mu} = \pi_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{k!}, \tag{3.4}$$

and, when $k \geq m$,

$$\pi_k = \pi_0 \prod_{i=0}^{m-1} \frac{\lambda}{(i+1)\mu} \prod_{j=m}^{k-1} \frac{\lambda}{m\mu} = \pi_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{m! m^{k-m}}. \tag{3.5}$$

Let $\rho = \frac{\lambda}{m\mu}$. Combining (3.4) and (3.5), we obtain

$$\pi_k = \begin{cases} \pi_0 \frac{(m\rho)^k}{k!} & k \leq m \\ \pi_0 \frac{\rho^k m^m}{m!} & k \geq m \end{cases} \tag{3.6}$$

Solving equation $\sum_{i=0}^{\infty} \pi_i = 1$ yields

$$\begin{aligned} \pi_0 &= \left\{ 1 + \sum_{k=1}^{m-1} \frac{(m\rho)^k}{k!} + \sum_{k=m}^{\infty} \frac{(m\rho)^k}{m!} \frac{1}{m^{k-m}} \right\}^{-1} \\ &= \left\{ \sum_{k=1}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^k}{m!} \left(\frac{1}{1-\rho} \right) \right\}^{-1} \end{aligned} \tag{3.7}$$

3.2 Call Request Buffering Optimization

This section considers the optimization problem of the call request buffering problem. For this purpose, we consider a performance index $J(\lambda)$ as follows:

$$J(\lambda) = \sum_{i=0}^{m} C_0 (m-i) \pi_i + C_1 P(W > D), \tag{3.8}$$

where C_0, C_1 are two positive constants denoting the cost of a channel being idle and the cost of a costumer waiting longer than a given D, which represents the quality of service of the network. W is a random variable denoting the waiting time in the system. Since $x_t < m$ means that some channels are idle, no waiting is involved, that is, $P(W > D|x_t < m) = 0$, we obtain

$$\begin{aligned} P(W > D) &= \sum_{i=m}^{\infty} P(W > D, x_t = i) \\ &= \sum_{i=m}^{\infty} P(W > D|x_t = i)P(x_t = i) \\ &= \sum_{i=m}^{\infty} \pi_i P(W > D|x_t = i) \end{aligned} \tag{3.9}$$

Direct computation gives: for $i \geq m$,

$$P(W > D|x_t = i) = \int_D^{\infty} \frac{m\mu e^{-m\mu x}(m\mu x)^{i-m}}{(i-m)!} dx \tag{3.10}$$

which combined with (3.5) yields

$$\begin{aligned} P(W > D) &= \sum_{i=m}^{\infty} \pi_i P(W > D|x_t = i) \\ &= \sum_{i=m}^{\infty} \pi_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{m!m^{k-m}} \int_D^{\infty} \frac{m\mu e^{-m\mu x}(m\mu x)^{i-m}}{(i-m)!} dx \\ &= \int_D^{\infty} \sum_{i=m}^{\infty} \pi_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{m!m^{k-m}} \frac{m\mu e^{-m\mu x}(m\mu x)^{i-m}}{(i-m)!} dx \\ &= \frac{\pi_0 \left(\frac{\lambda}{\mu}\right)^m}{m!} \int_D^{\infty} \sum_{i=m}^{\infty} \frac{m\mu e^{-m\mu x}(\lambda x)^{k-m}}{(k-m)!} dx \\ &= \frac{\pi_0 \left(\frac{\lambda}{\mu}\right)^m m\mu}{m!} \int_D^{\infty} e^{(\lambda - m\mu)x} dx \\ &= \frac{\pi_0 \left(\frac{\lambda}{\mu}\right)^m m\mu}{m!(\lambda - m\mu)} e^{(\lambda - m\mu)D} \end{aligned} \tag{3.11}$$

So, the performance function $J(\lambda)$ becomes

$$J(\lambda) = \sum_{i=0}^{m} C_0(m-i)\pi_i + C_1 P(W > D)$$

$$= \sum_{i=0}^{m-1} C_0(m-i)\frac{(m\rho)^i \pi_0}{i!} + C_1 \frac{\pi_0 \left(\frac{\lambda}{\mu}\right)^m m\mu}{m!(\lambda - m\mu)} e^{(\lambda - m\mu)D} \tag{3.12}$$

The optimization problem is to find an optimal arrival rate λ such that $J(\lambda)$ in (3.12) is minimized. The control on arrival rate can be achieved by deleting some arrivals from the original arrival process, which is called *thinning*. For this purpose, we need an important property of Poisson process as follows: Let $\{r_t, t \geq 0\}$ be a Poisson process with arrival rate λ, and let $\theta \in [0, 1]$ be a constant. If $\{r_t, t \geq 0\}$ is thinned with rate $1 - \theta$, that is, when a new event arrives, with probability θ admit it and with probability $1 - \theta$ drop it, then the thinned process is also a Poisson process with arrival rate $\lambda\theta$.

So, the optimization problem can be given by

$$\mathbf{P}_1 : \min_{\theta \in [0,1]} J(\theta\lambda) \tag{3.13}$$

In optimization problem $\mathbf{P}_1$, the thinning parameter θ is applied to all states. Obviously, if we apply a state feedback control, that is, let the thinning rate depend on the state of r_t, the system performance can improved a lot. Let $\theta_k, 0 \leq k < \infty$ denote the the thinning rate when $x_t = k$. Then the arrival rate of x_t when $x_t = k$ is $\lambda_k = \lambda\theta_k$. The infinitesimal generator of $\{x_t, t \geq 0\}$ is given by

$$A = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & 0 & \cdots \\ \mu_1 & -\mu_1 - \lambda_1 & \lambda_1 & 0 & 0 & \cdots \\ 0 & \mu_2 & -\lambda_2 - \mu_2 & \lambda_2 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mu_k & -\mu_k - \lambda_k & \lambda_k \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{3.14}$$

For this system, the stationary distribution π of $\{x_t, t \geq 0\}$ can be given directly (see [21]). When $k \leq m$,

$$\pi_k = \pi_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{(i+1)\mu}; \tag{3.15}$$

when $k \geq m$,

$$\begin{aligned}\pi_k &= \pi_0 \prod_{i=0}^{m-1} \frac{\lambda_i}{(i+1)\mu} \prod_{j=m}^{k-1} \frac{\lambda_j}{m\mu} \\ &= \pi_0 \frac{\prod_{i=0}^{k-1} \lambda_i}{m! m^{k-m} \mu^k} \end{aligned} \tag{3.16}$$

Combining (3.15), (3.16) and the fact that $\sum_{i=0}^{\infty} \pi_i = 1$ yields

$$\pi_0 = \frac{1}{1 + \sum_{k=1}^{m-1} \frac{\prod_{i=0}^{k-1} \lambda_i}{k! \mu^k} + \sum_{k=m}^{\infty} \frac{\prod_{i=0}^{k-1} \lambda_i}{m! m^{k-m} \mu^k}}$$

Let $\theta = (\theta_0, \theta_1, \cdots, \theta_k, \cdots) \in [0,1]^{\infty}$ and denote the performance index by $J(\theta)$. Then, the optimization problem is

$$\mathbf{P}_2 : \quad \min_{\theta_k \in [0,1], 0 \leq k < \infty} J(\theta) \tag{3.17}$$

Let us define $f(\cdot)$ as follows:

$$f(i) = C_0 \sum_{x=0}^{m} (m - x + 1)\delta_x(i) + C_1 \sum_{x=m+1}^{\infty} \rho_x \delta_x(i) \tag{3.18}$$

where $\rho_x = P(W > D | x_t = x)$. Then the cost function becomes

$$J(\theta) = \lim_{T \to \infty} \frac{1}{T} \mathrm{E} \left[\int_0^T f(x_t) dt \right] = \sum_{i=0}^{\infty} \pi_i f(i). \tag{3.19}$$

The dynamical programming equation for optimization problem (3.17) can be given by

$$v + h(i) = \min_{\theta_i \in [0,1]} \left[f(i) + \sum_{j \in \mathbf{S}} a_{ij} h(j) \right] \tag{3.20}$$

Therefore, if there exists a constant v and a bounded function $h(i), i \in \mathbf{S}$, satisfying (3.20), then v is the optimal cost $J(\theta^*)$ with θ^* denoting the optimal parameter. Equation (3.20) is the optimality equation for the optimization problem $\mathbf{P}_2$. Noting the special structure of (3.18), we find (3.20) is

$$\begin{aligned} v + h(i) &= \min_{\theta_i \in [0,1]} [f(i) - \mu_i(h(i) - h(i-1)) + \lambda\theta_i(h(i+1) - h(i))] \\ &= [f(i) - \mu_i(h(i) - h(i-1))] + \min_{\theta_i \in [0,1]} [\lambda\theta_i(h(i+1) - h(i))] \end{aligned} \tag{3.21}$$

from which it follows that the optimal θ_i is equal to either 0 or 1, that is,

$$\theta_i = \begin{cases} 1, & \text{if } h(i+1) \le h(i) \\ 0, & \text{otherwise} \end{cases} \tag{3.22}$$

Therefore, optimization problem $\mathbf{P}_2$ is equivalent to

$$\mathbf{P}_2' : \min_{\theta_k \in \{0,1\}, 0 \le k < \infty} J(\theta)$$

3.3 Simulation-Based Optimization

This section focuses on the simulation-based optimization of the call request buffering. To this end, we need to derive a single sample path-based estimator of $\partial\eta/\partial\theta$.

The following theorem provides the single sample path-based estimator of $\partial J(\theta)/\partial\theta$.

Theorem 3.3.1 *Let $J(\theta)$ be the performance index defined by (3.19). Then its gradient with respect to θ is given by*

$$\frac{\partial J(\theta)}{\partial\theta} = \frac{\lambda}{\lambda + m\mu}\mathrm{diag}(\pi)\hat{Q}D^{\top}\pi^{\top}$$

where D is the perturbation realization factor matrix of a Markov chain with transition matrix $P = I + \frac{1}{\lambda+m\mu}A$,

$$\hat{Q} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & \cdots \\ 0 & -1 & 1 & 0 & 0 & \cdots \\ 0 & 0 & -1 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \vdots \tag{3.23}$$

$$\mathrm{diag}(\pi) = \begin{pmatrix} \pi_0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \pi_1 & 0 & 0 & 0 & \cdots \\ \cdots & \cdots & \ddots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \pi_k & \cdots & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix} \tag{3.24}$$

Proof: Note that $0 < -a_{ii} \le m\mu + \lambda$, so we can choose the uniformization parameter as $m\mu + \lambda$. According Theorem 2.2.1, we have

$$\frac{dJ(\theta)}{d\theta_i} = \frac{1}{m\mu+\lambda}\frac{dA(\theta)}{d\theta_i}D^{\mathsf{T}}\pi^{\mathsf{T}}. \tag{3.25}$$

Direct computation gives

$$\frac{dA(\theta)}{d\theta_i} = \hat{Q}_i = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & -\lambda & \lambda & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}$$

That is, all elements of $\hat{Q}_i$ are zeros except its i-th row, which is denoted by α_i. So, we have

$$\begin{aligned} \frac{dJ(\theta)}{d\theta_0} &= \frac{1}{m\mu+\lambda}\pi_0\alpha_0 D^{\mathsf{T}}\pi^{\mathsf{T}} \\ \frac{dJ(\theta)}{d\theta_1} &= \frac{1}{m\mu+\lambda}\pi_1\alpha_1 D^{\mathsf{T}}\pi^{\mathsf{T}} \\ &\vdots \\ \frac{dJ(\theta)}{d\theta_k} &= \frac{1}{m\mu+\lambda}\pi_1\alpha_k D^{\mathsf{T}}\pi^{\mathsf{T}} \\ &\vdots \end{aligned}$$

from which follows the proof of Theorem 3.3.1. Q.E.D.

Example 3.3.1 *Consider a base station with* 5 *channels. The call arrival rate is* $\lambda = 1$. *Each call lasts a random time with exponential distribution with parameter* $\mu = 0.3$. *Set the upper bound of waiting time* $D = 4$ *seconds. Assume* $C_0 = 1, C_1 = 10$, *that is, a call request waiting time exceeding* D *costs* 10 *times as a channel being idle. With theses parameters, the probabilities of waiting time exceeding* D *are*

i	0	1	2	3	4	5
$P(W \ge D \mid x_t = i)$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001

6	*7*	*8*	*9*	*10*	*11*	*12*	*13*
0.0010	*0.0026*	*0.0045*	*0.0063*	*0.0081*	*0.0097*	*0.0113*	*0.0128*

14	*15*	*16*	*17*	*18*	*19*
0.0141	*0.0154*	*0.0167*	*0.0178*	*0.0189*	*0.0199*

Using the algorithm 2.3.2, the following results are obtained

$$\theta = \begin{cases} 1, & x_t \leq 9 \\ 0, & \textit{otherwise} \end{cases}$$

and the optimal cost $J = 0.2969$.

Remark 3.3.1 *In the above example, the optimal strategy can be implemented by simply setting a buffer with size* 4. *When a new phone call arrives, if there are some free space, buffer it; otherwise, drop it.*

Chapter 4

Mobile Terminal Location Tracking Management

4.1 Introduction

A personal communication network consists of a wired network and mobile terminals. Each mobile terminal (MT) communicates with the network through a nearby base station. In order to route incoming calls to a destination mobile terminal, the network must keep track of the location of each mobile terminal from time to time, which is called MT location tracking. Mobile terminal location tracking is based on two elementary events: paging and location update (or registration).

- When a new phone call arrives, the network has to search the network to find where the MT is so that it can deliver the call, which is called paging.
- By update, the MT sends a message to the network to report the network about its current location so that the network can update its locations in the Location Information Databases (LIDs).

There is a trade-off between the frequency of location update and the number of cells paged in order to track down a mobile terminal. If an MT updates its location immediately once it moves to a new cell, the network always keeps the precise location of the MT; however, the network has to waste a lot of time to process updates. On the other hand, if the MT doesn't update frequently, when a call arrives, the network has to page a large area, which wastes radio bandwidth. Therefore, the central problem of location management is to devise an algorithm that minimizes the overall cost of location update and paging.

4.1.1 Registration/Paging Schemes Used in Current Real Systems

At present, there are two commonly used standards: the EIA/TIA Interim Standard 41 (IS41) in North America and the Global System for Mobile Communications in Europe (GSM). In the two systems, the network coverage is partitioned into a number of location areas (LA), each consisting of a group of cells. When an MT enters an LA, it reports to the network the information about its current new location. When an incoming call arrives, the network simultaneously pages the mobile terminal in all cells within the LA where the mobile terminal is currently registered. In these standards, the LA coverage is fixed for all users.

4.2 Literature Survey

Due to its great importance, the MT location tracking problem has been extensively studied in the past ten years. I.F.Akyildiz, J. McNair, et. al. [31], and V. Wong and V. Leung [32] provided comprehensive surveys of this topic. Roughly speaking, location update algorithms can be divided into two groups: static and dynamic.

In a static algorithm, location update is triggered based on the topology of the network. Examples include the conventional location area(LA)-based scheme used in GSM and IS-41. The network coverage area is partitioned into a number of LAs, each having an LA ID. All base stations within the same LA broadcast the LA ID of their LA periodically. Each MT compares its registered LA ID with the current broadcasted LA ID. Location update is triggered if the two IDs are different. Upon a call arrival for a particular mobile terminal, all cells within its current LA are polled simultaneously, ensuring success within a single step.

In a dynamic algorithm, location update is based on the user's call and mobility patterns. In the literature, a lot of dynamic algorithms have been reported.

- *Selective LA update* [28]. Obviously, in the LA-based update scheme that is currently in use, the MTs that reside at the boundary of an LA and cross the boundary frequently have to update their locations frequently, which causes a lot of network resource waste. To overcome this drawback, Sen et. al [28] introduced a selective LA update scheme. They established an analytical model in which the interconnections of the LAs are characterized by a graph model and a Markov movement model is used. In this scheme, instead of performing location update whenever an MT crosses a new LA, the update process at certain LAs can be skipped.

- *Profile-Based scheme* [24]. In this scheme, the network maintains a profile for each MT, including a sequential list of the LAs the user is most likely to be located at in different time periods. This list is sorted from the most to least likely LA where an MT can be found. When a call arrives, the LAs on the list are paged sequentially. As long as the mobile terminal moves between LAs within the list, no location update is necessary. Location update is performed only when the mobile terminal moves to a new LA not on the list. The list may be derived from the user's movement history.

- *Movement-Based Update Scheme.* Amotz Bar-Noy et. al. [5] first introduced a movement-based update strategy, in which the MT counts the number of cell boundaries it has crossed and performs update once the counted number reaches a predefined threshold. The threshold is a system design parameter. Akyildiz et. al. [1] introduced an analytical model to determine the optimal movement threshold. The model is applicable for mesh and hexagonal cell configurations under the assumptions of a general cell residence time distribution and symmetric random walk movement pattern.

- *Time-based update scheme.* Amotz Bar-Noy et. al. [5] first studied a time-based update strategy, in which a timer is set immediately after an update. The MT is prompted to perform another update once its timer times out. Rose [26] proposed an analytical model which assumes Gaussian distribution of user location probability and Poisson call arrival. The optimal update period that minimizes the cost of location update and paging is derived.

Z. Naor and H. Levy [20] introduced a variation of the time-based scheme called *the adaptive threshold scheme*. The MT updates its location every T time slots, where the parameter T is not a constant, but varies with the current signaling load on the uplink control channel of the base station.

- *Distance-based scheme.* Amotz Bar-Noy et. al. [5] first introduced a distance-based update strategy, by which an MT calculates the distance (measured by the number of cells rather than Euclidian distance) it has traveled from its last registered cell, and performs update whenever the distance it traveled reaches a predefined threshold, which is a system parameter. Amotz Bar-Noy et. al. [5] first introduced time, movement, and distance-based strategies and declared that the distance-based strategy has better performance than the other two types of strategies; however, the distance-based scheme requires that the MT has some knowledge about the network topology so that it can calculate the distance traveled, which makes its implementation much harder than the other two kinds of strategies. Among these strategies, time-based strategy is the easiest one to implement, but it is the least accurate one. After the work of Amotz Bar-Noy et. al. [5], distance-based strategies have been extensively studied.

  - U. Madhow et. al.[18] considered a one-dimensional linear model and symmetric random walk movement patterns. In this model, they formulated the distance-based update scheme as an optimization problem that minimizes the expected total cost for update and paging.
  - Liang and Haas [22] introduced a variant of distance-based scheme called *predictive distance-based scheme*, in which the MT checks its position peri-

odically and performs update whenever its distance exceeds the threshold distance measured from the predicted location. It reports both its location and velocity during the update process. Based on the reported information, the network determines the probability density function of the mobile's location, which is used to predict the mobile terminal's location in the future time. Upon a call arrival, the network pages the mobile terminal starting from the predicted location and outwards, in a shortest-distance-first order, until the mobile terminal is found.

  - *Interactive Scheme.* Naor [30] proposed an interactive scheme which is based on the distance the mobile terminal has moved since its last update and the time elapsed since its last update.
  - *Adaptive Distance-Based Scheme* [23]. W.S. Wong and C.M. Leung [23] proposed a stochastic model to compute the optimal update boundary for the distance-based location update algorithm. The model allows that the cell residence time can follow general distributions which captures the fact that the mobile user may spend more time at certain locations than others, and the model also incorporates the concept of a trip in which the mobile user may follow a particular path to a destination. The objective is to minimize the expected paging and update cost, which is done by solving a Markov decision problem.

- *State-Based Update Scheme* [25]. Rose [25] introduced a state-based update scheme, where the system state includes the current location and the time elapsed since the last update. A time-varying Gaussian process is used to model the user's movement. A suboptimal solution for the average cost of location update and paging under no paging delay constraint is obtained by a greedy method.

At the same time, there are a lot of paging schemes available in practical networks and literature. *Blanket polling paging strategy* is currently used in real networks, and a lot of paging strategies are proposed in the literature: terminal paging, shortest-distance first, sequential paging based on a user's location probability, velocity paging, ensemble polling and so on.

This chapter proposes a Markov model that subsumes the time elapsed since last call arrival, the distance from the mobile's last registration, and call interarrival time estimation. Using Markov dynamical programming, the optimal state-feedback (state: time and distance) control is obtained. In order to implement a distance-involved mobile location update strategy, the MT has to learn the topology of the network so that it can calculate the distance it has traveled. For this purpose, we also introduce a scheme for an MT to compute the distance it has traveled.

The rest of this chapter is organized as follows: Section 3 presents the mobile movement models in one-dimensional and two dimensional systems. Section 4 solves the optimal control strategy using the algorithm of Markov decision processes. Section 5 addresses the implementation issues and proposes a traveled distance computation scheme. Section 6 studies the distance-based strategies and their time-based invariant.

4.3 System Models

In this chapter, one dimensional and two dimensional MT movement models are considered. With some mathematical processing, the two kinds of models have the same structure and thus can be handled by the same manner.

4.3.1 One-Dimensional Model Description with Interarrival Time Estimation

Assume that an MT moves on a straight line, e.g., on a highway, as shown in Figure 4.1.

Let τ_t be the time elapsed at time t since last call arrival and x_t be the distance traveled from the cell in which last update is performed (it is the MT's current location in the LIDs). In next time slot $t+1$ with p the mobile terminal will stay in the same

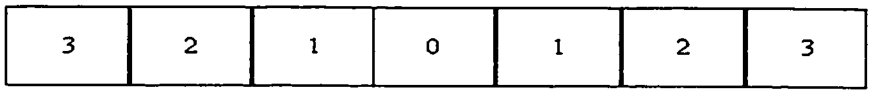

Figure 4.1: The One Dimensional Movement Model

cell and with probability $q/2(q = 1 - p)$ move to a cell with distance $x_t + 1$ or $x_t - 1$, that is,

$$x_{t+1} = \begin{cases} x_t, & \text{with probability } p \\ x_t + 1, & \text{with probability } q/2 \\ x_t - 1, & \text{with probability } q/2 \end{cases} \tag{4.1}$$

Remark 4.3.1 *Note that parameter p is determined by the cell size and mobile speed. Let L be the distance between the mobile's current location and the exit point. Then, $p = P(L < v\Delta) = \frac{v\Delta}{R}$, where v is the MT moving speed and Δ denotes the time slot length.*

Let $t_n, n \geq 1$ be the sequence of observed interarrival times. The next arrival time is estimated by

$$A_{n+1} = \alpha t_n + (1 - \alpha) A_n + W_{n+1},$$

where $\alpha \in (0, 1)$ is a given parameter, $\{W_n, n \geq 1\}$ is a sequence of random variables with zeros mean and variance σ^2. Let $e_{n+1} = \mathbb{E}[A_{n+1}]$ be the estimated mean of next call interval time, then,

$$e_{n+1} = \alpha t_n + (1 - \alpha) e_n. \tag{4.2}$$

If we further assume that $\{W_n\}$ has Gaussian distribution, the interarrival time distribution function $F(r, t)$ is given by

$$F(e_t, t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-e_t)^2}{2\sigma^2}} dx \tag{4.3}$$

Define stochastic process $y_t = (e_t, \tau_t, x_t), t \geq 0$, where e_t is the estimated mean of the next call arrival time which remains constant between two call arrivals, τ_t denotes the time elapsed since last call arrival, and the distance x_t is the distance from its last update location. Then, $\{y_t, t \geq 0\}$ is a Markov process. If the system current state is (e, t, k), a new call arrival switches the first component of the system state to

$$\alpha t + (1 - \alpha)e,$$

according to (4.2), and resets the elapsed time τ_t and x_t to zeros. Note that the arrival of a new phone call is independent of the mobile's position. The uncontrolled system state transition probabilities

$$P(y_{t+1} = (e', t', k') | y_t = (e, t, k)) \triangleq p_{(e,t,k)(e',t',k')}$$

are given by: if $k \neq 0$,

$$\begin{cases} p_{(e,t,k)(\alpha t+(1-\alpha)e,0,0)} = \frac{F(e,t+1)-F(e,t)}{1-F(e,t)}, & \text{(a new call comes)} \\ p_{(e,t,k)(e,t+1,k+1)} = \frac{q}{2}\frac{1-F(e,t+1)}{1-F(e,t)}, & \text{(no call, moves far)} \\ p_{(e,t,k)(e,t+1,k-1)} = \frac{q}{2}\frac{1-F(e,t+1)}{1-F(e,t)}, & \text{(no call, moves back)} \\ p_{(e,t,k)(e,t+1,k)} = p\frac{1-F(e,t+1)}{1-F(e,t)}, & \text{(no call, stays in the same cell)} \end{cases} \tag{4.4}$$

while $k = 0$,

$$\begin{cases} p_{(e,t,0)(\alpha t+(1-\alpha)e,0,0)} = \frac{F(e,t+1)-F(e,t)}{1-F(e,t)}, & \text{(a new call comes)} \\ p_{(e,t,0)(e,t+1,1)} = q\frac{1-F(e,t+1)}{1-F(e,t)}, & \text{(no call, moves far)} \\ p_{(e,t,0)(e,t+1,0)} = p\frac{1-F(e,t+1)}{1-F(e,t)}, & \text{(no call, stays in the same cell)} \end{cases} \tag{4.5}$$

All other state transitions are of probability zero.

One objective of this chapter is to find a state feedback control law $f : y_t \rightarrow \mathbf{A} = \{0, 1\}$ that optimizes the system performance, where 0 means no update and 1 means update.

For a given state (e, t, k), if $f(e, t, k) = 1$, the mobile should update its location when state (e, t, k) is reached, which switches the system state to $(e, t + 1, 0)$ with probability one.

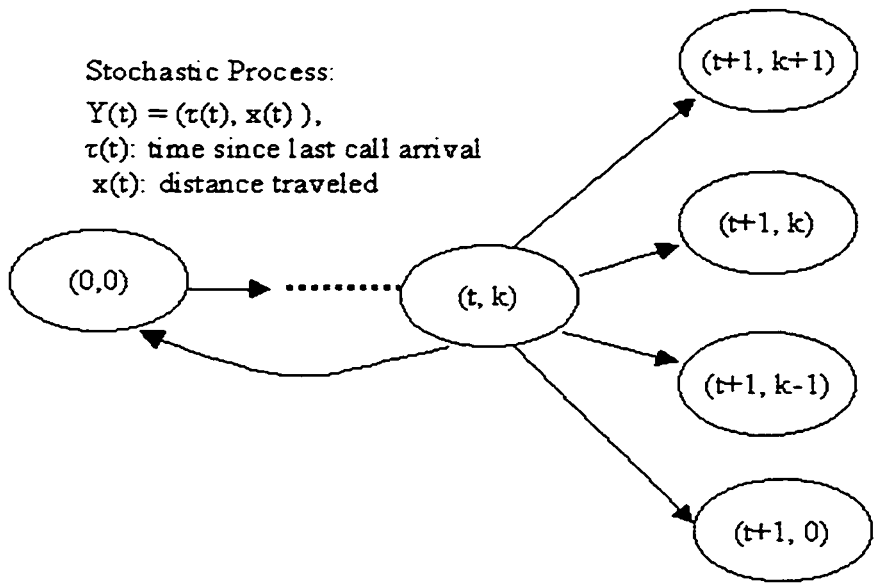

Figure 4.2: The state transition diagram

Note that the state component e_t of y_t changes only when a new call arrives which occurs much less frequently than that of the changes of τ_t and x_t. When we study the optimal control on (τ_t, x_t), we can consider e_t to be a constant. In the sequel of this chapter, system component e_t is not included in the system state. We assume that the interarrival time has a general distribution, with distribution function $F(\cdot)$.

Remark 4.3.2 *If the interarrival times are assumed to be independent and identically distributed, with geometric distribution, then the above model degenerates to a distance-based model.*

4.3.2 Two-Dimensional Model Description

Let us consider a hexagon cell model as shown in Figure 4.3. After current time slot, the mobile with probability p stays in the same cell, and with probability $q/6$ $(q = 1 - p)$ moves to one of its neighbor cells. As show in Figure 4.3, the cells are marketed with

$$S = \{(0,0), (1,0), (2,0), (2,1), (3,0), (3,1), (4,0), (4,1), (4,2), (5,0), (5,1), (5,2), \cdots\}$$

where (t, k) denotes a cell which is t cells away from its last updated location.

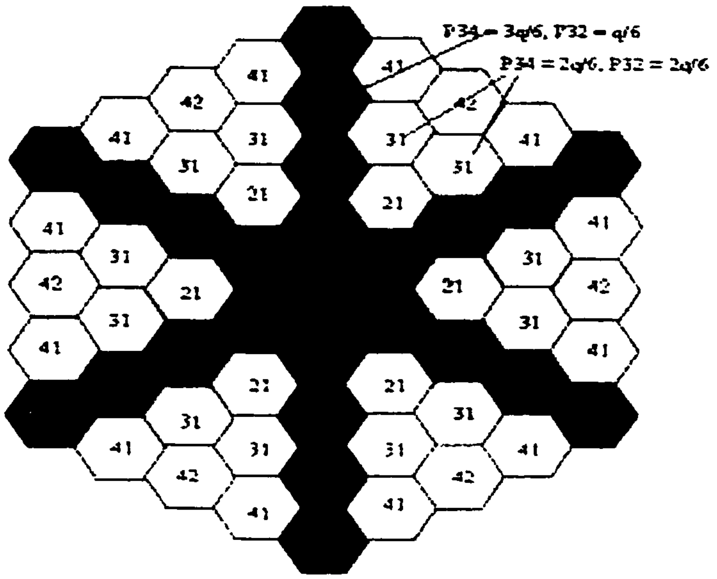

Figure 4.3: Hexagon Model

To develop a unified model with one-dimensional system, let us simplify the above model. Let $p_k = (\alpha_k, \beta_k, \gamma_k)$ be a probability vector, where

$$\gamma_k = P(\text{on ring k-1 at t+1 } | \text{on ring k at t})$$
$$\beta_k = P(\text{on ring k at t+1 } | \text{on ring k at t})$$
$$\alpha_k = P(\text{on ring k+1 at t+1 } | \text{ on ring k at t})$$

The ring of distance k from the last update location has $6k$ cells, among which six cells are marked as $k0$ as shown in Figure 4.3. The state transition probability p_k at these points is:

$$\begin{pmatrix} (1-p)/6 & 1-2(1-p)/3 & 3(1-p)/6 \end{pmatrix}$$

The other $6k - 6$ cells have transition probability vector:

$$\begin{pmatrix} 2(1-p)/6 & 1-2(1-p)/3 & 2(1-p)/6 \end{pmatrix}$$

Therefore, we have

$$\begin{aligned}\gamma_k &= P(\text{on ring k-1 at t+1} \mid \text{on ring k at t}) \\ &= \frac{6}{6k}P(\text{on ring k-1 at t+1} \mid \text{on ring k0 at t}) \\ &+\frac{6k-6}{6k}P(\text{on ring k-1 at t+1} \mid \text{on ring k but not k0 at t}) \\ &= \frac{1-p}{3} - \frac{1-p}{6k}\end{aligned}$$

Similarly,

$$\begin{aligned}\alpha_k &= P(\text{on ring k+1 at t+1} \mid \text{on ring k at t}) \\ &= \frac{6}{6k}P(\text{on ring k+1 at t+1} \mid \text{on ring k0 at t}) \\ &+\frac{6k-6}{6k}P(\text{on ring k+1 at t+1} \mid \text{on ring k but not k0 at t}) \\ &= \frac{1-p}{3} + \frac{1-p}{6k}\end{aligned}$$

The state space of the simplified two-dimensional movement model is much smaller than the original one. Moreover, the simplified model has the same structure as one dimensional model, so one-dimensional and two-dimensional models can be processed by the same manner.

4.4 Optimal Strategy

Process $\{y_t = (\tau_t, x_t), t \geq 0\}$ is a Markov process, with state space $\mathbf{S} = \{(n,d), 0 \leq n < \infty, 0 \leq d < \infty\}$. When a new call arrives and the network pages, (τ_t, x_t) switches to $(0,0)$, and an update action makes the system state switch from (t,k) to $(t+1,0)$. The difference between paging and updating lies in whether τ_t being reset to 0.

In the sequel of this section, we consider the process $y_t = (\tau_t, x_t), t \geq 0$. Assume all interarrival times are i.i.d. having distribution function $F(t)$. Denote by $P((\tau, x_t) = s'|(\tau, x_t) = s) \triangleq p_{ss'}$ the state transition probability. For a free system, the transition

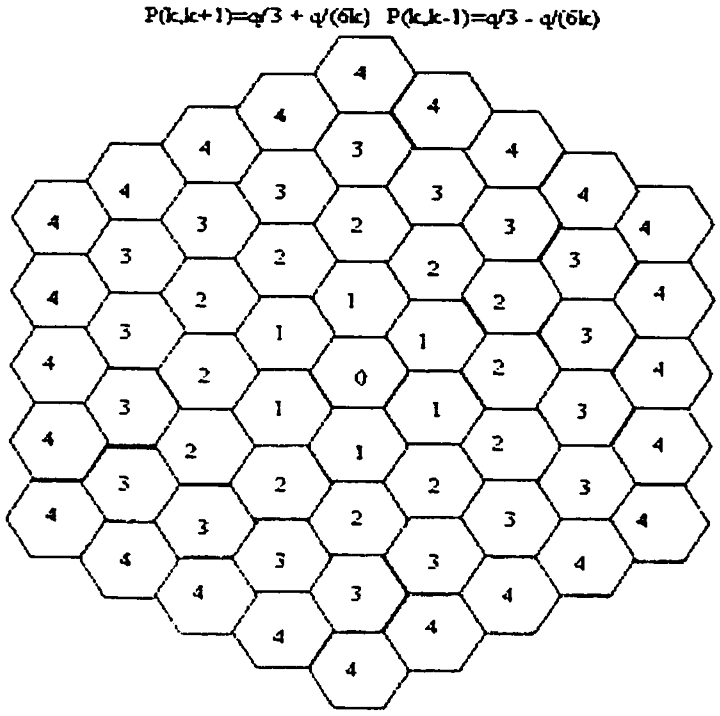

Figure 4.4: Simplified Hexagon Model

probabilities are given by: if $k > 0$

$$\begin{cases} p_{(t,k)(0,0)}(0) = \frac{F(t+1)-F(t)}{1-F(t)}, & \text{(a new call comes)} \\ p_{(t,k)(t+1,k+1)}(0) = \frac{q}{2}\frac{1-F(t+1)}{1-F(t)}, & \text{(no call, moves far)} \\ p_{(t,k)(t+1,k-1)}(0) = \frac{q}{2}\frac{1-F(t+1)}{1-F(t)}, & \text{(no call, and moves back)} \\ p_{(t,k)(t+1,k)}(0) = p\frac{1-F(t+1)}{1-F(t)}, & \text{(no call, stays in the same cell)} \end{cases} \tag{4.6}$$

while $k = 0$,

$$\begin{cases} p_{(t,0)(0,0)}(0) = \frac{F(t+1)-F(t)}{1-F(t)}, & \text{(a new call comes)} \\ p_{(t,0)(t+1,1)}(0) = q\frac{1-F(t+1)}{1-F(t)}, & \text{(no call, and moves far)} \\ p_{(t,0)(t+1,0)}(0) = p\frac{1-F(t+1)}{1-F(t)}, & \text{(no call, stays in the same cell)} \end{cases} \tag{4.7}$$

On state (t, k), if the mobile terminal takes action 1, i.e., update its location,

$$p_{(t,k)(t+1,0)}(1) = 1 \tag{4.8}$$

All other transitions are of probability zero.

Definition 4.4.1 *A function $f(\cdot) : \mathbf{S} \to \mathbf{A} = \{0, 1\}$ is said to be a stationary strategy. Let $\mathbf{F}$ be the set of all the stationary strategies.*

Let $p_{ss'}, s, s' \in \mathbf{S}$ denote the transition probability of $\{(\tau_t, x_t), t \geq 0\}$ and $p_{ss'}(f) = P((\tau_{t+1}, x_{t+1}) = s' | (\tau_t, x_t) = s, f(s))$ be the controlled transition probability under strategy f. Then $p_{ss'}(0)$ is the same as in (4.5)- (4.7), and

$$p_{ss'}(1) = \begin{cases} 1, & s'(2) = 0 \text{ and } s'(1) = s(1) + 1, \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

Let U be the cost of each update and V be the paging cost unit. Then, when the mobile is k cells far away from its last call arrival location, the paging cost is $(2k + 1)V$. The objective is to find an optimal strategy in $\mathbf{F}$, that is, to solve the following optimization problem:

$$\min_{f \in \mathbf{F}} v_f(s) = \mathbb{E}_f \left[\sum_{t=0}^{\infty} \gamma^t c(y_t, y_{t+1}) | y_0 = s \right] \tag{4.10}$$

where $\mathbb{E}_f$ represents the conditional expectation given that the strategy f is employed, $\gamma > 0$ is the discount factor, and $c((\tau_{t-1}, x_{t-1}), (\tau_t, x_t))$ denotes the cost incurred by transition $y_{t-1} \to y_t$, namely, $c(s, s') = 0$ except

$$c((t, k), (0, 0)) = V(2k + 1),$$
$$c((t, k), (t + 1, 0)) = U.$$

Remark 4.4.1 *Since $(0, 0)$ is a regenerative point of $\{y_t, t \geq 0\}$, we can formulate the optimization problem in another way by considering only one cycle. First, let us define cost functions $C(s, \alpha) : \mathbf{S} \times \mathbf{A} \to R$, and $h(s) : \mathbf{S} \to R$ by*

$$C((t, k), \alpha) = \begin{cases} U, & \alpha = 1 \\ 0, & \textit{otherwise} \end{cases}$$
$$h((t, k)) = (2k + 1)V.$$

Let τ denote the epoch of the next call arrival. With these notation, we can consider the following optimization problem

$$\min_{f \in \mathbf{F}} v_f(s) = \mathbb{E} \left[\sum_{t=0}^{\tau - 1} \gamma^t C(y_t, f(y_t)) + h(y(\tau)) \Big| y_0 = s \right]$$

In the above equation, τ is a random variable; with some mathematical manipulation, the above optimization problem can be transformed into a standard Markov decision problem with infinite time horizon and discounted cost.

Let the optimal value function be defined by

$$v(s) = \min_{f \in \mathbf{F}} v_f(s), \ \forall s \in \mathbf{S} \tag{4.11}$$

Then we obtain the following

Theorem 4.4.1 *The value function $v(s), s \in \mathbf{S}$ defined by (4.11) satisfies the following equations*

$$\begin{aligned} v(t,k) = \min \Big\{ & U + \gamma v(t+1,0); \frac{F(t+1)-F(t)}{1-F(t)}[(2k+1)V + \gamma v(0,0)] \\ & + \frac{q}{2}\frac{1-F(t+1)}{1-F(t)}[\gamma v(t+1,k+1)] + \frac{q}{2}\frac{1-F(t+1)}{1-F(t)}[\gamma v(t+1,k-1)] \\ & + p\frac{1-F(t+1)}{1-F(t)}[\gamma v(t+1,k)] \Big\}, \ if\, k > 0 \end{aligned} \tag{4.12}$$

$$\begin{aligned} v(t,k) = \min \Big\{ & U + \gamma v(t+1,0); \frac{F(t+1)-F(t)}{1-F(t)}[(2k+1)V + \gamma v(0,0)] \\ & + q\frac{1-F(t+1)}{1-F(t)}[\gamma v(t+1,k+1)] \\ & + p\frac{1-F(t+1)}{1-F(t)}[\gamma v(t+1,k)] \Big\}, \ if\, k = 0 \end{aligned} \tag{4.13}$$

Or, in another expression,

$$v(s) = \min_{a \in \mathbf{A}} \left\{ \sum_{s' \in \mathbf{S}} p_{ss'}(a)\,[c(s,s') + \gamma v(s')] \right\} \tag{4.14}$$

Proof: Let $f(\cdot)$ be an arbitrary strategy and the initial action a is chosen with probability r_a, then

$$\begin{aligned} v_f(s) &= \sum_{a \in \mathbf{A}} r_a \left[\mathbb{E}_f[c(y_0,y_1)|y_0 = s] + \mathbb{E}_f\left(\sum_{t=1}^{\infty} \gamma^t c(y_t,y_{t+1})|y_0 = s\right)\right] \\ &= \sum_{a \in \mathbf{A}} r_a \left[\sum_{s' \in \mathbf{S}} p_{ss'}(a) c(s,s') + \sum_{s' \in \mathbf{S}} p_{ss'}(a)\gamma \mathbb{E}_f\left(\sum_{t=1}^{\infty} \gamma^{t-1} c(y_t,y_{t+1})|y_1 = s'\right)\right] \end{aligned}$$

$$\begin{aligned}&\geq \min_{a\in \mathbf{A}}\left[\sum_{s'\in \mathbf{S}} p_{ss'}(a)c(s,s') + \gamma \sum_{s'\in \mathbf{S}} p_{ss'}(a)v(s')\right]\\ &= \text{ right hand side of (4.14),}\end{aligned} \tag{4.15}$$

implying

$$v(s) \geq \min_{a\in \mathbf{A}}\left\{\sum_{s'\in \mathbf{S}} p_{ss'}(a)\left[c(s,s') + \gamma v(s')\right]\right\}$$

because of the arbitrariness of f.

On the other hand, let a_0 satisfy

$$\sum_{s'\in \mathbf{S}} p_{ss'}(a_0)\left[c(s,s') + \gamma v(s')\right] = \min_{a\in \mathbf{A}}\left[\sum_{s'\in \mathbf{S}} p_{ss'}(a_0)\left[c(s,s') + \gamma v(s')\right]\right] \tag{4.16}$$

Let f be a strategy that chooses a_0 as the first step and then following another strategy g. If the next state is s', then view the process starting from s' with strategy g. By definition of $v(s)$, we can choose g such that

$$v_g(s') \leq v(s') + \varepsilon,$$

where $\varepsilon > 0$ is an arbitrary small constant. Therefore,

$$\begin{aligned}v_f(s) &= \sum_{s'\in \mathbf{S}} p_{ss'}(a_0)(c(s,s') + \gamma v_g(s'))\\ &\leq \sum_{s'\in \mathbf{S}} p_{ss'}(a_0)(c(s,s') + \gamma v(s')) + \gamma\varepsilon\\ &= \min_{a\in \mathbf{A}}\left\{\sum_{s'\in \mathbf{S}} p_{ss'}(a)(c(s,s') + \gamma v(s'))\right\} + \gamma\varepsilon\end{aligned}$$

which combined with $v(s) \leq v_f(s)$ yields

$$v(s) \leq \min_{a\in \mathbf{A}}\left\{\sum_{s'\in \mathbf{S}} p_{ss'}(a)\left[c(s,s') + \gamma v(s')\right]\right\} + \gamma\varepsilon \tag{4.17}$$

completing the proof of Theorem 4.4.1. Q.E.D.

Theorem 4.4.1 provides a characterization of the optimal value function by using a set of function equations. Next, we proceed to solve these functional equations. For this purpose, let us define $C(\mathbf{S}) = \{u | u : \mathbf{S} \rightarrow \mathbb{R}\}$ be a functional space with norm defined as $\|u\| = \sup_{s \in \mathbf{S}} |u(s)|$, and let us define an operator $\mathbf{T}$ on $C(\mathbf{S})$ by

$$(\mathbf{T}u)(s) = \min_{a \in \mathbf{A}} \left\{ \sum_{s' \in \mathbf{S}} p_{ss'}(a)(c(s, s') + \gamma u(s')) \right\} \tag{4.18}$$

then Theorem 4.4.1 implies $(\mathbf{T}v)(s) = v(s)$, which means that the optimal value function $v(\cdot)$ is a fixed point of $\mathbf{T}$.

Theorem 4.4.2 *Operator* $\mathbf{T}$ *defined by (4.18) is attractive.*

Proof: For any $u_1, u_2 \in C(\mathbf{S})$, let $\bar{a}$ be the action satisfying

$$\sum_{s' \in \mathbf{S}} p_{ss'}(\bar{a})(c(s, s') + \gamma u(s')) = \min_{a \in \mathbf{A}} \left\{ \sum_{s' \in \mathbf{S}} p_{ss'}(a)(c(s, s') + \gamma u(s')) \right\}.$$

Then, we have

$$\begin{aligned} (\mathbf{T}u_1)(s) - (\mathbf{T}u_2)(s) &= \min_{a \in \mathbf{A}} \left\{ \sum_{s' \in \mathbf{S}} p_{ss'}(a) \left[c(s, s') + \gamma u_1(s')\right] \right\} \\ &\quad - \min_{a \in \mathbf{A}} \left\{ \sum_{s' \in \mathbf{S}} p_{ss'}(a) \left[c(s, s') + \gamma u_2(s')\right] \right\} \\ &\leq \gamma \sum_{s' \in \mathbf{S}} p_{ss'}(\bar{a})(u_1(j) - u_2(j)) \\ &\leq \gamma \|u_1(j) - u_2(j)\| \end{aligned} \tag{4.19}$$

from which follows the proof of Theorem 4.4.2. Q.E.D.

Theorem 4.4.2 and Theorem 4.4.1 imply that $v(s)$ is the unique fixed point of $\mathbf{T}$; therefore, by the Fixed Point Theorem, it follows that for any $u \in C(\mathbf{S})$, $\lim_{n \to \infty} \mathbf{T}^n u = v$. With the optimal value function obtained, the following theorem gives a method to find the optimal strategy.

Theorem 4.4.3 *Let $v(s), s \in \mathbf{S}$ be the value function. Let $f(\cdot)$ be a strategy satisfying for any $s \in \mathbf{S}$,*

$$\sum_{s' \in \mathbf{S}} p_{ss'}(f(s)) \left[c(s, s') + \gamma v(s')\right] = \min_{a \in \mathbf{A}} \left\{ \sum_{s' \in \mathbf{S}} p_{ss'}(a) \left[c(s, s') + \gamma v(s')\right] \right\} \tag{4.20}$$

then $f(\cdot)$ is optimal.

Proof: The proof can be adapted from that of Theorem 6.3 of [27]. Q.E.D.

Example 4.4.1 *Consider a system as described above. The system parameters are as follows: The interarrival times have normal distribution with mean* 5 *and variance* 0.5, *the mobile with probability* $p = 0.4$ *stays in the same cell. The update cost* $U = 10$, *paging one cell costs* $V = 2$, *and discount rate* $\gamma = 0.95$. *With this set of data, the optimal update strategy is represented by an updating curve as plotted in Figure 4.5.*

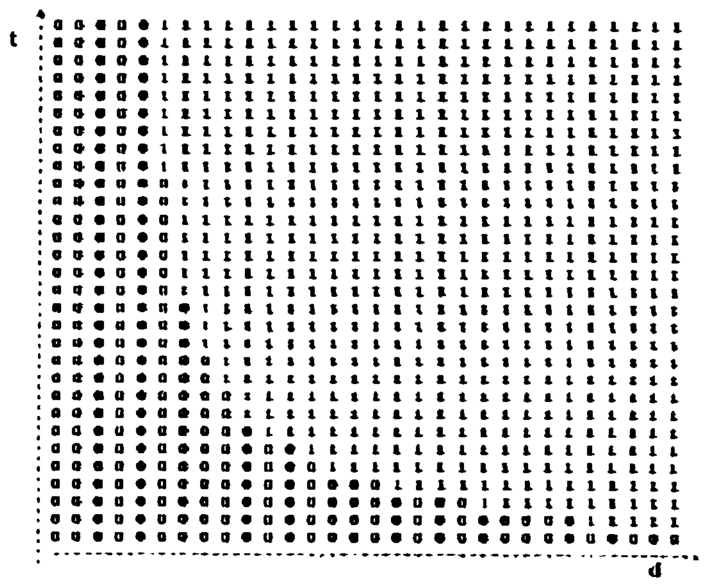

Figure 4.5: Optimal Update strategy, 0: not update, 1: update

Figure 4.5 shows that when the distance is less than 5, no update is needed. For any given d, the update strategy is a threshold control, that is, there is a time threshold value, once the elapsed time reaches this threshold location update is performed. For a given elapsed time, the optimal update strategy is also a threshold control with respect to the distance that the mobile terminal has moved since its last update. More precisely, if at point (t, k), it should be updated, then at all the points (t', k') with $t' \geq t$ and $k' \geq k$ it should also be updated.

Figure 4.5 reveals that there exists a convex function $t = y(d)$ to characterize the relationship of mobile's distance from its last updated location and the time elapsed since last call arrives, and the optimal update strategy is of the following structure:

$$f(t,k) = \begin{cases} 1 \text{ (update)}, & t \geq y(k), \\ 0 \text{ (no update)}, & t < y(k) \end{cases} \tag{4.21}$$

This particular structure of optimal control is very useful to represent the optimal strategy, and to develop suboptimal control and heuristic algorithms.

Instead of representing the optimal strategy using a huge matrix, we can denote the optimal strategy by a sequence of times $\mathbf{T} = \{t_1, t_2, \cdots\}$ satisfying $t_1 \geq t_2 \geq \cdots \geq t_D \geq 0$, where t_i is the threshold that the mobile user should update its location once it has traveled i cells away from its last update cell. For example, in the above numerical example:

$$\mathbf{T} = \{\infty, \infty, \infty, \infty, 21, 14, 11, 7, 6, 6, 5, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1\}.$$

Let D and T be the upper bound of distance and time considered. Then the original strategy space has 2^{T*D} strategies. However, the class of strategies with structure (4.21) has D^T strategies. So, restricted within the class of strategies with structure (4.21) can reduce the search task greatly.

4.5 Implementation Issues

4.5.1 Implementation Scheme

For any strategies, only if distance is involved, we have to calculate or estimate the distance the mobile terminal has moved since its last update. For a given network topology, from our bird eyes it seems a trivial task; however, a mobile is totally blind in the wireless network. In order to be able to calculate the traveled distance, an MT must learn the topology of the network. For this purpose, we introduce a concept of *chain code*, which is used in pattern recognition to represent the contour, skeleton, etc. of patterns. At any movement, the MT's six possible moving directions are denoted by '1, 2, 3, 4, 5, 6', which are called *chain code*. The trace of an MT is recorded by a sequence of chain codes, as shown in Figure 4.6. The sequence is said to be the chain code of the mobile terminal's trace.

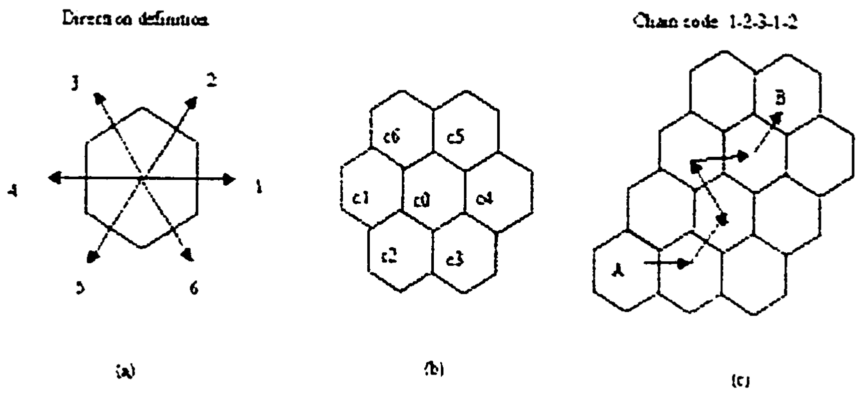

Figure 4.6: Mobile Movement Directions Definition

In Figure 4.6, (a) denotes the mobile movement direction. In (b), $c0, c1, \cdots, c6$ are cell IDs. Cell $c0$ periodically broadcast its own cell ID $c0$ and the list of its adjacent cell IDs ordered by $(c1, c2, \cdots, c6)$. (c) is a sample path of a mobile terminal moving from cell A to cell B. So, using the direction defined in (a), this trace can be coded as $1 - 2 - 3 - 1 - 2$.

The chain code is used to calculate the distance that the MT has traveled. The distance tracking mechanism consists of two elements:

- Each cell has a unique ID for measuring the distance traveled by the mobile terminal. Each cell periodically broadcasts its cell information

$$CellInfo\{cellID, listOfAjacency\}$$

  where $cellID$ is its own cell ID, and $listofAjacency$ is the list of its adjacent cell IDs. The list of the adjacency cell IDs has a fixed order and orientation, as shown in Figure 4.6.

- Each mobile terminal maintains a set of information denoted by $mobileInfo$:

$$mobileInfo\{timeSinceLastCall, preCellID, currCellID, traceArray, updateTimeList\}$$

where

- $timeSinceLastCall$ records the time passed since last call arrival
- $preCellID$ is the last visited cell ID
- $currCellID$ is the ID of the cell the mobile is presently in
- $traceArray$ is a list of chain code the mobile has traveled
- $updateTimeList$ is an array that represents the optimal strategy described previously.

Each time the MT receives a broadcast message from its current base station, the MT does the following:

- Step 1) Checks the broadcast cell ID.

```
if (currCellID == cellInfo.cellID ) // stay in the same cell as last time slot
   break;
else
   preCellID = currCellID;
   currCellID = cellInfo.CellID;
   directCode = 1; // chain code to be added to traceArray
end_if
// while loop to find the direction just moved
while (preCellID != cellInfo.adjacentList(directCode))
   directCode++;
end_while
   traceArray.AddLast(directCode);
end_if
```

- Step 2) Computes the distance traveled since last update by calling a subroutine that implements the Distance Computation Algorithm to operate on its *traceArray*. Let *curDistance* be the calculated distance from its last update.

– Step 3) With *curDistance* obtained in Step 2. the MT does the following

```
looks up its $updateTimeList(curDistance)$,
if $updateTimeList(curDistance) < timeSinceLastCall$, update;
else no update:
end_if
```

4.5.2 Distance Computation Algorithm

Once the chain code of an MT trace obtained, its traveled distance can computed as follows:

First, let code pairs $(1,4), (2,5)$. and $(3,6)$ cancel each other and delete them from the *traceArray*. Since 1 and 4 are opposite, if *traceArray* has a code 1 and a code 4. their contribution to the distance is canceled. so delete them from *traceArray*. Repeat this procedure until either all 1s or 4s or both are deleted from *traceArray*. The same procedure is done for code pairs $(2,5)$ and $(3,6)$.

After code cancelation, at most three different codes are left, denoted as *code*1. *code*2, and *code*3, and denote their quantities by *codeNumber*1, *codeNumber*2, *codeNumber*3. Without loss of generality, we assume

$$codeNumber1 \geq codeNumber2 \geq codeNumber3.$$

Note that in the remaining code, there may exist cycles, for example. there are two cycles in $traceArray = \{1-3-5-3-5-1-3\}$. So, the next step is to check if *code*1, *code*2, *code*3 can form a cycle. If the angle between *code*1 and *code*2, and the angle between *code*1 and *code*3 are $2\pi/3$, a cycle is formed. In this case, remove all *code*3 and delete *codeNumber*3 *code*1s and *code*2s from the *traceArray*. After cycles deletion, the sum of the number of *code*1 and the number of the one that has angle $\pi/3$ from it is the distance to be calculated. If the angle between *code*1 and *code*2, denoted by $\angle(code1, code2)$, and the angle between *code*1 and *code*3, denoted by $\angle(code1, code3)$, are $\pi/3$, then the distance is $codeNumber1 + codeNumber2 + codeNumber3$.

Algorithm 4.5.1 *(Distance Computation Algorithm)*

Step 1) Code cancelation. Code pairs $(1,4),(2,5),(3,6)$ *cancel each other and delete the canceled codes from the traceArray. Denote the remained codes as code*1, *code*2, *code*3 *(if three codes are left, otherwise, denoted as code*1, *code*2 *if two codes are left) and denote their quantities by codeNumber*1, *codeNumber*2, *codeNumber*3. *Without loss of generality, let us suppose* $codeNumber1 \geq codeNumber2 \geq codeNumber3$.

Step 2) Cycle deletion.

```
if ($\angle(code1, code1) == 2\pi/3$ && $\angle(code1, code3) == 2\pi/3$)
delete codeNumber3 code1s, code2s and code3s from traceArray
end_if
```

Step 3) Distance computation.

```
if ($\angle(code1, code2) == 2\pi/3$ && $\angle(code1, code3) == 2\pi/3$)
    $curDistance = codeNumber1 + codeNumber3$;
else if ($\angle(code1, code3) == 2\pi/3$ && $\angle(code1, code2) == 2\pi/3$)
    $curDistance = codeNumber1 + codeNumber2$;
else $curDistance = codeNumber1 + codeNumber2 + codeNumber3$;
end_if
```

To see how the above algorithm works, let us see an example

Example 4.5.1 *Consider a traceArray of length 10: traceArray* $= \{2,1,6,4,6,2,2,3,4,3,6\}$. *After first code cancelation, the remained trace is traceArray* $= \{2,2,2,4,6\}$. *Since 2,4,6 form a cycle, at the cycle deletion step delete 4, 6, and one 2. The left traceArray is* $\{2,2\}$, *so the distance is 2.*

4.6 Distance-Based Model and Its Variants

If the interarrival time is assumed to be of geometric distribution, the model becomes a distance-based model since the geometric distribution is memoryless. This section addresses the distance-based optimal strategy and its time-based invariant.

4.6.1 Distance-Based Optimal Strategy

Let us assume the interarrival times are independent and identically geometrically distributed with parameter λ. Define the system state $x_t \in \mathbf{S} = \{0, 1, 2, \cdots\}$ as the distance the MT has traveled since last registration. As before, let p denote the probability of an MT staying in the same cell as the previous time slot, then the transition probabilities are given by:

$$\begin{cases} p_{01} = q(1-\lambda), \quad p_{00} = 1 - q(1-\lambda) & \text{if } k = 0 \\ p_{10} = 1 - \frac{q}{2}(1-\lambda) - (1-\lambda)p, \quad p_{12} = \frac{q}{2}(1-\lambda), \quad p_{11} = (1-\lambda)p, & \text{if } k = 1 \\ p_{k0} = \lambda, \; p_{kk-1} = \frac{q}{2}(1-\lambda), \; p_{kk} = (1-\lambda)p, \; p_{kk-1} = \frac{q}{2}(1-\lambda), & \text{if } k > 1 \end{cases} \tag{4.22}$$

That is, P has the following structure.

$$P = \begin{pmatrix} p_{00} & p_{01} & 0 & 0 & 0 & 0 & 0 & \cdots \\ p_{10} & p_{11} & p_{12} & 0 & 0 & 0 & 0 & \cdots \\ p_{20} & p_{21} & p_{22} & p_{23} & 0 & 0 & 0 & \cdots \\ p_{30} & 0 & 0 & p_{32} & p_{33} & p_{34} & 0 & \cdots \\ p_{40} & 0 & 0 & 0 & p_{43} & p_{44} & p_{45} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots \end{pmatrix} \tag{4.23}$$

Obviously, in this case, we cannot distinguish between a new call arrival triggered state transitions and a location update triggered state transitions. So, we have to distinguish the state transitions in the cost function by considering the following cost:

$$v_f(d) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(f(t), x_t, x_{t+1}) \Big| x_0 = d\right],$$

where $c(a, i, j) = 0$ except $c(0, i, 0) = V(2i+1)$ (the paging cost), and $c(1, i, 0) = U$ (update cost).

Let $v(d) = \inf_{f \in \mathbf{F}} v_f(d)$ be the value function. Then we have the following theorem.

Theorem 4.6.1 *The optimal value function $v(d), d \in \mathbf{S}$ is the unique solution to the following equation:*

$$v(i) = \min_{a \in \mathbf{A}} \left\{ \sum_{j \in \mathbf{S}} p_{ij}(a) \left[c(a, i, j) + \gamma v(j)\right] \right\} \tag{4.24}$$

Theorem 4.6.2 *For the above discussed distance-based model, the optimal updating strategy is a threshold control, that is, there exists an integer i^* such that*

$$f(i) = \begin{cases} 0, & i \le i^* \\ 1, & i > i^* \end{cases}$$

Proof: First, let us prove that $v(i), i \in \mathbf{S}$, is increasing with respect to i. To this end, let us define a set of functions $v(i,d)$ as follows:

$$v(i,1) = \min\{c(1,i,0), p_{i0}V(2i+1)\},$$

which is, obviously, increasing in i. For $k > 1$, define

$$\begin{aligned} v(i,k) &= \min\left\{\sum_{j=0}^{\infty} p_{ij}(1)[c(1,i,j) + \gamma v(j,k-1)]; \sum_{j=0}^{\infty} p_{ij}[c(0,i,j) + \gamma v(j,k-1)]\right\} \\ &= \min\Big\{U + \gamma v(0,k-1); \lambda[(6i+1)V + \gamma v(0,k-1)] \\ &\quad + \frac{q(1-\lambda)}{2}\gamma v(i-1,k-1) + (1-\lambda)p\gamma v(i,k-1) + \frac{q(1-\lambda)}{2}\gamma v(i+1,k-1)\Big\} \end{aligned}$$

By assumption, $v(j,k-1)$ is increasing in j. Direct comparison gives

$$\begin{aligned} &\sum_{j=0}^{\infty} p_{ij}v(j,k-1) - \sum_{j=0}^{\infty} p_{i+1j}v(j,k-1) \\ &= p_{ii-1}v(i-1,k-1) + p_{ii}v(i,k-1) + p_{ii+1}v(i+1,k-1) \\ &\quad - p_{i+1i}v(i,k-1) - p_{i+1i+1}v(i+1,k-1) - p_{i+1i+2}v(i+2,k-1) \\ &= \frac{q(1-\lambda)}{2}[v(i-1,k-1) - v(i,k-1)] + (1-\lambda)p[v(i,k-1) - v(i+1,k-1)] \\ &\quad + \frac{q(1-\lambda)}{2}[v(i+1,k-1) - v(i+1,k-1)] \\ &\le 0 \end{aligned}$$

Combining the above two equations yields that $v(i,k)$ is increasing with respect to i for any k. Noting that $v(i) = \lim_{k\to\infty} v(i,k)$ proves that $v(i)$ is increasing.

Define

$$\begin{aligned} i^* = \max\Big\{i : \lambda[(6i+1)V + \gamma v(0)] + \frac{q(1-\lambda)}{2}\gamma v(i-1) \\ + (1-\lambda)p\gamma v(i) + \frac{q(1-\lambda)}{2}\gamma v(i+1) \le U + \gamma v(0)\Big\} \end{aligned}$$

Since $v(i)$ is increasing, from (4.6.1) we have

$$v(i) = \begin{cases} \lambda[(6i+1)V + \gamma v(0)] + \frac{q(1-\lambda)}{2}\gamma v(i-1) & \\ \qquad + (1-\lambda)p\gamma v(i) + \frac{q(1-\lambda)}{2}\gamma v(i+1), & i \leq i^* \\ U + \gamma v(0), & i > i^* \end{cases}$$

which means that the strategy with threshold i^* is optimal by Theorem 4.6.1. Q.E.D.

Example 4.6.1 *The following table 4.1 shows the optimal strategy versus call arrival rate λ and movement rate p.*

Table 4.1: Optimal Distance-Based Strategies

	$\lambda = 0.1$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
p=0.1	10	7	6	5	5	4	4	4	4
0.2	10	7	6	5	5	4	4	4	4
0.3	9	7	6	5	5	4	4	4	4
0.4	9	7	5	5	4	4	4	4	4
0.5	9	6	5	5	4	4	4	4	4
0.6	9	6	5	5	4	4	4	4	4
0.7	9	6	5	5	4	4	4	4	4
0.8	9	6	5	5	4	4	4	4	4
0.9	9	6	5	5	4	4	4	4	4

4.6.2 Time-Based Strategy

Although time-based strategy is not efficient as distance-based and movement-based strategies, due to its simplicity of implementation, it is of a lot of interests and importance. This section converts the optimal distance-based strategies into time-based strategies.

Let d_0 be the threshold of the optimal distance-based optimal strategy. We define a time threshold T_0 as the time at which the MT most likely first reaches d_0, in other words, T_0 is the solution of the following optimization problem:

$$\max_t P(x_t \text{ first reaches distance } d_0)$$

Let P be the state transition probability matrix of the distance-based model. Then, the probability of MT first reaching distance d_0 is:

$$P(\text{MT first reaches distance } d_0 \text{ at time } t) = P(x_t = d_0, x_s \neq d_0, 0 \leq s \leq t-1) \triangleq \rho(t).$$

Since the points where $\rho(t)$ reaches maximum may be not unique, we take the smallest one as our optimal threshold value, that is,

$$T_0 = \min\{t : \rho(t) \geq \rho(s), 1 \leq s < \infty\}.$$

Let $P_{0/d_0\cdot}$ be a row vector obtained from P by deleting the d_0-th element from the 0-th row of P, and $P_{\cdot d_0}$ be the column vector obtained by deleting d_0-th element in the d_0-th column of P. Define matrix $\tilde{P}_{d_0}$ to be a matrix obtained by deleting d_0-th row and d_0-th column from P. Then, we have

$$t = 1 : \rho(1) = p_{0d_0}$$
$$t \geq 2 : \rho(t) = P_{0/d_0\cdot}\left[\tilde{P}_{d_0}^{t-2}\right] P_{\cdot d_0}$$

Example 4.6.2 *Using the above scheme, the distance-based optimal strategies are converted into time-based strategies, which are shown as in the following table.*

Table 4.2: Optimal Time-Based Strategies

	$\lambda = 0.1$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$p = 0.1$	29	12	7	4	2	1	1	1	1
$p = 0.2$	32	14	9	6	6	3	3	3	3
$p = 0.3$	29	16	11	7	7	3	3	3	3
$p = 0.4$	34	19	8	8	4	4	4	4	4
$p = 0.5$	40	16	10	9	5	5	5	5	4
$p = 0.6$	49	19	12	11	6	6	6	5	5
$p = 0.7$	62	25	15	14	8	7	7	6	6
$p = 0.8$	85	34	21	18	10	9	8	7	6
$p = 0.9$	100	52	31	26	14	13	11	9	8

Chapter 5

On-Line Location Update Optimization

This chapter develops a simulation-based optimization algorithm for the MT location update problem. For this purpose, we treat the system under one strategy as the normal system and the another one as a perturbated system. The idea is to observe the trajectory of the normal system, use the observed trajectory to construct a trajectory of the perturbated system, and thus estimate their performance indices. That is, use a single sample with respect to a given strategy, e.g., f_1, to construct a sample path of another strategy, e.g. f_2, by which the performance difference between f_1 and another one f_2 is estimated. If the estimated result reveals that f_1 is better, continue using strategy f_1 and compare it with another one, f_3. On the other hand, if the simulation reveals f_2 is better than f_1, switch the strategy f_1 to f_2. Let the system evolve according to f_2 and estimate its performance difference with a new one, and so on, until no improvement can be made any more.

5.1 Sample Path Construction Algorithm

To illustrate the idea of how to construct a sample path of a different stochastic process based on the observed sample path of a given process, let us see a simple example.

Example 5.1.1 *Let $\{x_t, t \geq 0\}$ is a Markov process with state space* $\mathbf{S} = \{1, 2, 3\}$ *and state transition matrix* $P = \begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.4 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0.1 \end{pmatrix}$, *and $\{x'_t, t \geq 0\}$ is another Markov process with state space* $\mathbf{S}$ *and state transition matrix* $P_1 = \begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.4 & 0.4 & 0.2 \\ 1.0 & 0 & 0 \end{pmatrix}$. *Next, we proceed to construct a trajectory of x'_t based on the observation of x_t.*

Note that P differs from P_1 only in the 3rd row, so the trajectory of x'_t may deviate from that of x_t only when state 3 is visited. Since x_t and x'_t are Markov processes, they are also regenerative processes and state 3 is one of their regenerative point. So, we construct the trajectory of x'_t from that of x_t piece by piece. The following algorithm is to cut pieces of x_t trajectory, and paste them to x'_t trajectory.

Sample Path Construction Procedure:

Step 1) Simulate x_t until it reaches state 3 and record the observed path as a sample path of x'_t.

Step 2) Record state 3 for x'_t and continue simulation without adding to the path of x'_t until state 1 is reached.

Step 3) Continue observing x_t and concatenate its trajectory to that of x'_t until state 3 is reached.

Step 4) Go to Step 2.

```
Observed Sample path of x_t with 11 cycles

mcp=

Columns 1 through 12

3    1    3    3    2    2    2    2    2    2    1    3
```

```
Columns 13 through 24

2    3    2    1    3    2    2    2    1    3    2    1

Columns 25 through 36

2    2    3    3    2    1    2    2    2    1    2    2

Columns 37 through 48

2    1    3    2    2    1    2    2    1    1    3    2

Columns 49 through 57

2    1    2    1    2    3    2    2    1


Constructed path of x'_t

mcp1 =

Columns 1 through 12

3    1    3    1    3    1    3    1    3    1    2    2

Columns 13 through 24

3    1    2    2    2    1    2    2    2    1    3    1

Columns 25 through 35

2    2    1    1    3    1    2    1    2    3    1
```

In order to check the constructed trajectory is a sample path of a Markov process with transition matrix P_1, we need the following important property of Markov process.

Proposition 5.1.1 *Let $\{x_t, t \geq 0\}$ be a Markov process with state space* $\mathbf{S}$ *and transition matrix $P = (p_{ij}), i, j \in \mathbf{S}$. Let $S_n(i,j)$ and $S_n(i)$ be the number of the occurrence* *of pair (i,j) and the number of occurrence of state i in $x_0, x_1, \cdots, x_n$. Then, we have*

$$\lim_{n\to\infty} \frac{S_n(i,j)}{S_n(i)} = p_{ij}, \quad w.p.1$$

To verify that the above constructed path truly has transition matrix P_1, an experiment with 4000 cycles of simulation is performed. The observed path of x_t and constructed path x'_t are of 16108 and 11246 steps lengths, respectively.

The observed path estimates the transition probability matrix *Pest* of x_t and the estimation errors $Pest - P$ are

```
Pest =

    0.1945    0.2928    0.5127

    0.4096    0.3940    0.1964

    0.3960    0.5112    0.0928


Pest-P =

   -0.0055   -0.0072    0.0127

    0.0096   -0.0060   -0.0036

   -0.0040    0.0112   -0.0072
```

Using the constructed path estimates the transition probability matrix of x'_t and the estimation errors

```
Pest1 =

    0.1945    0.2928    0.5127

    0.4078    0.3912    0.2010
    1.0000         0         0


Pest1-P1 =

   -0.0055   -0.0072    0.0127

    0.0078   -0.0088    0.0010

         0         0         0
```

The above experiment shows that the constructed trajectory has transition matrix P_1. Let us proceed to see how to construct a path of x_t based on an observed sample path of x'_t.

Definition 5.1.1 *Let $f, g \in \mathbf{F}$ be two stationary strategies. $|f|$ is defined as the number of states $s \in \mathbf{S}$ such that $f(s) = 1$. The number of positions in which the corresponding binary values of f and g are different is said to be the* **Hamming distance** *between f and g, denoted by $H(f, g)$.*

In the sequel of this chapter, a controlled Markov process with control strategy f is denoted by $\{y(t, f), t \geq 0\}$. Let f and g be two stationary strategies satisfying $H(f, g) = 1$. Suppose $f(s_0) = 0$ and $g(s_0) = 1$. Denote by s'_0 be the state reachable from s_0 under strategy g. Using the following algorithm, a sample path of Markov process with control g, denoted by $\{y(t, g), t \geq 0\}$, can be constructed from that of process $y(t, f)$.

Algorithm 5.1.1 *(Sample Path Construction Algorithm 1)*

Step 1) Observe $y(t, f)$ and add the observed states to $y(t, g) = s_0$ until s_0 is reached.

Step 2) Record state s_0 on $y(t, g)$, and observe $y(t, f)$ without adding states to $y(t, g)$ until s'_0 is observed.

Step 3) Continue observing $y(t, f)$ and add its trajectory to $y(t, g)$ until s_0 is reached.

Step 4) Go to Step 2.

The above algorithm constructs a sample path of $y(t,g)$ from observed sample path of $y(t,f)$. However, construction of a sample path of $y(t,f)$ from an observed sample path of $y(t,g)$ is more complicated, as shown in the following Sample Path Construction Algorithm 2.

For this purpose, let us first introduce some notations. Let states $i_1,\cdots,i_r \in \mathbf{S}$ be the states reachable from s_0 in one-step under a strategy f that does not update at s_0, that is, $p_{s_0 i_j} > 0, j = 1,\cdots,r$. Let $\tilde{p}_{s_0}$ be the probability that s_0 occurs in a cycle of $y(t,f)$ with length greater than 1, then

$$\tilde{p}_{s_0} = p_{s_0\cdot}\left[\sum_{t=1}^{\infty}\tilde{P}^t_{s_0 s_0}\right]p_{\cdot s_0} = p_{s_0\cdot}\left[I - \tilde{P}_{s_0 s_0}\right]^{-1}p_{\cdot s_0}$$

where $p_{s_0\cdot}$ is the row vector obtained from the s_0-th row of the transition matrix $P(f)$ by deleting its s_0-th column, $p_{\cdot s_0}$ is the column vector obtained from the s_0-th column of the transition matrix $P(f)$ by deleting its s_0-th row, and $\tilde{P}_{s_0 s_0}$ is the matrix obtained from $P(f)$ by deleting its s_0-th row and s_0-th column. Since the cycle of $y(t,f)$ is longer than $y(t,g)$, we have to add some segments to the sample of $y(t,g)$ so that it becomes a path of $y(t,f)$. The set of state segments to be used are denoted by

$$pad(i_1), pad(i_2), \cdots, pad(i_r),$$

and

$$padc(i_1), padc(i_2), \cdots, padc(i_r).$$

They are small pieces of states copied from the observed sample path of $y(t,g)$. $pad(i_j)$ is a string of states copied from a piece of the sample path of $y(t,g)$ that starts with i_j and ends with s'_0 without including s'_0, e.g., if $i_1\alpha_1\cdots\alpha_m s'_0$ is a piece of sample path of $y(t,g)$, define $pad(i_1) = i_1\alpha_1\cdots\alpha_m$; $padc(i_j)$ is a string of states copied from a piece of the sample path of $y(t,g)$ that starts with i_j and ends with s_0 without including s_0, e.g., if $i_1\alpha_1\cdots\alpha_m s_0$ is a piece of sample path of $y(t,g)$, define $padc(i_1) = i_1\alpha_1\cdots\alpha_m$. When necessary, we choose a pad from $pad(i_1),\cdots,pad(i_r)$, and insert it between s_0 and s'_0; and choose one from $padc(i_1),\cdots,padc(i_r)$ and insert it into a 1-step length cycle, that is, insert it between the possible state pair s_0s_0. Once a pad is used, it will be refilled in the following cycles.

Algorithm 5.1.2 *(Sample Path Construction Algorithm 2)*

Step 0) Compute probability $\bar{p}_{s_0}$, and find $i_1, i_2, \cdots, i_r$;
initialize $pad(i_1), pad(i_2), \cdots, pad(i_r)$, $padc(i_1), padc(i_2), \cdots, padc(i_r)$

Step 1) Simulate $y(t,g)$ until state s_0 is reached, add the observed states to $y(t,f)$'s path under construction;

Step 2) With probability $\bar{p}_{s_0}$ do the following: select one state i_0 from $i_1, \cdots, i_r$ with probability $p_{s_0i_1}, \cdots, p_{s_0i_r}$, and add $padc(i_0)$ to $y(t,f)$.

Step 3) Generate a random integer R_g using geometric distribution with parameter $p_{s_0s_0}$, and add a string $s_0 \cdots s_0$ with length R_g to $y(t,f)$.

Step 4) With probability distribution $p_{s_0i_1}, \cdots, p_{s_0i_r}$, choose a state i_0 from $i_1, \cdots, i_r$, add $pad(i_0)$ to $y(t,f)$.

Step 5) Let $NextState$ = next state of $y(t,g)$, and let $CurrentState = NextState$.

```
while $CurrentState == s_0'$
  Add $CurrentState$ to $y(t,f)$;
  $CurrentState$ =next state of $y(t,g)$;
end_while
```

Step 6) Set Indicator $Ind(i_j) = 0, j = 1, \cdots, r$; Initialize $pd(i_1), \cdots, pd(i_r)$.

```
while $CurrentState! = s_0$ && $CurrentState! = s_0'$
  if $Ind(CurrentState) == 0$
    $Ind(CurrentState) = 1$;
  end_if
if $Ind(Current) == 1$
  Add $CurrentState$ to $pd(CurrentState)$;
  Add $CurrentState$ to $y(t,f)$;
end_if
  $CurrentState$ = Next state of $y(t,g)$;
end_while
```

Step 7)

```
if ($CurrentState == s_0$)
  for ($i = 1; i <= r; i++$)
  if ($pad(i_j)$ is used)
    $pad(i_j) = pd(i_j)$
  end_if
  end_for
else
for ($i = 1; i <= r; i++$)
  if ($padc(i_j)$ is used)      $padc(i_j) = pd(i_j)$
  end_if
end_for
end_if
```

Step 8) if the stop rule is satisfied, stop; otherwise go to Step 2.

To see illustrate the Algorithm 5.1.2, let us see the following example.

Example 5.1.2 *Suppose there are two Markov processes: $\{x_t, t \geq 0\}$ and $\{x'_t, t \geq 0\}$ with state space $\mathbf{S} = \{1, 2, 3, 4\}$. Their transition probability matrices are*

$$P = \begin{pmatrix} 0.2 & 0.3 & 0.2 & 0.3 \\ 0.4 & 0.2 & 0.2 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.1 \\ 0 & 0.3 & 0.4 & 0.3 \end{pmatrix}, \quad P_1 = \begin{pmatrix} 0.2 & 0.3 & 0.2 & 0.3 \\ 0.4 & 0.2 & 0.2 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{5.1}$$

Using Algorithm 5.1.2, an experiment is performed and the results are listed as follows: With 500 periods simulated, we get a sample of x'_t with length 5432 steps, using Algorithm 5.1.2 a sample of x_t with length 8033 steps is constructed. From the simulated sample path, the estimated transition probability matrix for x'_t is

```
Pest1 =

    0.2004    0.3034    0.2012    0.2950

    0.4163    0.2034    0.1982    0.1821
    0.2974    0.4944    0.1029    0.1053

    1.0000         0         0         0


P1-Pest1 =

   -0.0004   -0.0034   -0.0012    0.0050

   -0.0163   -0.0034    0.0018    0.0179

    0.0026    0.0056   -0.0029   -0.0053

         0         0         0         0


Pest =

    0.1997    0.3171    0.2157    0.2675

    0.4637    0.2057    0.2170    0.1136

    0.3397    0.4830    0.1202    0.0571

         0    0.3158    0.4036    0.2806


P-Pest =

    0.0003   -0.0171   -0.0157    0.0325

   -0.0637   -0.0057   -0.0170    0.0864

   -0.0397    0.0170   -0.0202    0.0429

         0   -0.0158   -0.0036    0.0194
```

5.2 Location Update Online Optimization Algorithm

In order to present the online optimization algorithm, let us give a definition.

Definition 5.2.1 *Let $f \in \mathbf{F}$ be a stationary strategy. A neighborhood of f, denoted by $\delta(f)$, is the collects of all the stationary strategies that has Hamming distance* 1 *with f, that is,*

$$\delta(f) = \{g | g \in \mathbf{F}, H(f,g) = 1\}. \tag{5.2}$$

In order to design an on-line algorithm, let us consider the average cost function, that is,

$$J_f = \limsup_{n\to\infty} \mathbb{E}_f \left[\frac{1}{n} \sum_{t=0}^{n-1} c(y_t, y_{t+1}, f) \right] \tag{5.3}$$

We need the following properties of regenerative processes (see Ross [27]).

Proposition 5.2.1 *Assume $\{y(t,f), t \geq 0\}$ is ergodic. Let $y(0,f) = s_0$ and $\{\xi_t, t \geq 1\}$, with $\xi_0 = 0$, be the sequence of times at which state s_0 is visited. Then, we have*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} c(y(t,f), y(t+1,f)) = \frac{\mathbb{E}\left[\sum_{t=0}^{\xi_1} c(y(t,f), y(t+1,f))\right]}{\mathbb{E}[\xi_1]} \tag{5.4}$$

with probability one, and

$$\lim_{n\to\infty} \mathbb{E}_f \left[\frac{1}{n} \sum_{t=0}^{n-1} c(y(t,f), y(t+1,f)) \right] = \frac{\mathbb{E}_f\left[\sum_{t=0}^{\xi_1} c(y(t,f), y(t+1,f))\right]}{\mathbb{E}[\xi_1]} \tag{5.5}$$

From the above proposition, we have the following theorem

Proposition 5.2.2

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} \left[\sum_{t=\xi_{i-1}}^{\xi_i - 1} c(y_t, y_{t+1}, f)\right]}{\sum_{i=1}^{n} \xi_i} = \lim_{n\to\infty} \frac{\frac{1}{n}\sum_{i=1}^{n} \left[\sum_{t=\xi_{i-1}}^{\xi_i - 1} c(y_t, y_{t+1}, f)\right]}{\frac{1}{n}\sum_{i=1}^{n} \xi_i} = J_f \tag{5.6}$$

with probability one.

The similar estimation technique (2.10) is used to estimate $\mathbb{E}\left[\sum_{t=0}^{\epsilon_1} c(y(t,f),y(t+1,f))\right]$. The following algorithm estimates J_f and J_g based on observation of $y(t,f)$.

Algorithm 5.2.1 *(Performance Estimation Algorithm)*

Step 1) Initialization: Set error bound ε, $\bar{C}_f = \bar{C}_g = \bar{T}_f = \bar{T}_g = 0$;

Step 2) Observe $y(t,f)$ until state s_0 is reached; set $i = 1$.

Step 3) Set $C_f = C_g = T_f = T_g = 0$;

```
while $y(t,f) \neq s_0$
  $C_f = C_f + c(y_t, y_{t+1})$;
  $T_f = T_f + 1$;
  $t = t + 1$;
end
while $y(t,f) \neq s_0$
  $C_f = C_f + c(y_t, y_{t+1})$;
  $T_f = T_f + 1$;
  $C_g = C_g + c(y_t, y_{t+1})$;
  $T_g = T_g + 1$;
  $t = t + 1$;
end
$i = i + 1$;
$\bar{C}_{fe} \triangleq \frac{1}{i}C_f + \frac{i-1}{i}\bar{C}_f$;
$\bar{T}_{fe} \triangleq \frac{1}{i}T_f + \frac{i-1}{i}\bar{T}_f$;
$\bar{C}_{ge} \triangleq \frac{1}{i}C_g + \frac{i-1}{i}\bar{C}_g$;
$\bar{T}_{ge} \triangleq \frac{1}{i}T_g + \frac{i-1}{i}\bar{T}_g$;
```

Step 4) If $\max\{|\bar{C}_{fe} - \bar{C}_f|, |\bar{T}_{fe} - \bar{T}_f|, |\bar{C}_{ge} - \bar{C}_g|, |\bar{T}_{ge} - \bar{T}_g|\} \geq \varepsilon$,
$\bar{C}_f = \bar{C}_{fe}, \bar{T}_f = \bar{T}_{fe}, \bar{C}_g = \bar{C}_{ge}, \bar{T}_g = \bar{T}_{ge}$, and go to Step 3;

Step 5) $J_f = \frac{\bar{C}_f}{\bar{T}_f}$, $J_g = \frac{\bar{C}_g}{\bar{T}_g}$ and $J_f - J_g = \frac{\bar{C}_f}{\bar{T}_f} - \frac{\bar{C}_g}{\bar{T}_g}$.

Algorithm 5.2.1 shows how to estimate the performance indices of strategies f and g based on observation of sample path $y(t,f)$ without recording the sample paths of $y(t,f)$ and $y(t,g)$. By the same way, we can estimate the performance indices of f, g based on observation of sample path $y(t,g)$.

For any two stationary strategies $f, g \in \mathbf{F}$ satisfying $H(f, g) = 1$, Algorithm 5.1.1 constructs a sample path of $y(t, g)$ based on observation of $y(t, f)$, and Algorithm 5.1.2 constructs a sample path of $y(t, f)$ based on observation of $y(t, g)$. So, using these two algorithm all sample paths corresponding to strategies in the neighborhood of a given strategy can be constructed. The following algorithm can find a suboptimal strategy.

Let $N = |\mathbf{S}|$, then for a given strategy there are N strategies in its neighborhood.

Algorithm 5.2.2 *(Strategy Optimization Algorithm)*

```
Step 1) Choose initial strategy $f$

Step 2) Let $\delta(f) = \{f_1, \cdots, f_N\}$. Observe $y(t, f)$. Based on observation of $y(t, f)$,

    for $i=1:N$
    if $|f_i| > |f|$, estimate $J(f_i)$ using Algorithm 5.1.2
    end_if
    if $|f_i| < |f|$, estimate $J(f_i)$ using Algorithm 5.1.1
    end_if
    end_for

Step 3) if $J(f) = \min\{J(f_1), \cdots, J(f_N)\}$, stop;
    else if $J(f_j) = \min\{J(f_1), \cdots, J(f_N)\}$, $f = f_j$ go to Step 2.
    end_if
```

Example 5.2.1 *To illustrate Algorithm 5.2.2, let us work an example. Consider a system with movement rate $p = 0.35$, call arrival rate $\lambda = 0.65$, $U = 7, V = 2$. Set initial strategy with threshold $d_0 = 30$. Algorithm 5.2.2 yields the optimal strategy with threshold $d_0 = 7$.*

5.3 Online Implementation

To implement the above algorithm, we suppose the same implementation mechanism as that of optimal update strategy proposed in Section 4.5. We assume each

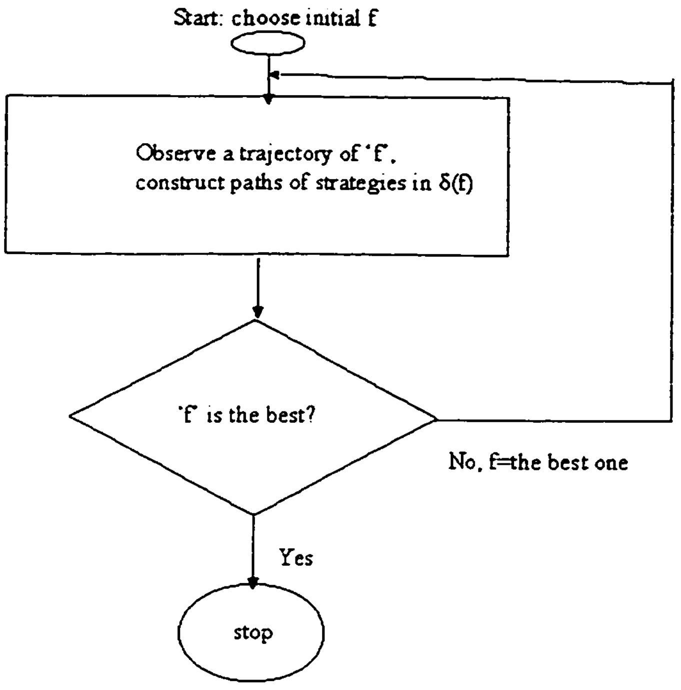
Start: choose initial f
Observe a trajectory of 'f', construct paths of strategies in δ(f)
'f' is the best?
No, f=the best one
Yes
stop

Figure 5.1: On-Line Optimization Algorithm Diagram

base station maintains its own cell information as described in Section 4.5. Each mobile terminal maintains a set of information denoted by *mobileInfo*:

$$mobileInfo\{timeSinceLastCall, preCellID, currCellID, traceArray, preState,$$
$$currentState, currentStrategy, listOfPerformance\}$$

where

- *timeSinceLastCall, preCellID, currCellID, traceArray* are as described in Section 4.5
- *preState* is the system state (time and distance) in last time slot
- *currentState* is the current system state
- *currentStrategy* is the strategy in use, which is represented by a matrix
- *listOfPerformance* records the estimated performance values for all the strategies in the neighborhood of the *currentStrategy*

Receiving the broadcasted $cellInfo$, the MT does the following to seek an suboptimal strategy.

Step 1) When the MT receives the broadcasted $cellInfo$, it checks the $cellInfo$, computes the distance traveled by the same way described in Section 4.5, calculates the new system state, and sets $preState = currentState$ and $currentState =$ the new state.

Step 2) Looks up the $currentStategy$ to decide whether to update:
If $currentStrategy(currentState) == 1$, update; otherwise, no update.

Step 3) Uses Algorithm 5.2.1 and Algorithm 5.2.2 to update the estimated values of $listOfPerformance$.

Step 4) If the observation is long enough, the MT switches the $currentStrategy$ to the one that has the best performance value in $listOfPerformance$, and initializes the $listOfPerformance$.

Remark 5.3.1 *If we consider the distance model, the MT maintains*

$$mobileInfo\{preCellID, currCellID, traceArray, preState,$$
$$currentState, currentStrategy, listOfPerformance\}$$

where $currentStrategy$ is represented by a binary vector.

Chapter 6

Epilogue

6.1 Summary of this thesis

This thesis applied the discrete event dynamical systems theory and methods to wireless communication systems, which consists of two parts. Part I studied continuous parameter optimization problem of continuous-Markov process using uniformization. An algorithm to estimate the sensitivity of the system performance of a Markov process with respect to its design parameters is proposed and a gradient-based algorithm is used to optimize the system performance; as an application of this algorithm, Chapter 3 studied the call request buffering problem in wireless communication systems.

Part II studied the mobile location tracking problem. The phone call arrival, mobile movement, and location registration are characterized using Markov decision processes, and the optimal location update strategies are solved using the Markov dynamical programming algorithm. The model considers the distance the MT has traveled since last registration and time elapsed since last call arrival. To implement the proposed update scheme, an interactive implementation scheme was proposed. In this scheme, the network broadcasts its cell information, based on which the mobile terminals learn the topology of the network and calculate the distance they have traveled. For this purpose, a new scheme to compute the traveled distance was proposed, which is useful for any distance-involved update models. Finally, a simulation-based optimization is developed to optimize the update strategies.

In this thesis, we have studied the optimal update strategy, distance-based strategy, time-based strategy, and an online algorithm. Among them, the optimal strategy is, of course, the best one, and is also the most difficult one to implement; the distance-based strategy is easier to handle because of the assumption of geometric distribution of the call interarrival times; the time-based strategy is the easiest one to implement, but it is not as efficient as the first two because it is converted from the optimal distance-based strategy. The online optimization algorithm can be implemented online, but it can only produce a local optimal strategy.

6.2 A New Markov Movement Model

In the previous chapters, the mobile terminal's mobility was modeled by a random walk model, that is, at each time slot, a mobile terminal with the same probability enters one of its neighbor cells. Its movements at different time slots are independent. Obviously, this is an average case. Intuitively, the moving direction in next time slot is dependent on its current moving direction. In this section we propose a new model which models the MT's moving direction by a Markov process. That is, its next step moving direction depends only on its current moving direction. For this purpose, let us label the six directions by $0, 1, 2, 3, 4, 5$, as shown in Figure 6.1

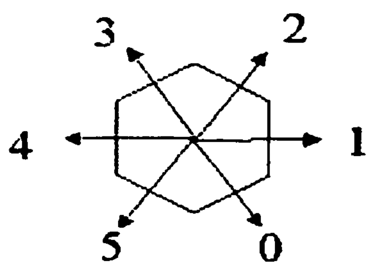

Figure 6.1: The Movement Direction Definition

Let $S = \{0, 1, 2, 3, 4, 5\}$ and $p_i, i = 1, \cdots, 4$ be four nonnegative constants satisfying $\sum_{i=1}^{4} p_i = 1$. p_1 designates the probability of keeping the same direction, p_2 moving at a direction with angle $\pi/3$ with current direction, p_3 moving at direction with angle $2\pi/3$, and p_4 moving at the opposing direction. Let r_t denote the direction at which the MT is moving at epoch t. Then,

$$P(r_{t+1} = j | r_t = i) \triangleq p_{ij} = \begin{cases} p_1, \text{ if } j = i \\ p_2, \text{ if } j = [i+1]_6 \text{ or } j = [i+5]_6 \\ p_3, \text{ if } j = [i+2]_6 \text{ or } j = [i+4]_6 \\ p_4, \text{ if } j = [i+3]_6 \end{cases} \tag{6.1}$$

where $[\cdot]_6$ is the operation of taking mode 6. Obviously, it is inadequate to make update decision according the current moving direction. So, we need more information to comprise of the system state. Let $Z_+ = \{0, 1, 2, \cdots\}$ be the set of nonnegative integers and put $\mathbf{S} = Z_+^6 \times S$. Then define a stochastic process $x_t, t \geq 0$ with $x_t \in Z_+^6$, where $x_t(i)$ denotes the number of steps the MT has traveled in direction i. Define

$$\begin{aligned} q_i(x, x') &= P(x_{t+1} = x' | x_t = x, r_t = i) \\ &= \begin{cases} q, & \text{if } x' = x + e_i \\ 1 - q, & \text{if } x' = x \\ 0, \text{else} \end{cases} \end{aligned} \tag{6.2}$$

where $e_i \in Z_+^6$ with all elements being 0 except its ith element. With these notations given, we define stochastic process $y_t = (x_t, r_t), t \geq 0$. Then $\{y_t, t \geq 0\}$ is a Markov process with transition probability given by

$$P(y_{t+1} = (x', i') | y_t = (x, i)) = p_{ii'} q_i(x, x'). \tag{6.3}$$

Let τ be a random time denoting the next phone call arrival time and $f(\cdot) : \mathbf{S} \longrightarrow \{0, 1\}$ be a strategy with 0 meaning no update and '1' update. And let $\mathbf{F}$ be the set of stationary strategies. Then the optimization problem is

$$\min_{f \in \mathbf{F}} J(f) \triangleq \mathbb{E} \left[\sum_{t=0}^{\tau-1} c(y_t, f(y_t)) + h(y_\tau) \right] \tag{6.4}$$

where $c(y, 1)$ is the cost of update when reaching state y, $c(y, 0) = 0$; and $h(y_T)$ is the paging cost. This is a standard Markov decision problem that can be solved using Markov dynamical programming algorithm, such at successive value iteration or policy improvement.

However, the challenge of solving the optimization lies in its large state space. A direct method to handle this is to aggregate all the states with the same distance and develop a distance based algorithm. This algorithm is heuristic since the process with aggregated state is not a Markov process any more. To develop an efficient algorithm for the above optimization problem is still under investigation.

6.3 Future Research Directions

Finally, we point out some future research directions:

- Continuous flow models and simulation-based optimization for telecommunication networks. Queuing system models for high speed telecommunication networks may be daunting because of the prohibitive computing requirements. This is due to the fact that in the packet-switching network, packets arrival and departure from a node constitute discrete events, and a simulation program processes a sequence of such events. For instance, in ATM networks, the packets consisting of 53 bytes are transmitted over lines operating at rates of hundreds of megabits per second; therefore, there could occur millions of events per second at each node. To handle this problem, we can model packets arrival and departure as continuous flows by the same way of modeling the job arrival and departure in flexible manufacturing systems.

- For digital communication systems, the received signal fading, caused by the propagation conditions of a mobile radio channel, leads to transmission errors. The occurrence of errors is greater during the signal fading. To represent adequately the error distribution in a real digital communication channel, Gilbert has proposed [14] an error generation model based on a two-state Markov chain with two states $\{E_1, E_2\}$. The first state E_1 represents a binary symmetrical channel with a zero error probability p_1 while the second state indicates a binary symmetrical channel having a crossing probability p_2. The errors generation process is determined by the transition probabilities p_{ij} from a state to the subsequence state. This model has been justified by many authors (see [33, 34] and references therein).

Combining the Markov channel model and the continuous flow model leads to Markov jump systems [2], so using Markov jump systems to model telecommunication networks is a promising research direction. Especially, the stochastic systems with time-delay can be used to characterize the delay in the telecommunication networks.

Bibliography

[1] I. F. Akyildiz, J. Ho, and Y. B. Lin, Movement-based location update and selective paging for PCS networks, *IEEE/ACM Trans. Networking*, vol. 4, pp. 629-638, 1996.

[2] E. K. Boukas and Z. K. Liu. *Deterministic and Stochastic Time Delay System*, Birkhauser Boston, Feb., 2002.

[3] K. L. Chung, *Markov Process with Stationary Transition Probabilities*, Berlin: Springer-Verlag, 1960.

[4] I. F. Akyildiz and J. S. M. Ho, Dynamic mobile user location update for wireless PCS networks, *ACM/Baltzer, J. Wireless Networks*, vol. 1, no.2, pp.187-196, 1995

[5] Amotz Bar-Noy, I. Kessler, and S. Moshe, Mobile users: to update or not to update?, *ACM/Baltzer, J. Wireless Networks*, vol. 1, No. 2, pp. 175-195, 1995

[6] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Boston, 1999.

[7] X. R. Cao and H. F. Chen, Perturbation realization, potentials, and sensitivity analysis of Markov process, *IEEE Transactions on Automatic Control*, vol. 42, No. 10, pp. 1382-1893, 1997.

[8] E. Cinlar, Introduction to Stochastic Processes, Prentice Hall, Inc., 1975.

[9] P. G. Escalle, V. C. Giner, and J. M. Oltra, Reducing location update and paging costs in a PCS network, *IEEE Transactions on Wireless Communications*, vol. 1, No. 1, pp. 200-209, 2002.

[10] L. Y. Dai, Perturbation analysis via coupling, *IEEE Transactions on Automatic Control*, vol. 45, No. 4, pp. 614-628, 2000.

[11] L. Y. Dai and Y. C. Ho, Structural infinitesimal perturbation analysis (SIPA) for derivative estimation of discrete event dynamic systems, *IEEE Transactions on Automatic Control*, vol. 40, No. 12, pp. 1154-1166, 1995.

[12] J. Doob, Stochastic Processes, Wiley, New York, 1953.

[13] D. Freedman, *Markov Chain*, Springer-Verlag, New York, 1983.

[14] E. N. Gilbert, Capacity of a burst-noise channel, *Bell Syst. Tech. J.*, vol. 39, pp. 1253-1265, 1960.

[15] P. Glasserman, Gradient Estimation via Perturbation Analysis, Norwell, MA: Kluwer, 1991.

[16] Y. B. Lin and W. Chen, Call request buffering in a PCS networks, *Mobicom*, pp.585-592, 1994

[17] Z. K. Liu and F. S. Tu, Single Sample-Based sensitivity analysis of Markov processes using uniformization, *IEEE Transactions on Automatic Control,* No. 4, pp. 872-875, 1999.

[18] U. Madhow, M. Honig, and K. Steiglitz, Optimization of wireless resources for personal communications mobility tracking, *Proc. of IEEE INFOCOM'94*, pp.577-584, 1994.

[19] H. G. Gene and C. D. Meyer, Using the QR factorization and group inversion to Compute, differentiate, and estimate the sensitivity of stationary probabilities for Markov chains, *SIAM, Alg. Disc. Math.* vol. 7, pp. 273-281, 1986.

[20] Z. Naor and H. Levy, Minimizing the wireless cost of tracking mobile users: an adaptive threshold scheme, *Proc. IEEE InFOCOM'98*, San Francisco, Ca, pp. 720-727, 1998.

[21] L. Kleinrock, *Queueing Systems* Volume 1: Theory, John Wiley and Sons, New York, 1975

[22] B. Liang and Z. Hass, Predictive distance-based mobility management for PCS networks, *Proc. IEE INFOCOM'99*, New York, 1999

[23] W. S. Wong and C. M. Leung, An adaptive distance-based location update algorithm for next-generation PCS networks, *IEEE J. on Selected Areas in Communications*, Vol. 19, No. 10, pp. 1942-1952. 2001.

[24] G. P. Pollini et. al., A profile-based location strategy and its performance, *IEEE JSAC*, vol.15, no. 8, pp.1415-24, 1997.

[25] C. Rose, State-based paging/registration: a greedy technique, *IEEE Trans. Veh. Technol.*, vol. 48, pp.166-173, 1999.

[26] C. Rose, Minimizing the average cost of paging and registration: a timer-based method, ACM/Baltzer J. Wireless Networks, vol 2, No. 2, pp. 109-116, 1996.

[27] S. M. Ross, *Applied Probability Models with Optimization Applications*, Holden-Day, 1970.

[28] S. K. Sen, A. Bhattacharya, and S. K. Das, A selective location update strategy for PCs users, *ACM/Baltzer J. Wireless Networks*, vol. 5, No. 5, pp. 313-326, 1999.

[29] W. Wang and I. F. Akyildiz, Intersystem location update and paging schemes for multitier wireless networks, 2000, preprint.

[30] Z. Naor, Tracking mobile users with uncertain parameters, 2000, preprint.

[31] I. F. Akyildiz, J. McNair, J. Ho, H. Zunalioglu, and W. Wang, Mobility management in next-generation wireless systems, *Proceeding of the IEEE,* vol.87, pp. 1347-1384, 1999

[32] V. Wong and V. Leung, Location management for next generation personal communication networks, *IEEE network*, vol. 14, pp. 18-24, 2000.

[33] H. S. Wang, On verifying the first-order Markovian assumption for a Rayleigh fading channel model, *IEEE Trans. Veh. Tech.*, vol. VT-45, pp. 353-357, 1996.

[34] M. Zorzi, R. R. Rao, and L. B. Milstein, On the accuracy of a first-order Markov model for data block transmission on fading channels, in *Proc. IEEE ICUPC'95*, pp. 211-215, Nov. 1995.