AN ABSTRACT OF THE THESIS OF

Thai Duong for the degree of Master of Science in Computer Science presented on May 20, 2015.

Title: Data Collection in Sensor Networks via the Novel Fast Markov Decision Process Framework

Abstract approved: _____

Thinh P. Nguyen

We investigate the data collection problem in sensor networks. The network consists of a number of stationary sensors deployed at different sites for sensing and storing data locally. A mobile element moves from sites to sites to collect data from the sensors periodically. There are different costs associated with the mobile element moving from one site to another, and different rewards for obtaining data at different sensors. Furthermore, the costs and the rewards are assumed to change abruptly. The goal is to find a "fast" optimal movement pattern/policy of the mobile element that optimizes for the costs and rewards in non-stationary environments. We formulate and solve this problem using a novel optimization framework called Fast Markov Decision Process (FMDP). The proposed FMDP framework extends the classical Markov Decision Process theory by incorporating the notion of mixing time that allows for the trade-off between the optimality and

the convergence rate to the optimality of a policy. Theoretical and simulation results are provided to verify the proposed approach.

Data Collection in Sensor Networks via the Novel Fast Markov Decision Process Framework

by

Thai Duong

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 20, 2015
Commencement June 2015

Master of Science thesis of Thai Duong presented on May 20, 2015.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Thai Duong, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Thinh Nguyen for his extraordinary guidance and support. I also thank my committee members Prof. Raviv Raich, Prof. Prasad Tadepalli, and Prof. Yevgeniy Kovchegov for serving on my committee and reviewing this work.

I also thank my colleagues in my group for their useful and interesting discussions. I thank all my friends in Oregon State University and in Corvallis for helping me with my life in America.

Finally, I would like to express my special thanks to my parents, my sister and my wife for their supports and encouragements.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1: Introduction

Wireless Sensor Networks (WSNs) have been deployed in many applications, including but not limited to, tracking and monitoring of cattle in agriculture, studies of species migration patterns in biological sciences, and detecting hazardous conditions. A typical WSN consists of a large number of battery-operated sensors that sense and collect data from the environment. The data are then sent to a sink node via a multi-hop routing scheme [1]. As a result, the sensors that are closer to the sink will consume more energy for relaying data from other sensors. And since communication energy constitutes a large percentage of the overall energy consumption, the batteries of these sensors will drain much earlier than others. Consequently, the life time of the entire sensor network is shortened significantly.

A number of strategies have been proposed to address this energy problem [2–5]. One approach is to not use the multi-hop communication. Instead, one or more mobile elements (MEs) are proposed to travel around in the network and collect data from the sensors. The MEs then hand off the collected data to the sink when they are in its vicinity. This approach not only enhances the life time of the sensor network, but also allows data collection in the environments where wireless communication are not efficient or possible, e.g., when sensors are geographically far apart or when they are placed in a forest where the water in the trees can attenuate the wireless signals significantly. On the other hand, relying on MEs

to deliver the data increases the latency and the chance of buffer overflow at a sensor node due to the possibility that the ME does not visit a node often enough to collect its data. Therefore, there has been a number of algorithms to find an optimal movement pattern of the ME that maximizes the amount of data collected in a given period of time while satisfying constraints on latency and buffer overflow [2, 4, 5].

That said, an optimal movement pattern or policy is determined by a number of parameters including (1) the cost associated with the ME moving from one sensor to another and (2) the various rates of useful data being collected at different sensors. In many cases, finding an optimal movement policy can be cast as a classic Markov Decision Process (MDP) problem [6]. The solution to the MDP problem is a policy that tells the ME where to go next given that it is currently at a certain location so that over time, the optimal policy will maximize a pre-specified objective. We note that an optimal MDP policy aims to maximize the given objective in the long run under the assumption of stationary environments. On the other hand, many real world settings are non-stationary, where the parameters can vary unexpectedly. Therefore, the optimal movement policy should also be designed not only to maximize the objective but also to be robust against these unexpected changes. In this thesis, we formalize the problem of finding the optimal movement policy for the ME in non-stationary environments as a random walk on a graph [7, 8] and solve it using a novel Fast Markov Decision Process framework (FMDP). The proposed FMDP framework extends the classical MDP theory by incorporating the theory of mixing time that allows for the trade-off between the

optimality of a policy and how fast the optimality can be obtained. Thus, in non-stationary environments, a FMDP policy might be preferable since it can achieve the given objective quickly before the environment changes. For example, if the costs of traveling between two sensors modeled as the distance between them are changed abruptly due to bad weather, or the rates of data being gathered at a sensor change significantly, then it is better to use the movement policy produced by the FMDP framework. We will quantitatively show these results in Chapter 6. In addition, we show that under certain assumptions, finding the optimal movement policy of the ME is in fact a convex problem, and can be solved effectively. As such, unlike many existing heuristic approaches, the optimality of the solution of the proposed algorithm can be quantified and obtained.

Our thesis is organized as follows. Chapter 2 presents the related literature on sensor networks. In Chapter 3, we provide a brief background on MDP, mixing time, and random walk on a graph, necessary for the development of the proposed solution approach. In Chapter 4, we describe the proposed FMDP framework and formalize the problem in its context. An algorithmic solution is proposed in Chapter 5. Chapter 6 provides the simulation results to demonstrate the benefits of the proposed approach. Finally, the Chapter 7 gives a few concluding remarks.

Chapter 2: Related Work

There is a rich research literature on WSNs. We can only mention a few in this thesis. A good survey of WSNs can be found in [1]. Many WSNs are assumed to be dense. Thus, data collected by the sensors are sent to the sink using multi-hop routing schemes. As discussed in the Introduction, this approach severely limits the longevity of a WSN. It might not even be feasible in some cases. As such, MEs are introduced into WSNs. For example, mobile data collectors such as mobile sinks [2, 9, 10] or mobile relays [3, 11–13] are used to collect data from the sensors periodically. They move around, connect to the sensors, and collect the data. In other applications such as animal tracking, the sensors are attached to the cattle, and thus are naturally mobile [14, 15].

There are two models for ME's mobility: uncontrolled mobility and controlled mobility. For the uncontrolled mobility, the movement of the MEs is either deterministic and cannot be changed [16], or follows a random distribution in trajectory and speed [17]. For controlled mobility, the trajectory and speed of an ME can be actively altered and controlled to a certain extent. There have been number of studies on this controlled mobility model. The goal is to control the mobility element's speeds and trajectories to maximize the amount of data collected while minimizing latency and buffer overflow at the sensors. For example, many algorithms for designing deterministic trajectories have been proposed in [2,18–21]. For

dynamic trajectories, there have been two main schemes to dynamically change the ME's trajectories. The first scheme is based on sensor's demand [4, 13]. The ME will change its course when a node request is received. The second scheme is based on node's priority which depends on the node's buffer overflow status [5]. Several speed control algorithms have also been proposed, including Stop to Collect, Adaptive Speed Control [22] and a heuristic control algorithm in [23], etc. Some algorithms for controlling both trajectories and speeds have also been investigated in [9, 24].

Since the proposed FMDP framework is based on the classic MDP and the theory of mixing time, we briefly discuss related work on these two topics. Markov Decision Processes (MDPs) have been widely studied and applied in many fields including machine learning, control theory, and communications. In 1957, Bellman proposed his well-known Bellman's equation for Markov Decision Processes in [25]. Good sources of MDPs can be found in books by Howard, Bertsekas, and Puterman [6, 26, 27]. Algorithms to find optimal policies such as Value Iteration and Policy Iteration have been well-studied. A generalized version of MDP called Partially Observable MDP (POMDP) which allows uncertainty in the MDP states is discussed in [28]. In the field of Reinforcement Learning, POMDP is introduced into Policy Gradient Method to estimate the gradient [29]. Regarding MDP in changing environment, [30] studied the Value Iteration algorithm for MDP in Adiabatic Settings, i.e. the environment is slowly changing and converges to a final state.

The idea of using mixing time to find fast policies is studied for several appli-

cations. In [31], it is shown that the convergence rate/mixing time of a Markov Chain is characterized by its second largest eigenvalue modulus (SLEM). Thus, the fastest mixing Markov Chain can be obtained by minimizing the SLEM [31]. For queuing systems, we can find fast queuing policies to achieve a desired stationary distribution by minimizing the SLEM via convex relaxation [32]. In [33], fast policies of collecting data in sensor networks with mobile elements are obtained by including mixing time or SLEM as a trade-off term in the objective function of the formulated optimization problem. In [34], the mixing time is considered as a regularization term to accelerate the learning phase of reinforcement learning.

Unlike many existing approaches with heuristic flavors, we cast the data collection problem in the framework of a novel Fast Markov Decision Process and solve it using the convex optimization. As such, we can provide guarantees of optimality of the proposed solution.

Chapter 3: Preliminaries

3.1 Markov Decision Process (MDP)

The MDP is a framework for studying optimal decision making process under uncertainty [6]. In an MDP setting, there is a controller who interacts with the environment by taking actions based on its observations at every discrete decision epoch. Each action by the controller induces a change in the environment. Typically, the environment is described by a finite set of states. An action will move the environment from the current state to some other states with certain probabilities. Associated with each action in each state is a reward given to the controller. The goal of the controller is to maximize the total expected reward or average reward over some finite or infinite planning horizon by making sequential decisions based on its current observations in certain settings.

Formally, a discrete-time MDP represents a dynamic system and is specified by a finite set of states S, representing the possible states of the system, a set of control actions A, a transition probability matrix $\mathbf{P}^{|S|\times|S|}$, and a reward function $\mathcal{R}$. The transition probability matrix $\mathbf{P}$ specifies the dynamics of the system. Each entry $\mathbf{P}(i,j) \triangleq \mathbf{P}(s_{n+1} = j | s_n = i, a_n = a)$ represents the conditional probability of the system moving to state $s_{n+1} = j$ in the next decision epoch after taking an action a in the current state $s_n = i$. The dynamics are Markovian in the

sense that the probability of the next state j depends only on the current state i and the action a, and not on any previous history. The reward function $\mathcal{R}(s, a)$ assigns a real number to the state s and the action a so that $\mathcal{R}(s, a)$ represents the immediate reward of being in state s and taking action a. A policy Π is a sequence of actions a_1, a_2, $\ldots a_n$ taken by the controller where n denotes the time index. Formally a policy specifies a mapping from states to actions at each decision epoch $\Pi_n : S \rightarrow A$. The policy Π that is called stationary if its actions depend only on the state s, independent of time index. A stationary policy induces a time-invariant transition probability matrix. Typically, time is assumed to be discrete and that the control policy selects one action at each decision epoch. Every policy Π is associated with a value function V^{Π} such that $V^{\Pi}(s)$ gives the objective by Π when starting in state s. The solution to an MDP problem is the optimal policy Π^* that maximizes the average reward given by:

$$V^{\Pi}(s) = E(\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^{N} \mathcal{R}(s_n, a_n)),$$

or the discounted reward in infinite planning horizon given by:

$$V^{\Pi}(s) = E(\sum_{n=1}^{\infty} \alpha^n \mathcal{R}(s_n, a_n)),$$

or the total reward in the finite planning horizon given by:

$$V^{\Pi}(s) = E(\sum_{i=1}^{N} \mathcal{R}(s_n, a_n)).$$

In this thesis, we use the average reward to be defined in Section 4.3 as a criterion for optimal policy.

3.2 Mixing Time

Proposition 1 (Stationary Distribution [8, 35]). *For an irreducible, aperiodic, finite and discrete Markov chain with transition probability matrix* $\mathbf{P}$*, there exists a unique stationary distribution* $\boldsymbol{\pi}$ *such that*

$$\lim_{n\to\infty} \boldsymbol{\nu}^T \mathbf{P}^n = \boldsymbol{\pi}^T, \tag{3.1}$$

where ν^T *denotes the transpose of a vector* ν*.*

In other words, the chain will converge to a stationary distribution of states after a large number of steps regardless of the initial state $\boldsymbol{\nu}$.

Definition 1 (Total variation distance [8, 35]). *For any two probability distributions* $\boldsymbol{\nu}$ *and* $\boldsymbol{\pi}$ *on a finite state space* Ω*, we define the total variation distance as:*

$$\|\boldsymbol{\nu} - \boldsymbol{\pi}\|_{TV} = \frac{1}{2} \sum_{i\in\Omega} |\boldsymbol{\nu}(i) - \boldsymbol{\pi}(i)| .$$

The total variation distance measures the distance between two distributions which represents the maximum difference between probabilities of an event assigned by the two distributions [8]. It is used to evaluate the convergence rate of the Markov Chain to the stationary distribution.

Definition 2 (Mixing time [8]). *For a discrete, aperiodic and irreducible Markov chain with transition probability* $\mathbf{P}$ *and stationary distribution* $\boldsymbol{\pi}$*, given an* $\epsilon > 0$*, the mixing time* $t_{mix}(\epsilon)$ *is defined as*

$$t_{mix}(\epsilon) = \inf\left\{n : \|\boldsymbol{\nu}^T\mathbf{P}^n - \boldsymbol{\pi}^T\|_{TV} \leq \epsilon, \textit{for all probability distributions } \boldsymbol{\nu}\right\}.$$

The mixing time characterizes the convergence rate of the Markov Chain to its stationary distribution. The smaller the mixing time is, the faster the convergence rate is.

Definition 3 (Reversible Markov Chain [8, 35]). *A discrete Markov chain with a transition probability* $\mathbf{P}$ *is said to be reversible if*

$$\mathbf{P}(i,j)\boldsymbol{\pi}(i) = \mathbf{P}(j,i)\boldsymbol{\pi}(j) \tag{3.2}$$

Theorem 1 (Bound on mixing time [8]). *Let* $\mathbf{P}$ *be the transition matrix of a reversible, irreducible and aperiodic Markov chain with state space* S*, and let* $\boldsymbol{\pi}_{min} := min_{x\in S}\boldsymbol{\pi}(x)$*. Then*

$$t_{mix}(\epsilon) \leq \frac{1}{1-\mu(\mathbf{P})}\log\left(\frac{1}{\epsilon\boldsymbol{\pi}_{min}}\right). \tag{3.3}$$

where $\mu(\mathbf{P})$ *is the second largest eigenvalue modulus (SLEM) of matrix* $\mathbf{P}$

As seen in the Theorem 1, the larger the spectral gap $(1-\mu(\mathbf{P}))$ is, the faster the

chain converges to the stationary distribution. Specifically, the error ϵ decreases over time with rate bounded by $\frac{e^{-(1-\mu(\mathbf{P}))t}}{\boldsymbol{\pi}_{min}}$. As a result, the spectral gap $1-\mu(\mathbf{P})$ will be used in our FMDP framework to characterize the convergence rate of a policy.

3.3 Random Walk on Graphs and Reversible Transition Matrix

Random walk on graphs is a probabilistic model that describes the random movement of a walker from one vertex to another on a given graph. Specifically, the movement is modeled locally in the following sense. If the walker is at a certain vertex V, the probability that it will move to a particular neighboring vertex depends only on the weights of edges connecting to V. It is shown that any reversible transition matrix can be represented as a weighted random walk on graph [7, 8]. Suppose a matrix $\mathbf{P}$ is reversible with stationary distribution $\boldsymbol{\pi}$ or $\boldsymbol{\pi}(s)\mathbf{P}(s,s') = \boldsymbol{\pi}(s')\mathbf{P}(s',s) = \mathcal{C}(s,s')$. Then we can use the conductance $\mathcal{C}(s,s')$ as the weight of the edge (s,s').

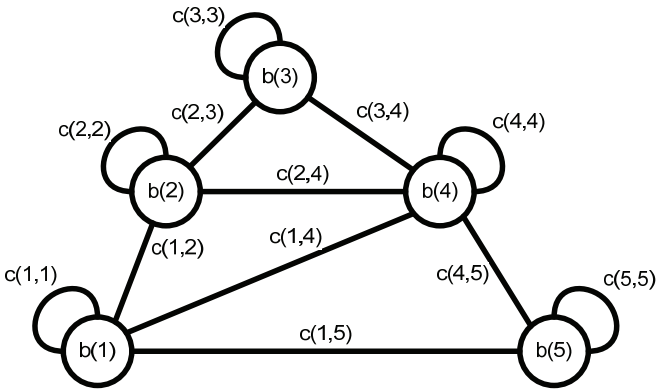
A weighted graph can be described by a symmetric weight matrix $\mathbf{W}$ where its positive entry $\mathbf{W}(s,s') = \mathbf{W}(s',s)$ is the weight of the edge (s,s'). Let $H(s) = \sum_{s'} \mathbf{W}(s,s')$ be the total weight of all edges connected to node s. If $H(s) = 0$, then the node s does not connect to any other nodes and we can ignore node s when calculating transition probability. Specifically, the probability of moving from node s to node s' will be $\mathbf{P}(s,s') = \frac{\mathbf{W}(s,s')}{H(s)}$ if $H(s) > 0$ and 0 if $H(s) = 0$. The stationary distribution for the transition matrix is calculated as $\boldsymbol{\pi}(s) = \frac{H(s)}{\sum_k H(k)}$.

Note that the transition matrix and its stationary distribution only depend on the ratio between weights. Therefore, without loss of generality, we normalize the sum of all the weights in the weight matrix, i.e. $\sum_{(i,j)} \mathbf{W}(i,j) = \sum_i H(i) = 1$.

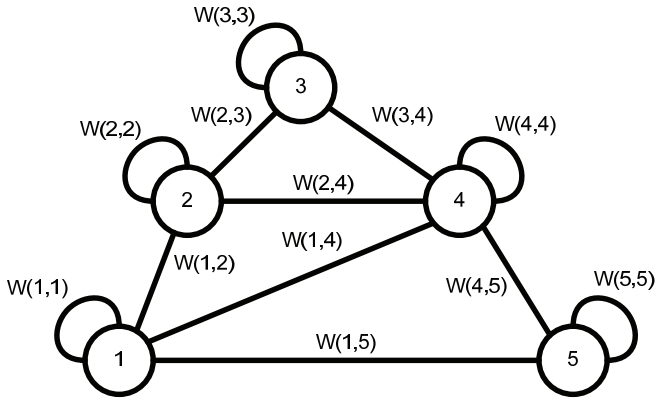
Chapter 4: Problem Modeling and Formulation

4.1 Modeling

We model a sensor network as a graph (V, E) (Figure 4.1a) with the set of vertices or nodes V representing the sensors or groups of sensors close to each other such that the ME can collect the data from all nodes in the same group at the same time. The edges in E represent the path the ME can travel from one node to another node to collect data in discrete time step/decision epoch. Once the ME reaches to a node i, it collects data from that node and obtains a benefit modeled as a real positive number $b(i)$. One way the value $b(i)$ can be interpreted is the amount of data collected at node i. Thus, if the rate of useful data being sensed and stored at a node i per unit time is high then $b(i)$ should be large. In addition to rewards, there are costs. Specifically, traveling from one node to another incurs a cost $c(i, j)$ which can represents among many quantities, the energy spent to travel the actual distance between the two sensors i and j. The self-loop in a node represents the cost of staying in the node after collecting data. This self-loop models the situation where data have already been collected, so if the ME stays at the same node at the next decision epoch, there is a high chance that there is little data to collect, and thus wasting precious time to collect fresh data from other nodes. Finally, the benefits at each node and costs on each edge might change



(a) Reward and cost structures on a graph



(b) Corresponding random walk on the graph

Figure 4.1: An example of graph to model sensor network and its random walk induced by an MDP policy

abruptly.

Objective. The goal is to find an optimal movement policy of the ME that maximizes the average reward defined as the difference between the benefit and the cost per unit time, but the policy must be robust to changes in costs and benefits, i.e., it should obtain reasonably high average reward. We will make this statement precise shortly when describing the FMDP framework.

4.2 Random Walk on Graph and Probabilistic MDP Policy

An optimal MDP policy in our problem is the rule on how the ME moves to another node given that it is currently at a certain node. Assume that the ME is at some node, there might be several neighboring nodes that the ME can move to. Each way represents an action a as described in Section 3.1. For example, if ME is currently at node 5 in Figure 4.1b, then for one policy, an action a_1 could be "going to node 1" and for another policy, the action a_2 is "going to node 4". One can also employ a probabilistic policy that includes an action a_3 such as "going to node 1 with probability 0.4, going to node 4 with probability 0.6." As a result, we can compute the cost of selecting an action a based on the cost of edges on the graph. For example, $c(a_1) = c(1,5), c(a_2) = c(4,5)$ and $c(a_3) = 0.4c(1,5) + 0.6c(4,5)$. Since deterministic policies are special cases of randomized ones, without loss of generality, we only consider randomized policies that produce reversible transition matrices. As such, each randomized policy induces a random walk on a graph with different weights $W(i,j)$ as shown in Figure 4.1b. Recall that in Section 3.3, the

weights $W(i,j)$ on graph specifies how the walker moves around the vertices of the graph.

Now we will formalize the connection between a random walk on a graph and a probabilistic policy. Let us consider an MDP with state space S, a set of actions for each states A and a reward function $\mathcal{R}(s,a)$. We model the reward $\mathcal{R}(s,a)$ as the difference between the benefit $b(s)$ and the cost of the action $c(a)$: $\mathcal{R}(s,a) = b(s) - c(a)$. The number of possible discrete policies is $|A|^{|S|}$. For typical MDP problems, this is a very large number. As such, we will parameterize the discrete policy space using a smaller subset of policies Φ. Since we only consider reversible policies, each discrete policy $i \in \Phi$ can be characterized by a symmetric weight matrix $\mathbf{W}_i$ of size $n \times n$ where n is the number of sensor nodes in the network. Specifically, we define a probabilistic weight matrix as a weighted combination of all the weight matrices: $\mathbf{W} = \sum_i \theta_i \mathbf{W}_i$ where $\theta_i \geq 0 \quad \forall i$ and $\sum_i \theta_i = 1$. Now, we define a probabilistic policy as follows. Let $p(s,i)$ be the probability that we select the policy $i \in \Phi$ while in state s. Then $p(s,i)$ can be computed using the following Proposition:

Proposition 2. *For each combined weight matrix with θ, the probability that we select policy i in state s is:*

$$p(s,i) = \frac{\theta_i H_i(s)}{H(s)} \quad \textit{for all } s \in S, i \in \Phi$$

where $H_i(s) = \sum_j \mathbf{W}_i(s,j)$, $H(s) = \sum_j \mathbf{W}(s,j)$.

Proof. We have $\mathbf{W} = \sum_i \theta_i \mathbf{W}_i$, the corresponding transition matrix:

$$\begin{aligned}
\mathbf{P}(s,s') &= \frac{\mathbf{W}(s,s')}{H(s)} = \frac{\sum_i \theta_i \mathbf{W}_i(s,s')}{H(s)}, \\
&= \sum_i \frac{\theta_i \mathbf{W}_i(s,s')}{H(s)} = \sum_i \frac{\mathbf{W}_i(s,s')}{H_i(s)} \frac{\theta_i H_i(s)}{H(s)}, \\
&= \sum_i \mathbf{P}_i(s,s') \frac{\theta_i H_i(s)}{H(s)}. \qquad (4.1)
\end{aligned}$$

Therefore, $p(s,i) = \frac{\theta_i H_i(s)}{H(s)}$ □

The Proposition 2 tells us for a given probabilistic policy parameterized by a specific set of θ_i's, the probability of picking a policy i while in state s is $\frac{\theta_i H_i(s)}{H(s)}$. Then, since given state s, an action a might appear in many policies i in Φ, the probability that an action a is selected while in state s can be computed as:

$$d(s,a) = \sum_{i(s)=a} p(s,i) \qquad (4.2)$$

4.3 Fast Markov Decision Processes

There are many algorithms to find out the optimal deterministic policies for an MDP problem. These include algorithms such as Value Iteration, Policy Iteration, etc. [6]. However, these policies are asymptotically optimal in the sense that the policies have to run a for long time under stationary environment. The underlying reason for this is the transition probability matrix associated with one of these policies has a small spectral gap (Section 3.2), which leads to the slow convergence

rate to the its stationary distribution π which directly affects the optimal reward. Specifically, the average reward for a given probabilistic policy is as follows:

$$\begin{aligned} \rho_1^{\boldsymbol{\pi}} &= \sum_s \boldsymbol{\pi}(s) \sum_a d(s,a)\mathcal{R}(s,a), \\ &= \sum_s \boldsymbol{\pi}(s) \sum_i p(s,i)\mathcal{R}(s,i) \end{aligned} \tag{4.3}$$

where $d(s,a)$ (computed in Eq. (4.2)) is the probability of taking action a in state s, $\mathcal{R}(s,a)$ is the immediate reward of taking action a in state s [36], $p(s,i)$ is the probability of picking policy i and $\mathcal{R}(s,i) = \mathcal{R}(s,i(s))$. As seen, the reward depends on the stationary distribution, the probabilistic policy and the immediate reward. However, a policy will obtain the reward faster if its transition probability matrix has faster convergence rate due to larger spectral gap or smaller SLEM. A faster policy is especially useful in a non-stationary environment in the sense that it attempts to get a large reward as quick as possible before the environment changes. In contrast, an optimal but slow policy will not perform as well since the optimal reward is never obtained as it quickly becomes a sub-optimal policy in a new environment. On the other hand, a faster policy might have lower average reward if the environment is stationary. Therefore, we propose an FMDP framework that allows us to quantify the trade-off between obtaining a high reward and the ability to adapt to time-varying environments.

Now, we note that the mixing rate or convergence rate is characterized by the second largest eigenvalue modulus (SLEM) $\mu(\mathbf{P})$ of a reversible, irreducible and aperiodic transition matrix $\mathbf{P}$ [8,35] or its spectral gap $(1-\mu(\mathbf{P}))$. For a fast policy

characterized by its transition probability matrix $\mathbf{P}$, $(1-\mu(\mathbf{P}))$ needs to be large. Therefore, we want to maximize $\rho_2 = 1-\mu(\mathbf{P})$ or minimize $\mu(\mathbf{P})$.

To take into accoung both the average reward and fast convergence rate of a policy, we want to find $\theta_1, \theta_2, \ldots, \theta_n$ that maximize the following objective function:

$$\begin{aligned} \rho &= \gamma\rho_1 + (1-\gamma)\rho_2, \\ &= \gamma \sum_s \boldsymbol{\pi}(s) \sum_i p(s,i)\mathcal{R}(s,i) + (1-\gamma)(1-\mu(\mathbf{P})), \end{aligned} \tag{4.4}$$

where the coefficient $0 < \gamma < 1$ specifies the trade-off between the average reward and the convergence rate.

The constraints are used to guarantee the solution is a valid transition probability matrix. These includes $\mathbf{P1} = \mathbf{1}$ which enforces a matrix to be a stochastic transition matrix, $\boldsymbol{\pi}^T\mathbf{P} = \boldsymbol{\pi}^T$ which enforces reversiblity of the matrix as stated in Definition 3. We note that a probabilistic MDP policy using a Random Walk on Weighted Graph (as shown in Section 3.3, Section 4.2) is enforced by letting $\boldsymbol{\theta} > 0, \quad \sum_{i=1}^{n} \theta_i = 1$. As a result, the problem can be formulated as:

$$\begin{aligned} \text{maximize} \quad & \gamma \sum_s \boldsymbol{\pi}(s) \sum_i p(s,i)\mathcal{R}(s,i) + (1-\gamma)(1-\mu(\mathbf{P})) \\ \text{s.t.} \quad & \boldsymbol{\theta} \geq 0 \\ & \sum_{i=1}^{n} \theta_i = 1 \end{aligned} \tag{4.5}$$

We note that once θ_i's are found, they can be used to compute $p(s,i)$ in Proposition 2. Next, $p(s,i)$ is used to compute the probability of taking action a in state s via

Eq. (4.2). There might be other constraints depending on each application such as $\boldsymbol{\pi} = \boldsymbol{\pi}^*$ as discussed in the Section 5.

Chapter 5: Solution Approach via Fast Markov Decision Process Framework

We describe an algorithm for finding the optimal FMDP policy, i.e., θ_i's. The proposed algorithm is based on the projected sub-gradient method used extensively in convex optimization. The main idea is to search in the direction of the sub-gradient of the objective function. Sometimes the search direction results in a point outside the feasible set. In that case, the point is projected to closest point in the feasible set, then the process repeats. In our problem, the overall objective consists of two components: the average reward and the SLEM. We now show how to compute the gradient for the average reward and the sub-gradient (directional gradient) for SLEM since they are the essential components of the proposed algorithm.

5.1 Gradient for Average Reward

Since $\boldsymbol{\pi}(s) = \frac{H(s)}{\sum_{k \in S} H(k)}, p(s,i) = \frac{\theta_i H_i(s)}{H(s)}, \quad {}_k H(k) = 1$ and $\quad {}_i \theta_i = 1$, the average reward ρ_1 is simplified as:

$$\begin{aligned}
\rho_1 &= \sum_s \boldsymbol{\pi}(s) \sum_i p(s,i) \mathcal{R}(s,i), \\
&= \sum_s \frac{H(s)}{\sum_{k \in S} H(k)} \sum_i \frac{\theta_i H_i(s)}{H(s)} \mathcal{R}(s,i), \\
&= \sum_{i=1}^{n} \theta_i \sum_s H_i(s) \mathcal{R}(s,i), \\
&= \theta_n \sum_s H_n(s) \mathcal{R}(s,n) + \sum_{i=1}^{n-1} \theta_i \sum_s H_i(s) \mathcal{R}(s,i), \\
&= (1 - \sum_{i=1}^{n-1} \theta_i) \sum_s H_n(s) \mathcal{R}(s,n) + \\
&\quad + \sum_{i=1}^{n-1} \theta_i \sum_s H_i(s) \mathcal{R}(s,i), \\
&= \sum_s H_n(s) \mathcal{R}(s,n) \sum_{i=1}^{n-1} \theta_i (\sum_s (H_i(s) \mathcal{R}(s,i) \\
&\qquad\qquad - H_n(s) \mathcal{R}(s,n))). \qquad (5.1)
\end{aligned}$$

Then, $\frac{d\rho_1}{d\theta_i} = \sum_s (H_i(s)\mathcal{R}(s,i) - H_n(s)\mathcal{R}(s,n)), \forall i = 1, 2, \ldots, n-1.$

5.2 Sub-gradient (Directional Gradient) for SLEM $\mu(\mathbf{P})$

Consider the second largest eigenvalue modulus (SLEM) $\mu(\mathbf{P})$ of a reversible transition matrix $\mathbf{P}$ with stationary distribution $\boldsymbol{\pi}$. $\mu(\mathbf{P})$ is also the largest eigenvalue

of the symmetric matrix $A = \mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T$ [31]. Therefore, it can be calculated as

$$\mu(\mathbf{P}) = ||\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T||_2, \tag{5.2}$$

where $\mathbf{D}$ is the diagonal matrix with diagonal elements taken from the stationary distribution $\boldsymbol{\pi}$ [31].

For a symmetric matrix $A(\mathbf{x}) = A_0 + x_1A_1 + \ldots + x_mA_m$ where $\mathbf{x} = (x_1, \ldots, x_m)$, A_i is a symmetric matrix for all $i = 1, 2 \ldots, m$. Let $f(\mathbf{x})$ be the largest eigenvalue of $A(\mathbf{x})$. Let y be the eigenvector corresponding to the largest eigenvalue. Then $y^TA_1y, \ldots, y^TA_my$ are the sub-gradient [38, 39] or mentioned as directional gradients [37, 39] (if the largest eigenvalue is distinct) for the largest eigenvalue of A at point $\mathbf{x}$ in the direction $A_1, \ldots, A_m$, respectively. More details about this could be found in Appendix A. In our problem, $A = \mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T$. From Section 3.3 we can see that $\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ (normalized Laplacian

of a graph [40]). Therefore,

$$\begin{aligned}
A &= \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T \\
&= \mathbf{D}^{-1/2}(\sum_i \theta_i \mathbf{W}_i)\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T \\
&= \sum_{i=0}^{n} \theta_i \mathbf{D}^{-1/2}\mathbf{W}_i\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T \\
&= \theta_n \mathbf{D}^{-1/2}\mathbf{W}_n\mathbf{D}^{-1/2} + \sum_{i=0}^{n-1} \theta_i \mathbf{D}^{-1/2}\mathbf{W}_i\mathbf{D}^{-1/2} \\
&\quad -\sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T \\
&= (1 - \sum_{i=1}^{n-1} \theta_i)\mathbf{D}^{-1/2}\mathbf{W}_n\mathbf{D}^{-1/2} + \\
&\qquad \sum_{i=0}^{n-1} \theta_i \mathbf{D}^{-1/2}\mathbf{W}_i\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T \\
&= \sum_{i=0}^{n-1} \theta_i \mathbf{D}^{-1/2}(\mathbf{W}_i - \mathbf{W}_n)\mathbf{D}^{-1/2} + \\
&\qquad \mathbf{D}^{-1/2}\mathbf{W}_n\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T
\end{aligned}$$

At each step, the stationary distribution $\boldsymbol{\pi}$ is considered fixed. Let

$$A_0 = \mathbf{D}^{-1/2}\mathbf{W}_n\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T, \quad A_i = \mathbf{D}^{-1/2}(\mathbf{W}_i - \mathbf{W}_n)\mathbf{D}^{-1/2},$$

which are symmetric matrices, then we have the sub-gradient $g_i = y^T A_i y$ of $\boldsymbol{\mu}(P)$ (directional gradient in the direction of A_i for all i from 1 to $n-1$). Note that unlike gradient of ρ_1, to compute g_i we need to perform eigen-decomposition of A.

Since $\sum_i \theta_i = 1$, the sub-gradient (directional gradient) is taken only on the

parameter vector:

$$\boldsymbol{\theta} = (\theta_1, \quad \theta_1, \quad \ldots, \quad \theta_{n-1}).$$

Let $\mathbf{g} = (y^T A_1 y, y^T A_2 y, \ldots, y^T A_{n-1} y)$, then we have the sub-gradient (directional gradient) for the combined objective function ρ with respect to $\boldsymbol{\theta}$:

$$\frac{d\rho}{d\boldsymbol{\theta}} = \gamma \frac{d\rho_1}{d\boldsymbol{\theta}} + (1-\gamma)\frac{d\rho_2}{d\boldsymbol{\theta}} = \gamma \frac{d\rho_1}{d\boldsymbol{\theta}} - (1-\gamma)\mathbf{g} \tag{5.3}$$

5.3 Algorithm

Consider the combined objective: $\rho = \gamma\rho_1 + (1-\gamma)\rho_2$. The method combines gradient of the average reward and the sub-gradient (directional gradient) of the SLEM according to the coefficient γ. Denote $\boldsymbol{\theta}_i$ as the parameter vector at step i. The algorithm is shown below (Algorithm 1). Essentially, the algorithm searches

Algorithm 1 Sub-gradient Method for Fast MDP

Require: $\boldsymbol{\theta}_0, \boldsymbol{\theta}_{best} = \boldsymbol{\theta}_0, \alpha, k = 0$

1: **repeat**
2: $\quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \frac{d\rho}{d\boldsymbol{\theta}}$
3: $\quad k = k + 1$
4: $\quad$ Project $\boldsymbol{\theta}_{k+1}$ back to the feasible set.
5: $\quad$ Keep track of the best policy: Update $\boldsymbol{\theta}_{best} = \boldsymbol{\theta}_{k+1}$ if $\rho(\boldsymbol{\theta}_{k+1}) > \rho(\boldsymbol{\theta}_{best})$.
6: **until** *convergence* (The change in the objective function is smaller than a given ϵ)

in sub-gradient direction with step size α. If the point is outside the feasible set, it is projected back to nearest point in the feasible set. Specifically, the feasible set includes the constraint on θ_i: $\theta_i \geq 0 \quad \forall i$, the constraint on the sum of all θ_i's:

$_i\theta_i = 1$ and constraint on the stationary distribution $\boldsymbol{\pi}$: $\boldsymbol{\pi} = \boldsymbol{\pi}^*$ where $\boldsymbol{\pi}^*$ is the designed stationary distribution.

The projection could be done by solving the following convex optimization problem:

$$\begin{aligned} \textbf{minimize}_{\boldsymbol{\theta}'} \quad & (\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_1) \\ s.t. \quad & \boldsymbol{\theta}' \geq 0, \\ & \|\boldsymbol{\theta}'\|_1 \leq 1, \\ & \boldsymbol{\theta}'\boldsymbol{\Delta H} + H_n = \boldsymbol{\pi}^*, \end{aligned} \tag{5.4}$$

where the matrix $\boldsymbol{\Delta} H$ is defined as $\boldsymbol{\Delta H}(i,s) = H_i(s) - H_n(s)$.

The condition $\boldsymbol{\theta}'\boldsymbol{\Delta H} + H_n = \boldsymbol{\pi}^*$ comes from the constraint on fixed stationary distribution $\boldsymbol{\pi} = \boldsymbol{\pi}^*$ as shown below:

$$\begin{aligned} \boldsymbol{\pi}^*(s) &= \boldsymbol{\pi}(s) = \frac{H(s)}{\sum_{k\in S} H(k)} = H(s) = \sum_{i=1}^{n} \theta_i' H_i(s), \\ &= \sum_{i=1}^{n-1} \theta_i' H_i(s) + \theta_n' H_n(s), \\ &= \sum_{i=1}^{n-1} \theta_i' H_i(s) + (1 - \sum_{i=1}^{n} \theta_i') H_n(s), \\ &= \sum_{i=1}^{n-1} \theta_i' (H_i(s) - H_n(s)) + H_n(s), \quad \forall s \in S \end{aligned} \tag{5.5}$$

or in matrix form: $\boldsymbol{\theta}'\boldsymbol{\Delta H} + H_n = \boldsymbol{\pi}^*$ where the matrix ΔH is defined as $\boldsymbol{\Delta H}(i,s) = H_i(s) - H_n(s)$.

In other words, we find the closest point to θ that satisfies all the constraints in the problem formulation. Since we use sub-gradient method, we have to keep track of the best policy with the best combined objective function. Please refer to [41] [42] for more information on sub-gradient methods.

5.4 Convexity

In many cases, the optimal stationary distribution $\boldsymbol{\pi}^*$ of the ME is given due to the design requirements of the problem. For example, when the rate of data generated at each node is known a priori. Then, to obtain high average reward, the ME should visit the nodes with higher generating rates more often than others. In other words, the stationary distribution $\boldsymbol{\pi}^*$ of the ME should be proportional to data rate at each node. In this case, our optimization problem is concave.

Proposition 3. *For a fixed $\boldsymbol{\pi}^*$, the objective function ρ is a concave function in θ_i's*

Proof. From (5.1), ρ_1 is a linear function of θ_i's. Therefore, it is a concave function of θ_i's. As shown in [31], $\mu(\mathbf{P})$ is convex in term of the transition matrix $\mathbf{P}$. Since $\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ as shown above, $\mu(\mathbf{P})$ is also convex in term of $\mathbf{W}$, and therefore convex in term of $\boldsymbol{\theta}$ without constraint (5.5). Fortunately, constraint (5.5) is convex. Therefore, with constraint (5.5), $\mu(\mathbf{P})$ is still convex in term of $\boldsymbol{\theta}$ or $(1-\mu(\mathbf{P}))$ is concave in term of $\boldsymbol{\theta}$. As a result, the combined objective function ρ is concave in term of $\boldsymbol{\theta}$. □

Since ρ is a concave function, we propose to find a global optimal solution using a sub-gradient method discussed in Section 5.3. Note that, without constraint that $\boldsymbol{\pi}^*$ is fixed, the algorithm still converges and returns local optimal point.

5.5 Relaxation

To obtain faster policy, we enlarge the feasible set being searched for optimal solution. As a result, we might find a faster policy whose stationary distribution is close to the target stationary distribution. Specifically, instead of fixing $\boldsymbol{\pi}$, we allow $\boldsymbol{\pi}$ to be an optimization variable, but it must be close to the target stationary distribution $\boldsymbol{\pi}^*$, i.e., $\|\boldsymbol{\pi} - \boldsymbol{\pi}^*\| \leq \varepsilon$ for some constant ε. However, in doing so, we make $\boldsymbol{\pi}$ to be a variable in addition to $\mathbf{P}$. As a result, the SLEM $\mu(\mathbf{P}) = ||\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T||_2$ is no longer convex in $\mathbf{P}$ and $\boldsymbol{\pi}$. Now, to make the problem convex, we modified the SLEM in the objective function from $\mu(\mathbf{P}) = ||\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T||_2$ to $\tilde{\mu}(\mathbf{P}) = ||\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} - \sqrt{\boldsymbol{\pi}^*}\sqrt{\boldsymbol{\pi}^*}^T||_2$. The modified SLEM can be shown to be convex in $\mathbf{P}$ since $\boldsymbol{\pi}^*$ is given. The combined objective function of the reward and SLEM is now different. However, since $\boldsymbol{\pi}^*$ and $\boldsymbol{\pi}$ are guaranteed to be no difference than ε. We can bound the approximate objective and the true objective as a function of ε. Specifically, in our previous work [32], we provided a bound on the constraint on $\|\boldsymbol{\pi} - \boldsymbol{\pi}^*\|$ in Proposition 2 of the thesis, and showed that, such bound implies an upper bound on the difference between the true SLEM and the approximated SLEM in Proposition 4.

After relaxation, the projection (5.4) becomes

$$\begin{aligned} \textbf{minimize}_{\boldsymbol{\theta}'} \quad & (\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_1) \\ s.t. \quad & \boldsymbol{\theta}' \geq 0, \\ & \|\boldsymbol{\theta}'\|_1 \leq 1, \\ & \|\boldsymbol{\theta}'\boldsymbol{\Delta}\boldsymbol{H} + H_n - \boldsymbol{\pi}^*\|_\infty \leq \varepsilon. \end{aligned}$$

In this thesis, we use ∞-norm for the difference between $\boldsymbol{\pi}$ and $\boldsymbol{\pi}^*$.

Chapter 6: Simulations and Results

6.1 Simulation Scenarios

In this section, we consider two graphs of WSN, one with 9 nodes on a 3×3 grid and one with 10 nodes on a triangle-topology graph as shown in Figures 6.1 and 6.2. The corresponding benefits and costs are also shown in Figures 6.1 and 6.2. Given the benefits, i.e. the data generating rate, we can compute the designed stationary distribution which is proportional to the benefits. Note that this is only an example. In general, one can use any benefit-cost structure to model the problem at hand. We show the results for various values of γ. $\gamma = 1$ implies that we only care about the average reward while $\gamma = 0$ implies that we only care about the convergence rate. For $0 < \gamma < 1$, we care both about the reward and the convergence rate with the weight γ and $1 - \gamma$, respectively.

For the first scenario, 9 nodes are on a 3×3 grid with benefits and costs shown in the Figure 6.1. Note that if the ME stays at the same node after it collects data, there is a high chance that there is little data to collect, and therefore, wasting time to collect fresh data from other nodes. To discourage this behavior, the cost of staying at one node is set to 1 (not shown) which is larger than other costs. The desired stationary distribution $\boldsymbol{\pi}^* = [0.06 \quad 0.07 \quad 0.1 \quad 0.02 \quad 0.03 \quad 0.09 \quad 0.1 \quad 0.11 \quad 0.42]$, which is proportional to the nodes' benefits. For example, the rate of data col-

lected at node 1 is 7 times less than that of node 9. The initial distribution for the ME is generated randomly. Here, the algorithm step size is set at $\alpha = 0.0001$. The SLEM, the average reward and the combined objective function ρ are plotted on Figures 6.3, 6.4, 6.5 as a function of algorithm steps.

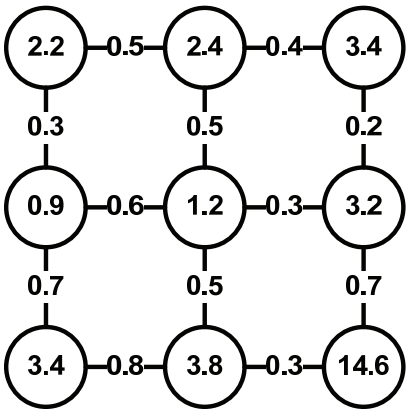


Figure 6.1: The graph for scenario 1 with benefits and costs shown in the nodes and along the edges, respectively.

For the second scenario, 10 nodes are on triangle-topology network as seen in the Figure 6.2. The benefit at each node and the cost of moving on edges are also shown in the Figure 6.2. The cost of staying at one node is 0.5 (not shown) which is also the largest cost of the graph. The desired stationary distribution $\boldsymbol{\pi}^* = [0.16 \quad 0.1 \quad 0.02 \quad 0.11 \quad 0.04 \quad 0.03 \quad 0.03 \quad 0.04 \quad 0.14 \quad 0.33]$, which is proportional to the nodes' benefits. The initial distribution for the ME is also generated randomly. Here, the algorithm step size α is also 0.0001. The SLEM, the average reward and the combined objective function ρ versus the algorithm steps are plotted on Figures 6.6, 6.7, and 6.8.

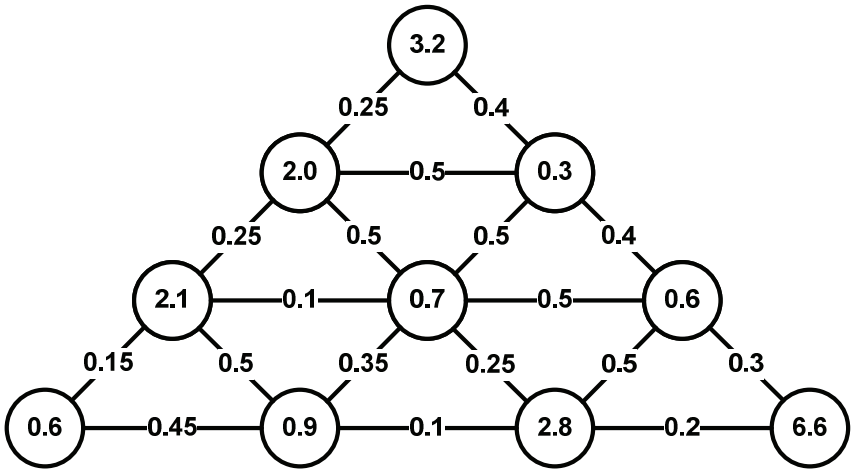


Figure 6.2: The graph for scenario 2 with benefits and costs shown in the nodes and along the edges, respectively.

6.2 Results and Comparisons

From Figures 6.3 and 6.4 for scenario 1 as well as Figures 6.6 and 6.7 for scenario 2, we can see that the policy with $\gamma = 1$ provides the largest reward but also with a large SLEM, i.e., slow convergence (average reward ≈ 7.25 and SLEM ≈ 1 for scenario 1, average reward ≈ 3.36 and SLEM ≈ 1 for scenario 2). For the policy with $\gamma = 0$, we get a policy with fastest converge but lower reward (average reward ≈ 7.1 and SLEM ≈ 0.81 for scenario 1, the average reward ≈ 3.29 and SLEM ≈ 0.91 for scenario 2). For the policy with $\gamma = 0.5$, we get a policy that converges fast and at the same time collects a large reward (average reward ≈ 7.15 and SLEM ≈ 0.82 for scenario 1, average reward ≈ 3.33 and SLEM ≈ 0.82 for scenario 2). As seen in the Figures 6.5 and 6.8, the combined objective function ρ of our method outperforms others in both scenarios.

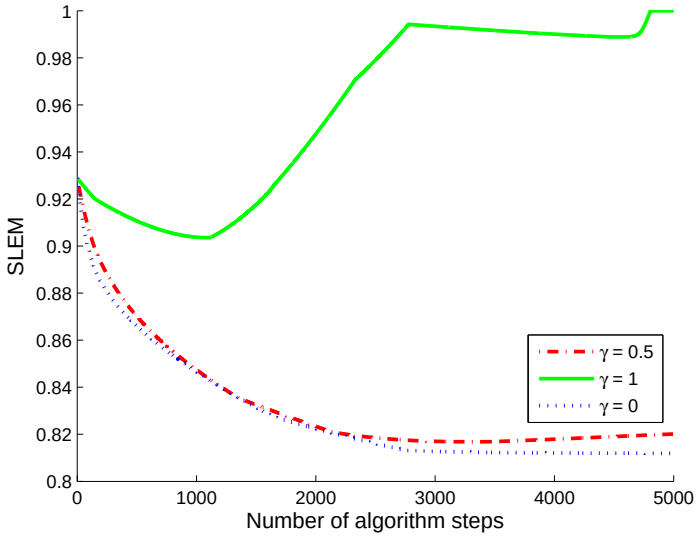


Figure 6.3: The SLEM for scenario 1 as a function of algorithm steps

To evaluate the impacts of the relaxation on the projection constraints, we run the algorithm formulated in Section 5.5 with $\varepsilon = 0.2$ for all the scenarios. We obtain better policies with larger average reward and smaller SLEM (average reward ≈ 9.5 and SLEM ≈ 0.6 for scenario 1, average reward ≈ 4.5 and SLEM ≈ 0.6 for scenario 2). This is because we expand the space of feasible solutions by allowing the set of all feasible distributions $\varepsilon-$close to the target distribution $\boldsymbol{\pi}^*$.

To show the advantage of the proposed FMDP approach in changing network conditions or limited collection time, we perform the following simulations. Based on the optimal policies for $\gamma = 1$ and $\gamma = 0.5$, the policies with projection relaxation $\varepsilon = 0.2$, the probabilistic greedy and random policies, we compute their corresponding transition matrices and the average rewards over time in changing environments. We also compare these policies with two other policies including

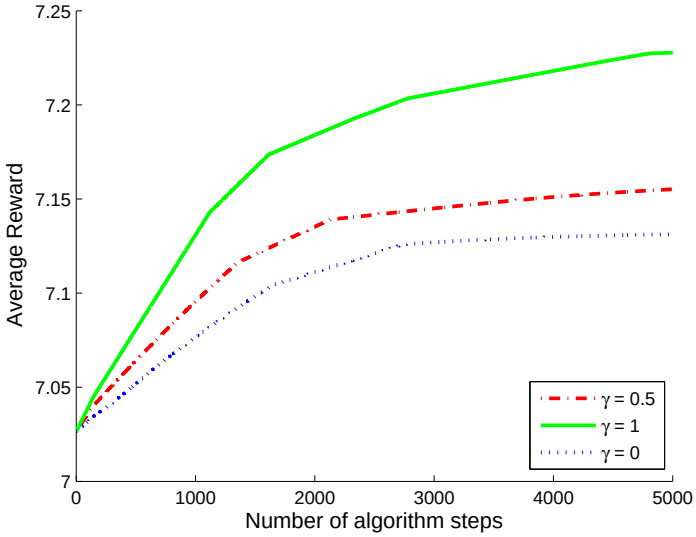


Figure 6.4: The average reward for scenario 1 as a function of algorithm steps

probabilistic greedy policy and random policy. For the probabilistic greedy policy, the probability of taking action a in state s is calculated as $d(s,a) = \frac{\mathcal{R}(s,a)}{{}_b\,\mathcal{R}(s,b)}$. This implies that actions with larger reward for a given state s have larger probability of being taken. For random policy, the probability of taking action a in state s is randomly chosen.

Assume that different sensors have different rates of acquiring new data. This is due to the frequency of "new" data being generated at different nodes. Thus, a policy in which the ME visits a node with higher data acquiring rate would be prefer in order to avoid buffer overflow. Consequently, we model the benefit of a node being proportional to its generating rate. Furthermore, we assume these rates at sensors are non-stationary, rather changes abruptly. This non-stationary setting can model the scenarios such as day time and night time data collection

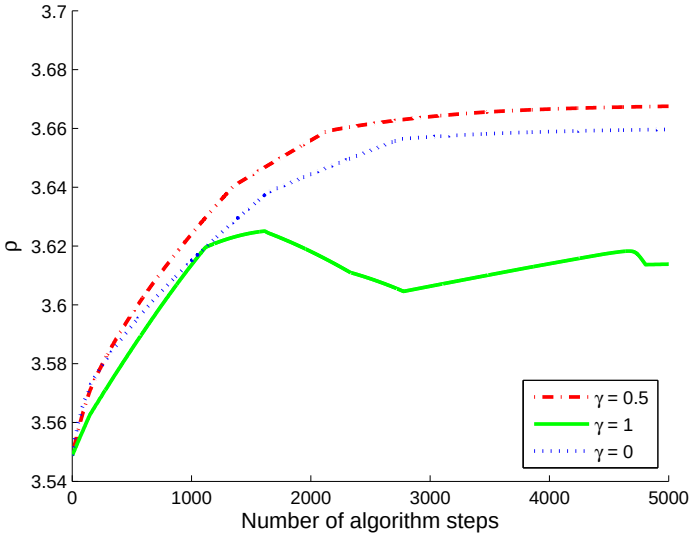


Figure 6.5: The combined objective function for scenario 1 as a function of algorithm steps

where the rate of "new" data generated can be drastically different. Specifically, the benefits of the data collected, which is proportional to the generating rates, changes twice at epochs 50 and 90 for scenario 1 and at epochs 40 and 90 for scenario 2. Each time the reward changes, we recompute the average reward.

Figures 6.9 and 6.11 show the performance of the considered policies in changing environment. The area under the curve represents the total expected reward. Therefore, the policy with larger area is the better one. Clearly, with the FMDP based policy ($\gamma = 0.5, \varepsilon = 0$ (no relaxation)), the ME collects more rewards than the classical optimal MDP policy ($\gamma = 1, \varepsilon = 0$). This is due to the fact that the environment changes before the classical optimal MDP policy accumulate the optimal reward. In addition, the policies with relaxed projection ($\gamma = 0.5, \varepsilon = 0.2$)

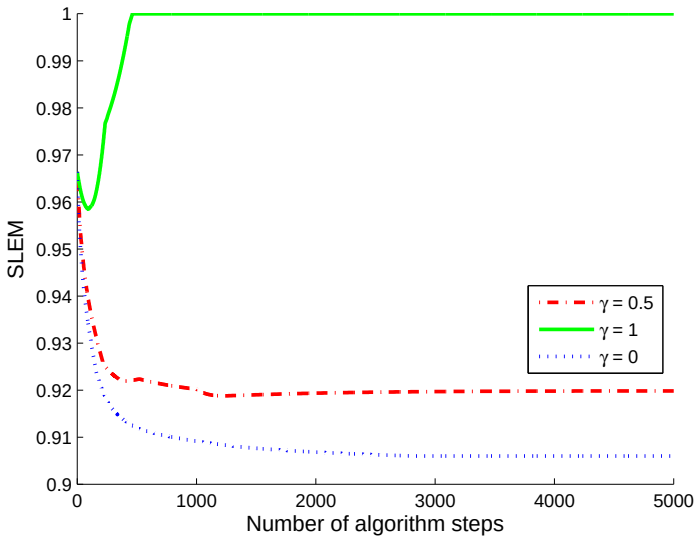


Figure 6.6: The SLEM for scenario 2 as a function of algorithm steps

perform better than the one without relaxation $\gamma = 0.5, \varepsilon = 0$). The reason is that the former one gets better average reward and SLEM as shown above. Besides, these three optimal policies with different γ and ε provide better reward than the greedy and random policies.

For completeness, we also show the rates of convergence for the two policies to the new stationary distributions when the reward structure change for the above scenarios. Figures 6.10 and 6.12 shows the total variation distances decreases for the policies as the ME move around the nodes. As seen, the FMDP policy ($\gamma = 0.5, \varepsilon = 0$) has a faster convergence rate than the one with reward direction only ($\gamma = 1, \varepsilon = 0$). This explains the reason for the FMDP policy to obtain faster reward. However, we should note that in a stationary environment, the policy with $\gamma = 1$ will eventually outperform the other. Since we use projection relaxation

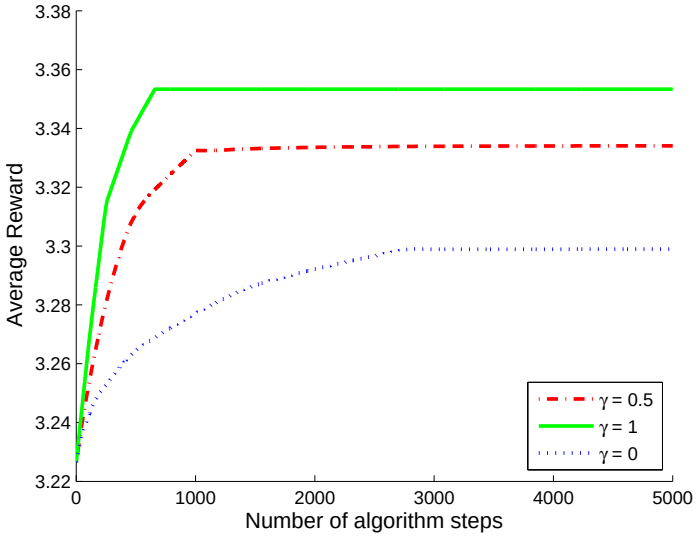


Figure 6.7: The average reward for scenario 2 as a function of algorithm steps

for the policy with ($\gamma = 0.5, \varepsilon = 0.2$), it might converge to different stationary distribution. For random and greedy policies, since there is no constraints on the fixed stationary distribution, they might converge to different distributions as well.

In many real-world scenarios where statistical characterizations of how long the environment remain stationary are unknown, one often uses the discounted reward formulation for the infinite horizon problems. In this case, the optimal policy aims to emphasize the rewards in the near future (next few time steps) than those of the far future (later time steps) via the use of exponential weighting/discount on the rewards. Specifically, in the discount infinite horizon problem, the goal is to minimize $V^{\Pi}(s) = E(\sum_{n=1}^{\infty} \alpha^n \mathcal{R}(s_n, a_n))$. A greedy policy is an extreme example of maximizing the immediate reward in the next time step, and ignoring rewards in all later time steps. Of course, it is well-known that a greedy algorithm

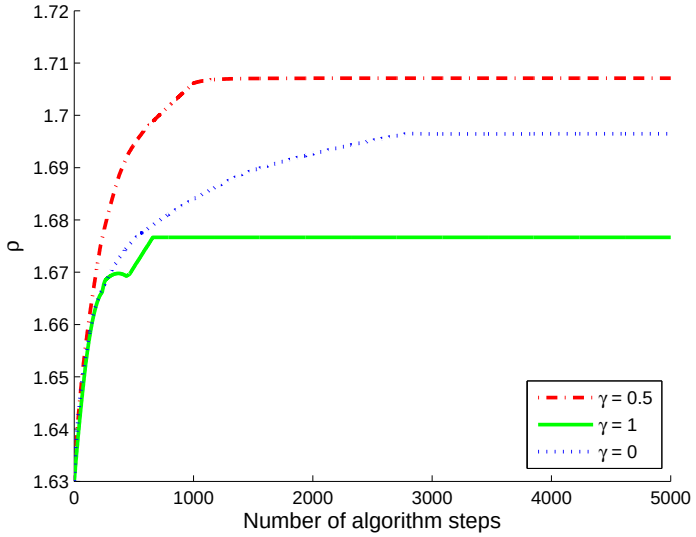


Figure 6.8: The combined objective function for scenario 2 as a function of algorithm steps

is often not an optimal algorithm when the environment is stationary since the optimization is performed myopically. Setting the discount factor α to a certain value attempts to provide a gradual de-emphasization of future rewards. But this is an adhoc way to cope with the uncertained changes in the environments. In other words, if the environment changes quickly, α is set to a small value while if the environment changes slowly, α is set to a large value, but there is no precise analysis on the values of α since the precise statistical characterizations of how long the environment remain stationary are assumed unknown.

Let us now compare the proposed FMDP approach and the discount infinite horizon MDP formulation. First, the FMDP approach optimizes for the average value, thus will outperform the infinite horizon approach if the environment re-

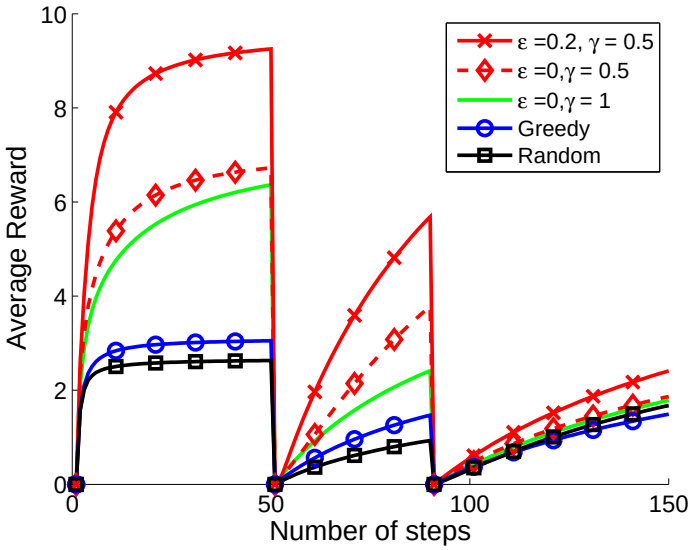


Figure 6.9: The average reward collected in non-stationary distribution for scenario 1

mains stationary. This is due to the fact there might be multiple optimal policies for the given infinite horizon problem, the FMDP approach will use the one with fast convergence time and result in optimal average value. On the other hand, the discount infinite horizon MDP policy will maximize the discount value not the average value, thus will be suboptimal when the environment is stationary. Second, when the environments are non-stationary, for example, one cannot statistically characterize when the sudden change will happen, then the best one can do is try to produce a policy with fast convergence rate and a good average reward. The trade-off between average reward and the convergence rate is controlled by γ in the objective function . This is the principle employed by the FMDP approach. On the other hand, using the discount infinite horizon formulation, one have to set

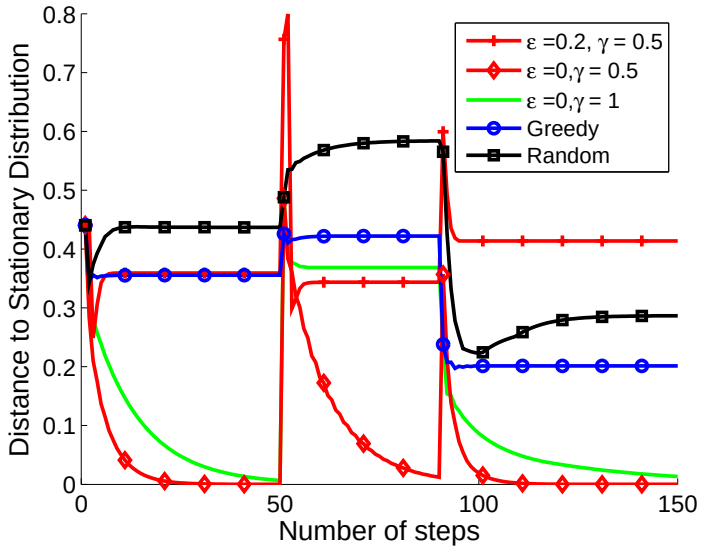


Figure 6.10: The total variation distance to the stationary distribution in non-stationary environment for scenario 1

the α value heuristically. If the value of α is set incorrectly, the average reward will be small. To illustrate this, we evaluate performances of the the FMDP and the discount infinite horizon policies for the network in Scenario 1. Figure 6.13 below shows the average reward of two policies over time, one from FMDP framework and one from Value Iteration for classic MDP with different discounted factors. Note that the stationary distribution of the classic policy is not necessarily to be fixed. Therefore, to have fair comparison, we compare its performance versus the relaxed FMDP policy where the stationary distribution also can be changed in ε boundary. We can see that for the first few time steps, the classic MDP policies perform better due to their greedy design. However, as we discussed above, later on our policy outperforms the discount infinite horizon policies.

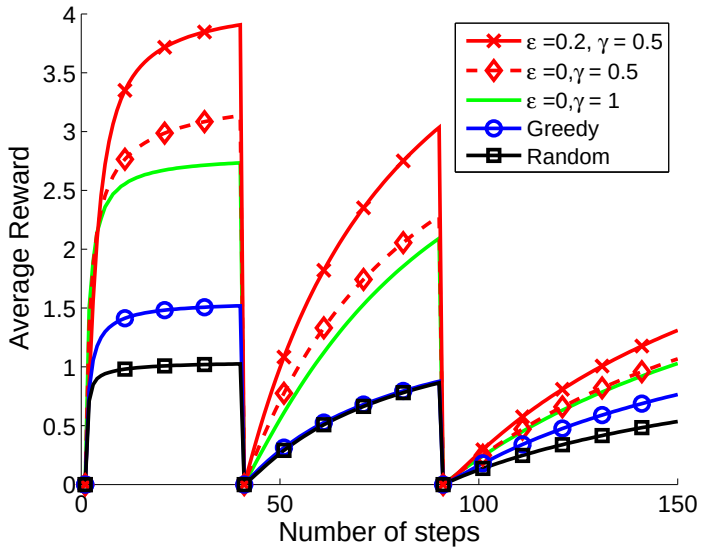


Figure 6.11: The average reward collected in non-stationary distribution for scenario 2

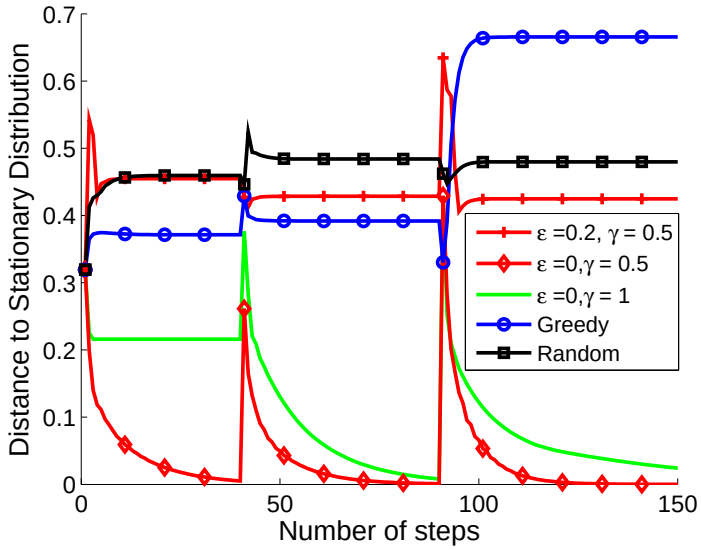


Figure 6.12: The total variation distance to the stationary distribution in non-stationary environment for scenario 2

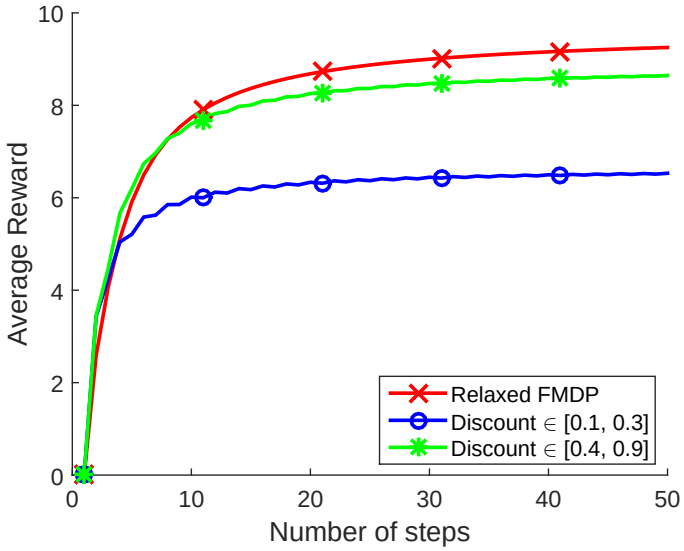


Figure 6.13: Relaxed FMDP policy's performance versus classic MDP with different discount factors for scenario 1

Chapter 7: Conclusion

We introduced the FMDP framework for designing trajectories/policies for data collection in sensor networks in fast-changing environments. The proposed FMDP framework and its algorithmic tools allow the engineers to systematically design trajectories/policies that can be optimized for a wide range of objectives. Importantly, the proposed FMDP framework enables one to design a policy that strikes a balance between maximizing a given objective and doing so in quantifiable fast time. The proposed framework can be extended to model explicitly many other real world constraints including the limitations on latency and buffer size.

APPENDICES

Appendix A: Sub-gradient/Directional Derivatives of Maximum Eigenvalue of Symmetric Matrices

Definition 4. *Directional Derivatives [37, 42]. The directional derivative of a function f at the point $x \in \mathbf{R}^n$ in the direction of $\nu \in \mathbf{R}^n$ is defined as*

$$f'(x;\nu) := \lim_{t \to 0^+} \frac{f(x+t\nu) - f(x)}{t}.$$

Note that $f'(x;\nu)$ exists if f is convex and finite at point x [42].

Proposition 4. *Directional derivatives of maximum eigenvalue of symmetric matrices [37]. Let A and E be $n \times n$ symmetric matrices. Let λ_m be a distinct eigenvalue of A. Then we have the first-order expansion of λ_m:*

$$\lambda_m(A+tE) = \lambda_m(A) + ty^T Ey + O(t^2) \qquad \textit{for } t > 0,$$

where y is the normalized eigenvector associated with λ_m. As a result, the first order directional derivative of λ_m in the direction of E is $y^T Ey$

Proof. Since A, E are $n \times n$ symmetric matrices and the multiplicity of λ_m is 1, by applying Theorem 1.5 in [37], we have:

$$\lambda_m(A+tE) = \lambda_m(A) + t\lambda_1(y^T Ey) + O(t^2) \qquad \text{for } t > 0,$$

where $\lambda_1(y^T E y)$ is the largest eigenvalue of $y^T E y$. However, we can easily see that $y^T E y$ is a scalar. Thus, $\lambda_1(y^T E y) = y^T E y$. Therefore, we have

$$\lambda_m(A + tE) = \lambda_m(A) + t y^T E y + O(t^2) \qquad \text{for } t > 0.$$

In other words, the first order directional derivative of the largest eigenvalue of A is

$$\lambda'_m(A; E) = \lim_{t \to 0^+} \frac{\lambda_m(A + tE) - \lambda_m(A)}{t} = y^T E y$$

□

Note that this is a simplified version of Theorem 1.5 in [37] in the case of distinct eigenvalues.

Definition 5. *Sub-gradient [42]. A vector $g \in \mathbf{R}^m$ is a subgradient of $f : \mathbf{R}^m \to \mathbf{R}$ at $x \in \boldsymbol{dom} f$ if for all $z \in \boldsymbol{dom} f$,*

$$f(z) \geq f(x) + g^T(z - x),$$

where $\boldsymbol{dom} f$ is the domain where f is defined.

Proposition 5. *Sub-gradient of maximum eigenvalue of a symmetric matrix ([42], Sections 3.4 and 3.5). Let $f(x) = \lambda_{max}(A(x))$ where $x = (x_1, x_2, \ldots, x_m) \in \mathbf{R}^m$, A is an $n \times n$ symmetric matrix and decomposed as*

$$A(x) = A_0 + x_1 A_1 + \ldots + x_m A_m,$$

where A_i's are $n \times n$ symmetric matrices for all $i = 0, \ldots, m$. The largest eigenvalue $f(x)$ could be represented as

$$f(x) = \lambda_{max}(A(x)) = \sup_{\|y\|_2=1} y^T A(x) y,$$

where $y \in \mathbf{R}^n$. Then, the sub-gradient of $f(x)$ is given by

$$g = (y^T A_1 y, y^T A_2 y, \ldots, y^T A_m y),$$

where y is the normalized eigenvector of $A(x)$.

Proof. At a point x, $f(x) = \lambda_{max}(A(x)) = \sup_{\|y\|_2=1} y^T A(x) y = u^T A(x) u$ where u is the normalized eigenvector associated with the largest eigenvalue of $A(x)$.

Let $f_u(x) = u^T A(x) u = u^T A_0 u + x_1 u^T A_1 u + \ldots + x_m u^T A_m u$. Easily, we can see that $f_u(x)$ is a convex function and has g as gradient where

$$g = (u^T A_1 u, u^T A_2 u, \ldots, u^T A_m u).$$

Therefore, $f_u(z) \geq f_u(x) + g^T(z - x)$ for any point z (see [38], Section 3.1.3). As a result,

$$f(z) = \sup_{\|y\|_2=1} y^T A(z) y \geq u^T A(z) u = f_u(z) \geq f_u(x) + g^T(z - x) = f(x) + g^T(z - x)$$

In other words, g is a sub-gradient of $f(x)$. □

Bibliography

[1] M. Di Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM Trans. Sen. Netw.*, vol. 8, no. 1, pp. 7:1–7:31, Aug. 2011.

[2] J. Rao and S. Biswas, "Joint routing and navigation protocols for data harvesting in sensor networks," in *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, Sept 2008, pp. 143–152.

[3] S. R. Shah, Rahul C. and, S. Jain, and W. Brunette, "Data mules: modeling and analysis of a three-tier architecture for sparse sensor networks." *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 215–233, 2003.

[4] Y.-C. Tseng, Y.-C. Wang, K.-Y. Cheng, and Y.-Y. Hsieh, "imouse: An integrated mobile surveillance and wireless sensor system," *Computer*, vol. 40, no. 6, pp. 60–66, June 2007.

[5] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, ser. RTSS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 296–305.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[7] D. Aldous and J. A. Fill, *Reversible Markov Chains and Random Walks on Graphs*. In preparation. [Online]. Available: http://www.stat.berkeley.edu/~aldous/RWG/book.html

[8] D. A.Levin, Y. Peres, and E. L.Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

[9] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, March 2005, pp. 2302–2312 vol. 4.

[10] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "A hybrid adaptive protocol for reliable data delivery in wsns with multiple mobile sinks," *The Computer Journal*, vol. 54, no. 2, pp. 213–229, 2011.

[11] S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy, "Exploiting mobility for energy efficient data collection in wireless sensor networks," *Mob. Netw. Appl.*, vol. 11, no. 3, pp. 327–339, Jun. 2006.

[12] H. Jun, M. H. Ammar, and E. W. Zegura, "Power management in delay tolerant networks: A framework and knowledge-based mechanisms," 2005.

[13] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '04. New York, NY, USA: ACM, 2004, pp. 187–198.

[14] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," *SIGARCH Comput. Archit. News*, vol. 30, no. 5, pp. 96–107, Oct. 2002.

[15] Z. Haas and T. Small, "A new networking model for biological applications of ad hoc sensor networks," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 1, pp. 27–40, Feb 2006.

[16] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using predictable observer mobility for power efficient design of sensor networks," in *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, ser. IPSN'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 129–145.

[17] S. Poduri and G. S. Sukhatme, "Achieving connectivity through coalescence in mobile robot networks," in *Proceedings of the 1st International Conference on Robot Communication and Coordination*, ser. RoboComm '07. Piscataway, NJ, USA: IEEE Press, 2007, pp. 4:1–4:6.

[18] G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 12, pp. 1430–1443, Dec 2008.

[19] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C.-G. Lee, "Partitioning based mobile element scheduling in wireless sensor networks," in *Sensor and Ad*

Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on, Sept 2005, pp. 386–395.

[20] M. Ma and Y. Yang, "Sencar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 10, pp. 1476–1488, Oct 2007.

[21] D. Kim, B. Abay, R. N. Uma, W. Wu, W. Wang, and A. Tokuta, "Minimizing data collection latency in wireless sensor network with multiple mobile elements," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 504–512.

[22] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *In Proc. ACM MobiSys04*, 2004.

[23] R. Sugihara and R. Gupta, "Optimal speed control of mobile node for data collection in sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 1, pp. 127–139, Jan 2010.

[24] I. Papadimitriou and L. Georgiadis, "Energy-aware routing to maximize lifetime in wireless sensor networks with mobile sink," *Journal of Communications Software and Systems*, vol. 2, pp. 141–151, 2006.

[25] R. Bellman, "A markovian decision process," DTIC Document, Tech. Rep., 1957.

[26] R. A. Howard, "Dynamic programming and markov processes.." 1960.

[27] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[28] G. E. Monahan, "State of the arta survey of partially observable markov decision processes: Theory, models, and algorithms," *Management Science*, vol. 28, no. 1, pp. 1–16, 1982.

[29] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, pp. 319–350, 2001.

[30] T. Duong, D. Nguyen-Huu, and T. Nguyen, "Adiabatic markov decision process with application to queuing systems," in *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*. IEEE, 2013, pp. 1–6.

[31] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM REVIEW*, vol. 46, pp. 667–689, 2003.

[32] D. Nguyen-Huu, T. Duong, and T. Nguyen, "Network protocol designs: Fast queuing policies via convex relaxation," *Communications, IEEE Transactions on*, vol. 62, no. 1, pp. 182–193, January 2014.

[33] T. Duong and T. Nguyen, "Fast markov decision process for data collection in sensor networks," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, Aug 2014, pp. 1–8.

[34] T. Morimura, T. Osogami, and T. Shirai, "Mixing-time regularized policy gradient." in *AAAI*, 2014, pp. 1997–2003.

[35] P. Bremaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, ser. Texts in Applied Mathematics. Springer, 1999.

[36] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *In Advances in Neural Information Processing Systems 12*. MIT Press, 2000, pp. 1057–1063.

[37] M. Torki, "Second-order directional derivatives of all eigenvalues of a symmetric matrix," *Nonlinear Anal.*, vol. 46, no. 8, pp. 1133–1150, Dec. 2001.

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[39] T. Duong and T. Nguyen, "Data collection in sensor networks via the novel fast markov decision process framework," *Wireless Communications, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[40] F. Chung, *Spectral Graph Theory*, ser. CBMS Regional Conference Series. American Mathematical Society, 1997, no. 92.

[41] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *Lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, pp. 2004–2005, 2003.

[42] S.Boyd, J. Duchi, and L. Vandenberghe, "Subgradient methods," *Notes for EE364b, Stanford University, Spring 2014-15*, vol. 2014, pp. 2014–2015, 2014.