UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

JOINT OPTIMIZATION OF ENERGY AND DATA BUFFERS IN WSN

(Ottimizzazione congiunta di buffer di energia e dati in reti di sensori wireless)

Laureando

Valentina Girotto

Relatore

Prof. Tomaso Erseghe

ANNO ACCADEMICO 2013/2014

13 OTTOBRE 2014

A coloro che
mi hanno sempre
supportato e sopportato

Io non ho finito
Perché ho sete ancora

N. Agliardi

Indice

Abstract

The study of wireless sensor networks (WSNs) with the use of energy harvesting (EH) technologies is become more and more recurring and evaluated, to the detriment of the traditional networks with non-rechargeable batteries. This change on the subject matter of the research is due to the need of planning more versatile and autonomous devices and placing them also in hostile environments. In this work, we take into account such network in order to study the energy allocation for data transmission and sensing. We consider a single wireless sensor node characterized by a rechargeable energy battery with finite capacity and a data buffer with finite size; in addition, we evaluate the operation in the presence of a processing energy cost and an efficiency constant. The aim is to maximize the total number of data really transmitted, i.e. the instantaneous rate, and, for this purpose, we propose an algorithm, based on a Markov decision process (MDP) and solved using value iteration in order to find the optimal action in terms of transmitted and sensed data. Finally, we compare the results of our optimal policy with the results of other empirical policies.

Sommario

Lo studio delle reti di sensori wireless con l'uso di tecnologie per la raccolta di energia è diventato sempre più frequente ed esaminato, a discapito delle tradizionali reti con batterie non ricaricabili. Il cambio dell'oggetto di studio è dovuto l bisogno di progettare meccanismi più versatili e autonomi e posizionarli anche in ambienti ostili. In questa tesi, prendiamo in considerazione una tale rete per studiare l'allocazione di energia per la trasmissione e il sensing di dati. Consideriamo un singolo sensore wireless caratterizzato da una batteria ricaricabile con capacità finita e un buffer dati di dimensioni limitate; in aggiunta, valutiamo il funzionamento in presenza di un costo di processo e una costante di efficienza. Lo scopo è massimizzare il numero totale di dati realmente trasmessi; a tal fine, proponiamo un algoritmo basato su un processo di decisione di Markov e, risolto usando la *value iteration*, troviamo l'azione ottima in termini di dati trasmessi e ricevuti. Infine, confrontiamo i risultati della nostra soluzione con quelli di altre soluzioni empiriche.

Chapter 1

Introduction

A wireless sensor network (WSN) is a large set of autonomous sensor nodes powered by limited batteries. It can be implemented for many applications, such as traffic monitoring, military tracking, building safety, pollution monitoring, wildlife monitoring, patient security and can be also placed in hostile and unreachable environments, so that batteries can't be changed very often. Moreover, nodes can't use large batteries because of their weight and volume. In such a scenario, the recent EH technology becomes established in order to enable sensor nodes in a self-powered mode for a long time period. Energy is harvested from the environment, as solar or wind energy for example, and then is converted to electrical energy. However, the energy sources aren't active at all time, we just think to the solar energy that isn't available during the night or in a rainy day. So the rate of energy generation can be limited and the energy arrivals haven't a deterministic distribution. Therefore, it is necessary to find an intelligent management of the harvested energy in order to well play the WSN.

Another weak point of a wireless transmitter is the energy consumption because, in addition to the transmission power, we should take into account the battery inefficiencies, [1], [2]; the finite capacity battery, [3], [4]; the sensing cost, [3], [5]; the processing energy cost, [6], [7]. In particular, the last one, the processing energy cost, depends on the communication range and the processing circuitry and heavily conditions the performance. In effect, if the processing cost is negligible, increasing the transmission time and lowering the transmission power is energy efficient if the rate-power functionis non-negative, strictly concave and monotonically increasing. On the other hand, if the processing cost isn't negligible, the optimal transmission scheme becomes bursty because this cost dominates the consumed energy until a certain point, [8], as shown in Fig.1.1.

In this paper, we consider a wireless sensor node, characterized by a finite capacity rechargeable battery and a finite size data buffer, that communicates with a receiver over

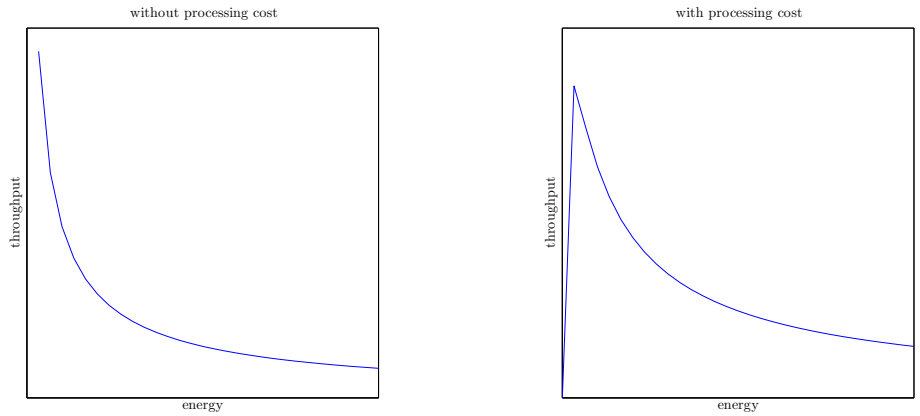


Figure 1.1: Throughput versus energy without and with processing cost

an additive white Gaussian noise (AWGN) fading channel. The objective is to maximize the online throughput, taking into account also the sensing energy process, a constant processing cost and an efficiency parameter. In this way, we want to find an optimal online policy, using value iteration, [9], in order to analyze a more realistic system model.

Looking at the recent literature, many papers focus on at most one or two system inefficiencies; for example, in [10], they consider an additional processing cost and a finite capacity battery; in [11], they consider again a finite capacity battery and a sensing process; in [6], only a non-ideal circuit power. Moreover, most of them propose an offline optimal policy, that assumes the full knowledge of energy arrivals and channel states. Instead, we examine an online solution, characterized by a causal knowledge of the energy arrivals and the channel conditions, given by stochastic processes at transmitter.

This work is organized as follows:

- in Chapter 2 we describe other studies found in the recent literature, based on the analysis of a WSN and then, we introduce our system model;

- in Chapter 3 we analyze more inside two proposed algorithms, the optimal energy allocation (OEA) algorithm of [3] and the directional backward glue-pouring algorithm of [10], in order to better present some other concurrent works;

- in Chapter 4 we first explain the mathematical background of our analysis, represented by the MDP, then, we formulate our policy;

- in Chapter 5 we give some theoretical results of our proposed policy and we prove them;

- in Chapter 6 we first set up the useful parameters, choosing a specific device and a particular application and then we give the numerical results of our policy and we make some comparisons with other empirical policies;
- in Chapter 7 we give some conclusions about our work.

Chapter 2

Related works and our system model

2.1 Related works

Recently, the study of EH has achieved a lot of interest in order to draw "green" techniques and to efficiently manage the harvested energy. In these terms, many different works have been proposed, that can be differentiated depending on the implemented system model or on the choosen policy.

A practical solar EH system model is described in [12], in which they propose a two nested optimization steps solution, providing the optimal operating point and the optimal energy management policy to make the system self-sufficient and taking into account the different exposure to the sun during the day or in different months. They employ the results given in [13], which presents a methodology to model the energy inflow as a function of time through stochastic Markov processes.

More theoretical works are [1], [2], [4], [14], [15], [11], [3], [16], [7], [10], [17], [5] and [6]. First of all, not all the system models include a fading channel, that brings itself some inefficiencies. On the contrary, [4] considers the fading level as a Poisson counting process with rate λ_α, that changes at countable time instants and is known by the transmitter. In [14], they study three different channel models:

- known fading values for each instant;
- random and independent across time fading with a log-normal distribution function;
- fading modeled as a Markov chain, using Rician and Rayleigh distribution functions.

In [15] too, they consider a block fading channel, where the channel gain is a random variable with exponential, Nakagami or log-normal distribution.

Rajesh, in [1] and [2], inserts sleep-wake modes in order to conserve energy when the system has some inefficiencies in energy storage. In effect, every time the sensor node has less energy than that it consumes, it sleeps; every time the sensor node has enough energy than that it consumes, it can decide to sleep and to harvest energy only with probability p.

In some other works, [11] and [3], they take into account also the process of energy allocation for sensing, as in our model. In this way, the sensor node requires a method to decide how much energy it should allocate for sensing and for transmission, depending on the battery energy level, the data buffer level, the energy arrivals and the channel conditions. The process of energy allocation for sensing can be viewed as a process that consumes energy to the detriment of energy for transmission in a throughput maximization policy. More often, instead, as in [2] and [10], data are already available in the data buffer.

Another way to save energy is to use two energy storage devices, as [7] and [16]. [7] considers an hybrid energy storage unit composed of an ideal super-capacitor (SC), that has a finite capacity, and of an inefficient battery with unlimited capacity. The battery can only store energy and transfer it to the SC instantaneously; however, not all the available energy in it can be drained, due to the efficiency parameter η, $0 \leq \eta < 1$. On the other hand, the SC can store energy from the environment and from the battery, ensuring to not have overflows, and can use it to transmit data. Moreover, in data transmission, the transmitter's circuitry has an additional processing cost ξ, given by both the devices. In [16] too, two rechargeable energy storage device (ESD)s form the model: the main ESD receives power and uses it to transmit data, while the secondary ESD stores energy when the main ESD transmits, and, at the end of transmission, transfers its stored energy to the main. Also in this paper, the main ESD is an high-efficient SC and the secondary ESD is a low-efficiency rechargeable battery, charactherized by the parameter η, $0 \leq \eta < 1$.

However, it is more realistic to consider a single device with some inefficiencies in energy storage or with additional energy costs. Examples of such studies are [10], [17], [5], [6], [1] and [2]. In [10], [17], [5], Orhan et al. take into account a constant processing energy cost ξ every time the transmission power is positive, independently of its value. In [6], they consider an on-off transmitter model with non-ideal circuit power, i.e., when the transmitter is on, its consumed power is the sum of the transmission power and a constant circuit power. In [1] and [2], they study two different system models, one in which they include only the energy consumed by sensing and processing, modeled as a random variable; and the other, in which they take into account the inefficiency in storing energy in the buffer and the leakage from the energy buffer, using two different multiplicative parameters, β_1 and β_2, with $0 < \beta_1 \leq 1$ and $0 < \beta_2 < \infty$.

In terms of policies we can first divide them into two approaches, *offline* and *online*. The offline policies assume that the sensor node has full knowledge of the arrival process and the channel gain to find the optimal solution. This setting is unrealistic because it is non-causal and it can't be implemented in practical designs; however, the complete knowledge of the system gives us the instruments to find the optimal solution and the upper bound of the corresponding online policy. On the other hand, the online approaches assume a causal and statistical knowledge about the energy and data arrivals and the channel states, i.e., the transmitter knows only the past and the present of the system features. In such a scenario, the optimal online policy can be found using a dynamic programming (DP) solution, as [9], but these algorithms usually require an high computational complexity. Then, many reseachers propose less complex heuristic online algorithm, that are based on the properties of the corresponding optimal offline policies and perform very close to them.

Most of the papers previously mentioned adopt an offline approach and so a convex optimization problem. A *directional water-filling* has been studied in [4], in which the walls are placed at the points of energy arrival and the water taps are in the right part of each wall. The aim is to maximize the number of bits sent by a deadline. Based on it, in [7], [10] and [5], a *directional glue-pouring* algorithm is adopted, in which each harvested energy packet is allocated to subsequent epochs using the glue-pouring algorithm, [18]. Therefore, [7] and [10] solve the throughput maximization problem and [5] apply it to the remaining energy maximization problem by a deadline and the transmission completion time minimization problem. Differently, [6] considers as an optimal solution a *two-phase* transmission, where the first phase is an energy efficiency maximizing on-off power allocation and the second is a spectrum efficiency maximing power allocation.

Regarding the online policies, [3] formulates the energy allocation and the transmission energy allocation problems as an infinite-horizon MDP and proposes optimal algorithms using the value iteration. In [16], a *save-and-transmit* protocol is introduced, in which in a fraction of time energy is harvested and in the remaining time energy is used for data transmission, so that the aim of it is to minimize the outage probability, i.e., the probability to not transmit data. Finally, some online studies have been proposed, inspired by the corresponding offline solutions, such as [4], [5] and [6].

2.2 Our system model

Our system is a point-to-point model, composed by a transmitter and a receiver that communicate over an AWGN fading channel, as shown in Fig. 2.1. The transmitter is an EH sensor

node TX, connected by two finite queues:

- a *rechargeable battery* with a finite capacity $\overline{b}$ in [J];
- a *data buffer* with a finite size $\overline{q}$ in [Mbit].

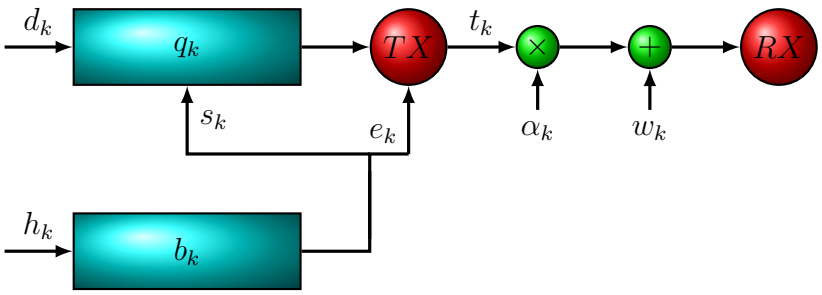


Figure 2.1: System model

We take the system time-slotted, where the duration of each slot is fixed and equal to T, so that for each slot k, with $k \in \mathcal{K} = \{0, 1, 2, \ldots\} = \mathbb{Z}$, the time interval, called *epoch*, is equal to $[kT, (k+1)T)$ and all the characteristics of the system remain the same within the current slot but may change between two consecutive slots. To be more realistic, we assume that the sensor node is an *on-off* node, that alternates periods in which it is on, so it can collect energy from the environment and consume it, and transmit data with periods in which it is off and it does no action. For instance, a photovoltaic panel is on and works during the day and is turned off during the night, when there is no light source. On the other hand, we don't take into account any probability of survival from physical destruction or hardware failure, as in [3].

Looking at Fig. 2.1, when data are transmitted, they are sent through a channel that is characterized by a random channel gain α_k and an additive white Gaussian noise w_k with zero mean and variance $\sigma^2 = N_0 W$, for every slot k. At the end, when data are received, we assume that the receiver RX sends back to the transmitter a causal channel state information (CSI) of the previous instant. So, at each epoch k we know:

- q_k, the amount of stored data in the buffer at the beginning of the slot;
- b_k, the battery level at the beginning of the slot;
- α_k, the channel gain known at TX, for which we consider two cases:

  (a) the channel gain α_k is that of the current epoch k (ideal case);

(b) the channel gain α_k is that of the previos epoch $k-1$ (estimated case);

- h_k, the energy collected from the environment during slot k and available for next slot $k+1$.

Then, during slot k, in addition to h_k and α_k, other quantities are fundamental:

- e_k, the energy allocated for transmission at slot k;
- t_k, the amount of data trasmitted at slot k;
- s_k, the energy needed to sense data that arrive in the data buffer at slot k;
- d_k, the amount of data sensed at slot k;
- p_k, the total consumed energy at slot k, where the sum $e_k + s_k$, is the consumed energy for data sensing and transmission;

Moreover, in order to consider a more practical model, such useful amount of energy is then divided by an efficiency term η, $0 < \eta \leq 1$, and we take into account another constant quantity of dissipated energy, as considered in [1] and in [6], due to the fact that the sensor node itself consumes energy when it is on. We call this dissipated energy ξ and the energy consumption model for the transmitter becomes

$$p_k = \begin{cases} \frac{e_k+s_k}{\eta} + \xi & \text{if node is ON, i.e., } e_k + s_k > 0 \\ 0 & \text{if node is OFF, i.e., } e_k + s_k = 0 \end{cases} \tag{2.1}$$

So p_k is a function of the sum between e_k and s_k and we can simply write (2.1) as

$$p_k = f(e_k + s_k) = \frac{e_k + s_k}{\eta} + \xi 1(e_k + s_k).$$

We consider energy with discretized values according to a given step δ, and when we talk about units of energy we refer to an amount of fixed quantity δ of energy. For the processes of arrivals of data and energy units we follow different approaches:

- the amount of generated data $d(s_k) = d_k$ is a function of the units of energy used for sensing and it is reasonable to keep a monotonically non-decreasing and concave function in s_k, such that as many units of energy are dedicated for sensing as the data that can be accepted and put in the buffer, accordingly with the buffer size. In this way, data that arrive to the system but can't be stored, are discarded. An example of function $d(s_k)$, $k = 0, 1, \ldots$, can be a linear function of s_k, as assumed in [3], $d_k = \beta s_k$, with β called *data-sensing efficiency* parameter.

- the energy units h_k that arrive from the environment and must be stored in the battery are modeled as a Poisson process with rate λ_h, as commonly assumed in literature, in particular in [4]. However, since we have assumed a discrete-time model with a fixed slot duration, we consider in h_k all the energy arrivals that occur in the current time interval $[kT, (k+1)T)$ and are available for slot $k+1$.

With regards to the channel, its gain α_k remains constant during each slot, but it may change from one slot to another, according with a continuos probability density function (PDF). Given that at epoch k the channel gain is α_k and the allocated transmission energy is e_k, for case (a), the sensor node is able to transmit t_k bits of data, called *instantaneous rate*,

$$t_k = r(e_k, \alpha_k) = TW \log_2 \left(1 + \frac{\alpha_k^2 \eta e_k}{N_0 W T \Delta_\Gamma}\right) \quad [\text{bit}], \tag{2.2}$$

where $\frac{\eta e_k}{T}$ is the associated average instantaneous transmission power in $[\text{W}]$ and Δ_Γ is the signal-to-noise ratio (SNR) gap, which depends on the code length, as proved in [19]. For case (b), we postpone to Section 4.2 and Appendix A.1. We note that the instantaneous rate is a concave function because it is a logarithmic function.

By inverting (2.2), we also have

$$e_k = \frac{N_0 W T \Delta_\Gamma}{\eta \alpha_k^2} (2^{\frac{t_k}{TW}} - 1) \qquad \forall\, k. \tag{2.3}$$

On the other hand, we can simply write s_k as a function function of d_k, i.e.,

$$s_k = \beta^{-1} d_k \qquad \forall\, k. \tag{2.4}$$

Therefore, equation (2.1) changes as a function of t_k and d_k too and we refer to it in the following through $\tilde{f}(t_k, d_k, \alpha_k)$;

$$p_k = \tilde{f}(t_k, d_k, \alpha_k) = \begin{cases} \frac{N_0 W T \Delta_\Gamma}{(\eta \alpha_k)^2} (2^{\frac{t_k}{TW}} - 1) + \frac{d_k}{\beta \eta} + \xi & \text{if } t_k + d_k > 0 \\ 0 & \text{if } t_k + d_k = 0 \end{cases} \tag{2.5}$$

It is important to underline the reliance on the channel gain α_k too, derived from the definition of instantaneous rate $r(e_k, \alpha_k)$.

Furthermore, we consider all quantities limited in a positive interval, such as,

$$0 \leq b_k \leq \overline{b};$$

$$0 \leq q_k \leq \overline{q};$$

$$0 \leq h_k \leq \overline{h};$$

$$0 \leq \alpha_k \leq \overline{\alpha};$$

$$0 \leq t_k \leq \overline{t};$$

$$0 \leq d_k \leq \overline{d}.$$

The values $\overline{x}$ will take particular expressions, that we will discover in Section 6.1.

Chapter 3

Other proposed projects

In this chapter, we focus in particular on two different proposed policies, that can be compared with ours. For this purpose, we consider the approach of [11] and [3], from which we have taken the basic structure of their system model; and the offline glue-pouring algorithm, [18], adapted then to an online optimization. For each one, we show the system model, the problem formulation and some results. In the following, we adopt our notation, where possible, to be more clear.

3.1 The *optimal energy allocation* (OEA) algorithm and related works

First, we want to analyze the works, which represent the base of our system model, [11] and [3]. Compared with other recent projects, Mao et al., in [11] and [3], consider also the energy consumed for data sensing, and so they assume to have a finite data buffer.

Their system model is the same of ours and is shown in Fig. 2.1. It is composed by a single EH sensor node, characterized by a rechargeable battery with capacity $\overline{b}$ and a finite data buffer with size $\overline{q}$. At each time slot k the transmitter harvests h_k energy from the environment into the battery; after, by an OEA algorithm, the amount of available energy for transmission (e_k) and for sensing (s_k) is decided. Data are then transmitted over an AWGN channel with block flat fading, whose channel gain values are specified by α_k for slot k. However, they don't consider any other energy cost except for sensing and transmission, while, in [3], they take into account also the probability ν that the sensor node survives after the physical destruction or an hardware failure and continues to function.

In this scenario, they want to maximize the expected total amount of transmitted data

and, for this purpose, they formulate the OEA for sensing and transmission through a MDP and using value iteration, that will be explained in Section 4.1 and following. Concentrating on the OEA given in [3], the objective function of infinite-horizon MDP with discounted reward is

$$J^{\pi}(\mathbf{x}_0) = E\left[\sum_{k=0}^{\infty} \nu^k \gamma(\mathbf{a}_k, \mathbf{x}_k)\Big|\mathbf{x}_0, \pi\right], \tag{3.1}$$

where

- π is a general policy, that contains the decision rules to be used at all slots k, i.e., $\pi = (\delta_0, \delta_1, \ldots) \in \Pi$;
- $\mathbf{x}_k = [b_k, q_k, h_{k-1}, \alpha_{k-1}]$ is the system state at slot k, that includes
  - b_k, the current battery energy state;
  - q_k, the current data buffer state;
  - h_{k-1}, the previous harvested energy state;
  - α_{k-1}, the previous channel state;
- $\mathbf{a}_k = (e_k, s_k)$ is the action taken at slot k for transmission and sensing energy allocation;
- ν is the discount factor, that is the probability of sensor node to survive;
- $\gamma(\mathbf{a}_k, \mathbf{x}_k)$ is the expected amount of data transmitted at slot k, i.e.,

  $$E_{\alpha_k}[\min\{r(e_k, \alpha_k), q_k\}|\alpha_{k-1}]$$

  with $r(e_k, \alpha_k) = TW \log_2\left(1 + \frac{\alpha_k e_k}{N_0 TW\Gamma}\right)$.

At the end, the optimal expected total discounted reward and the optimal policy are defined as

$$J(\mathbf{x}_0) = \max_{\pi \in \Pi} J^{\pi}(\mathbf{x}_0) \qquad \text{and} \qquad \pi^* = \operatorname*{argmax}_{\pi \in \Pi} J^{\pi}(\mathbf{x}_0). \tag{3.2}$$

To solve the optimal policy, they propose an OEA algorithm based on the value iteration, [9]. Therefore, to find the optimal expected total discounted reward $J(\mathbf{x})$, they compute the Bellmann's equation, [9],

$$J(\mathbf{x}) = \max_{\mathbf{a} \in \mathcal{A}(\mathbf{x})} \left[\gamma(\mathbf{a}, \mathbf{x}) + \nu \sum_{\mathbf{y} \in \mathcal{X}} p_{\mathbf{x},\mathbf{y}}(\mathbf{a}) J(\mathbf{y})\right], \tag{3.3}$$

where $\mathbf{y}$ is the future state, characterized by

- $b_{k+1} = \min\{b_k - (e_k + s_k) + h_k, \overline{b}\}$;
- $q_{k+1} = \min\{[q_k - r(e_k, \alpha_k)]^+ + d_k, \overline{q}\}$;
- h_k;
- α_k.

The OEA algorithm, [3], is then composed by two phases, the planning phase and the sensing and transmission phase, whose crucial points are the following:

1. *Planning phase*
2. Arbitrarily select $J_0(\mathbf{x})$ for each $\mathbf{x} \in \mathcal{X}$, specify $\epsilon > 0$, and set $k = 0$.
3. For each $\mathbf{x} \in \mathcal{X}$, compute $J_{k+1}(\mathbf{x})$ by
$$J_{k+1}(\mathbf{x}) = \max_{\mathbf{a} \in \mathcal{A}(\mathbf{x})} \left[\gamma(\mathbf{a}, \mathbf{x}) + \nu \sum_{\mathbf{y} \in \mathcal{X}} p_{\mathbf{x},\mathbf{y}}(\mathbf{a}) J_k(\mathbf{y}) \right].$$
4. If $||J_{k+1} - J_k|| < \frac{\epsilon(1-\nu)}{2\nu}$ go to step 5; otherwise increment k by 1 and go to step 3.
5. For each $\mathbf{x} \in \mathcal{X}$, choose stationary $\epsilon-$optimal policy
$$\delta^*(\mathbf{x}) = \arg\max_{\mathbf{a} \in \mathcal{A}(\mathbf{x})} \left[\gamma(\mathbf{a}, \mathbf{x}) + \nu \sum_{\mathbf{y} \in \mathcal{X}} p_{\mathbf{x},\mathbf{y}}(\mathbf{a}) J_{k+1}(\mathbf{y}) \right] \qquad \text{and stop.}$$
6. *Sensing and Transmission phase*
7. Set again $k = 0$.
8. *while* $k \leq K - 1$ *do*
9. Track the energy harvesting rate of the previous slot h_{k-1}.
10. Track the energy available for use in the battery b_k.
11. Track the amount of data in the buffer q_k.
12. Obtain the channel gain α_{k-1} from the receiver.
13. Set $\mathbf{x} = (b_k, q_k, h_{k-1}, \alpha_{k-1})$.
14. Obtain action $\delta^*(\mathbf{x}) = (e^*(\mathbf{x}), s^*(\mathbf{x}))$ based on the optimal policy.
15. Consume $e^*(\mathbf{x})$ for transmission and $s^*(\mathbf{x})$ for sensing.

16. Update the battery energy b_{k+1} and the data in the buffer q_{k+1}.

17. Set $k = k + 1$

18. *end while*.

In order to evaluate some performance, they consider a special case too, in which they don't take into account the energy for sensing and the data buffer has an infinite size. This algorithm is called optimal transmission energy allocation (OTEA) algorithm and it is able to reduce the computational complexity because the system state is characterized by three elements, $\mathbf{x} = (b, h, \alpha)$, and the action corresponds to the e energy units choosen for transmission. However, to compare the OTEA algorithm with the OEA, they assume that the sensor node allocates a fixed percentage of battery energy for sensing in each slot and they find that the best percentage to transmit the largest amount of data is around 50%, as shown in Fig. 3.1 (Fig. 4 in [3]). Fixed the percentage of 50% for sensing, they study the impact of

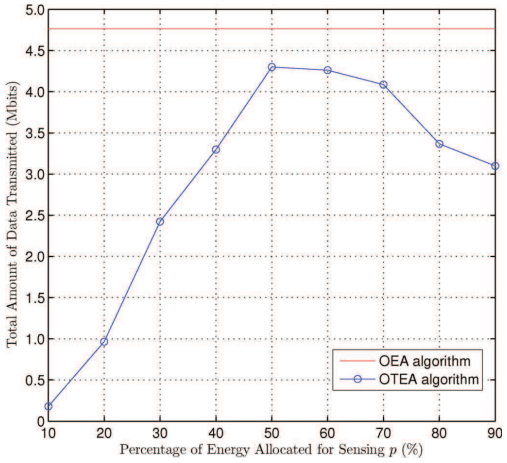


Figure 3.1: total amount of transmitted data of OTEA algorithm under different percentage of energy allocated for sensing

some system parameters on the total amount of transmitted data using both algorithms. For example, they examine the impact of discount factor ν, as in Fig. 3.2 (Fig. 12 of [3]). We can observe that, increasing ν, the lifetime becomes longer and the total amount of transmitted data increases. On the other hand, however, increasing ν requires a larger number of iterations for the value iteration algorithm and a greater computational complexity.

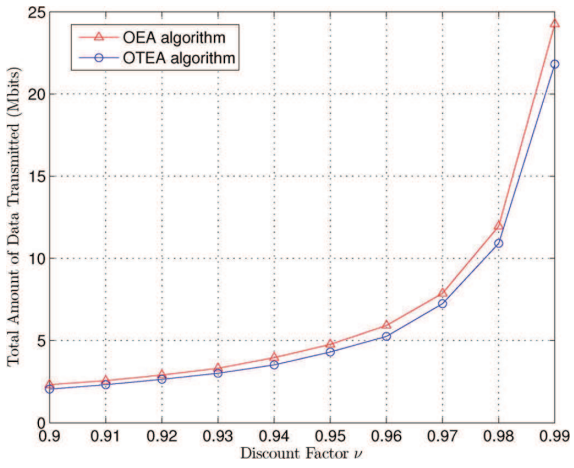


Figure 3.2: Total amount of transmitted data of for different different values of discount factor ν

3.2 The *glue-pouring algorithm* and related works

We present now the glue-pouring algorithm, a policy based on the well-known water filling algorithm and applied to a bursty Gaussian transmission in multiple parallel channels with different noise levels. It has been introduced by [18] and then it has been developed, for example, by Orhan et al. in [10], [17] and [5], for different optimization problems. This technique is adopted in offline optimization problems, where the system is completely known; however, in [17] an online policy is also proposed as a less complex heuristic online algorithm based on the structure of the offline one.

Going in order, when the processing energy cost becomes not negligible, the optimal signal should only transmit in a fraction of the entire time period in order to save energy because increasing the time spent to transmit means increasing also the energy spent for the processing; so the best solution is to apply a bursty transmission. In these terms, the classical water filling is a power allocation process that successively allocates the total signal energy to parallel channels or to consecutive time slots. During the process, each increment of signal energy is done in that sub-channel (or time slot) with the lowest sum of the noise power and the signaling power already allocated. In the glue-pouring algorithm, when the processing cost is considered, a new degree of freedom is used if the sum of signal and noise levels already used is strictly larger than $(1 + \Gamma)$ times the noise level, where Γ the average signal power.

In [10], they consider an EH point-to-point communication system, in which the transmitter has a rechargeable battery with finite capacity $\bar{b}$. The transmitter receives energy packets h_i of finite size at time instants t_i, $i = 0, \ldots, N$; at the same instants, the channel

gains α_i, modeled as an AWGN channel with unit variance, change. The processing energy cost ξ is consumed every time slot the transmitter is on, i.e., the transmission energy is strictly positive.

In such a scenario, the aim is to maximize the throughput; at the same time, we remember that, with non negligible processing cost, the transmission duration in each slot is θ_i, $0 \leq \theta_i \leq \tau_i$, where $\tau_i = t_i - t_{i-1}$ is the epoch, [18]. Therefore, the optimization problem is the following

$$\begin{aligned} \max_{e_i,\theta_i} \; & \sum_{i=1}^{N} \frac{\theta_i}{2} \log\left(1 + \frac{\alpha_i e_i}{\theta_i}\right) \\ \text{st.} \quad & \sum_{j=1}^{i} (h_{j-1} - e_j - \xi\theta_j) \geq 0 \quad \forall\, i \\ & \sum_{j=1}^{i+1} h_{j-1} - \sum_{j=1}^{i} (e_j + \xi\theta_j) \geq \overline{b} \quad \forall\, i \\ & 0 \leq \theta_i \leq \tau_i \quad \forall\, i \\ & e_i \geq 0 \quad \forall\, i \end{aligned} \tag{3.4}$$

where $e_i = \theta_i p_i$ is a new variable, that indicates the transmission energy in order to obtain a convex optimization problem. About the constraints, the first one is the energy causality constraint and the second is the no battery overflows constraint, because of the finite capacity of battery. Therefore, since (3.4) is a convex optimization problem, they resolve it using the Lagrangian of (3.4) with Lagrange multipliers $\lambda_i \geq 0$, $\mu_i \geq 0$, $\gamma_i \geq 0$, $\nu_i \geq 0$ and $\sigma_i \geq 0$, for $i = 1, \ldots, N$,

$$\begin{aligned} \mathcal{L} = & \sum_{i=1}^{N} \frac{\theta_i}{2} \log\left(1 + \frac{\alpha_i e_i}{\theta_i}\right) \\ & - \sum_{i=1}^{N} \lambda_i \left(\sum_{j=1}^{i} (e_j + \xi\theta_j - h_{j-1}) \right) \\ & - \sum_{i=1}^{N} \mu_i \left(\sum_{j=1}^{i+1} h_{j-1} - \sum_{j=1}^{i} (e_j + \xi\theta_j) - \overline{b} \right) \\ & - \sum_{i=1}^{N} \gamma_i (\theta_i - \tau_i) + \sum_{i=1}^{N} \nu_i \theta_i + \sum_{i=1}^{N} \sigma_i e_i. \end{aligned} \tag{3.5}$$

Taking the derivatives of (3.5) with respect to e_i and θ_i and the corresponding complementary slackness conditions, they obtain the optimal transmission power p_i^*, based on the optimal θ_i^*.

- if $\theta_i^* = 0$, then $e_i^* = 0$ and no power is allocated to epoch i, i.e., $p_i^* = 0$;

- if $0 < \theta_i^* < \tau_i$ and $e_i^* > 0$, then $p_i^* = v_i^*$, that is

$$\log\left(1 + \frac{\alpha_i e_i^*}{\theta_i^*}\right) = \frac{\alpha_i(e_i^* + \xi\theta_i^*)}{\theta_i^* + \alpha_i e_i^*};$$

- if $\theta_i^* = \tau_i$ and $e_i^* > 0$, then $p_i^* > v_i^*$, that is

$$\log\left(1 + \frac{\alpha_i e_i^*}{\theta_i^*}\right) > \frac{\alpha_i(e_i^* + \xi\theta_i^*)}{\theta_i^* + \alpha_i e_i^*}$$

  because now $\lambda_i > 0$.

Moreover, λ_i and μ_i can't be simultaneously positive, so, calling the sum of the inverse channel gain and the optimal power level at epoch i, $i = 1, \ldots, N$, i.e. $\left(\frac{1}{\alpha_i} + p_i\right)$ the *glue level* ρ_i, they get

- $\lambda_i > 0$ and $\mu_i = 0$ whenever the battery at transmitter depletes, therefore the glue level at epoch $i + 1$ is greater than that of current epoch;

- $\lambda_i = 0$ and $\mu_i > 0$ whenever the battery at transmitter is full, so that glue level at epoch $i + 1$ is less than that of current epoch.

Finally, they use the above results to play the optimal transmission policy, called *directional backward glue-pouring* algorithm, in which they are able to allocate the harvested energy to epochs starting from the last non-zero energy packet to the first. The steps to find the optimal transmission policy are the following:

1. initialize the glue level for each epoch j, $j = 1, \ldots, N$ to $\rho_j = 0$ and set $i = N$;

2. allocate the energy arrivals h_i of epoch i using the glue pouring algorithm and compute the glue level ρ_i, while satisfying the condition $p_i^* > v_i^*$;

3. set $m = i$. If $m = N$ go to step 6;

4. if $\rho_m > \rho_{m+1}$, reallocate previously allocated energies to epochs $i, \ldots, m + 1$, so that the transferred energy to epoch j, $j = i + 1, \ldots, m + 1$ is less than or equal to the energy into the battery, i.e. $\overline{b} - h_j$;

5. if $m = N$, go to step 6; otherwise increase m by one and go to step 4;

6. if $i = 1$ stop; otherwise, decrease i by one and go to step 2.

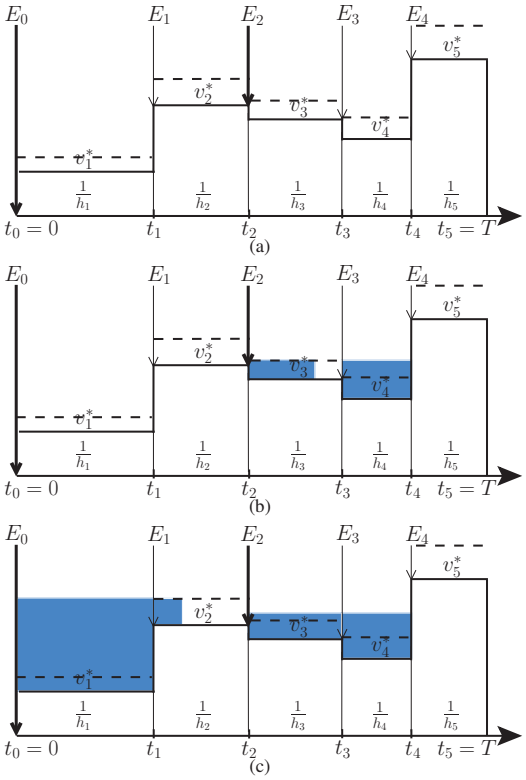


Figure 3.3: Directional backward glue-pouring algorithm

In Fig. 3.3 (Fig. 1 of [10]) there is an example of the directional backward glue-pouring algorithm. Non-zero energy packet arrivals E_i (our h_i) are indicated by thick downward arrows, while the thin downward arrows indicate the zero energy arrivals and, simultaneously, the channel gain changes. The inverse of channel gain $\frac{1}{h_i}$ $\left(\text{our}\frac{1}{\alpha_i}\right)$ are shown with solid blocks and the optimal power levels v_i^* correspond to the blocks from the end of solid blocks to the dashed horizontal lines. Fig. 3.3(a) describes the initial stage; then, the algorithm begins from the last non-zero energy arrival E_2 (our h_2) and the new configuration is shown in Fig. 3.3(b), in which E_2 is allocated to the third and the fourth epochs using glue pouring algorithm. Finally, energy arrival E_0 (our h_0) is considered and is allocated to the first four epochs, as shown in Fig. 3.3(c).

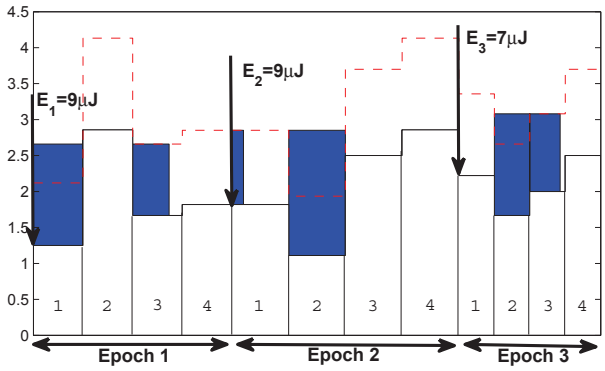


Figure 3.4: Throughput maximization with processing energy cost

In Fig. 3.4 (Fig. 3(b) of [17]) it is shown an example of the optimal transmission policy for the throughput maximization problem, where the blue blocks indicate the optimal power levels. It is important to note that, with non-zero processing energy cost, the optimal transmission policy becomes bursty.

Since we are interested to optimal online policies, we focus on the results given in [17] about the throughput maximization. They use the directional backward glue-pouring algorithm so that the transmitter continues its transmission following the algorithm until the battery depletes or a new event occurs. To evaluate the performance of the online algorithm, they choose an exponential distribution with parameter λ for the channel gain and a uniform distribution in the interval $[0, E]$, with E a random value in a specific energy range, for the energy packets size. They compare the performance of the proposed online algorithm with the corresponding offline algorithm and a DP based solution and the results are given in Fig. 3.5 (Fig. 9(a) of [17]). We can note from it that the offline performance represents the upper

bound for the online policies; on the other hand, the proposed online policy performs close to the DP solution, despite with high energy rates, where the probability of battery overflows increases.

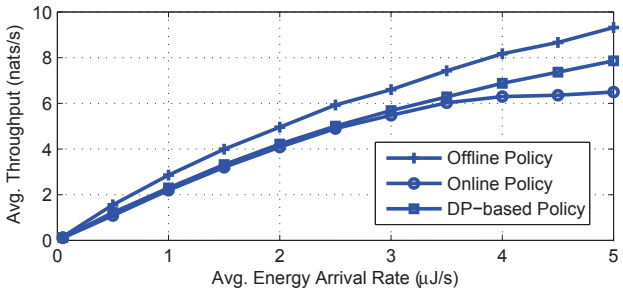


Figure 3.5: Average performance of online and offline throughput maximization

Chapter 4

Dynamic Programming solution

4.1 Markov decision processes

As defined in [20], a MDP is a mathematical framework to model decision making problems in stochastic systems where the state evaluation is partly random and partly under the control of a decision maker. Infact, a MDP is a discrete-time stochastic control process and can be modeled as an extended Markov chain characterized by (in this section we don't use the bold font for states and actions to be more general):

- a countable finite set $\mathcal{X}$ of possible states x, $\mathcal{X} = \{1, 2, \ldots, n\}$;
- a finite admissible region $\mathcal{A}(x)$ of all possible actions a that must be chosen, by observing the state of the process;
- a transition probability $p(y|x, a) = p_{xy}(a)$ from state x at time k to state y at time $k + 1$, given that at time k action a has been chosen;
- an immediate cost function $\gamma(a, x)$, for each transition.

The aim of this problem is to find the policy π that minimizes a particular objective function and this policy is no more than the action associated with the state x, $\pi(x) = a$. However, the set of all possible policies is $\Pi = \{\pi(\cdot) : \pi(x) \in \mathcal{A}(x) \; \forall \; x \in \mathcal{X}\}$.

An important subclass of all policies is the class of *stationary* policies: a *stationary* policy is such that is nonrandomized and the chosen action at time k only depends on the state x of the process at time k. Then, if the policy is stationary, the sequence of states $x_k, \; k = 0, 1, 2, \ldots$, describes a Markov chain and the MDP satisfies the Markov property. To meet the aim of this problem, there are different optimality criteria, including the discounted

problem with bounded cost per stage and the average cost per stage problem, both applied in a infinite horizon, i.e., where the number of decision stages is infinite.

4.1.1 Discounted problem with bounded cost per stage

Given an initial state x_0, as in [9] and [20], we want to find the policy π that minimizes the expected total discounted cost function

$$J_\pi(x_0) = \lim_{N\to\infty} E_\pi \left[\sum_{k=0}^{N-1} \alpha^k \gamma(\pi(x_k), x_k) \right] \tag{4.1}$$

where α is called *discount factor*, $\alpha \in (0, 1)$; and E_π is the expectation given a particular policy π, among all the actions $\pi(x_k)$ associated with the possible transition from state x_k. The meaning of the discount factor α is that, as the time elapses, the corresponding cost function γ becomes less influential than those taken in previous instants.

Given the set of admissible policies π, that corresponds to the set $\mathcal{A}(x)$ of the possible actions, only when an action x is given, the optimal cost function is defined by

$$J^*(x) = \min_{\pi\in\Pi} J_\pi(x) \qquad x \in \mathcal{X} \tag{4.2}$$

and the optimal policy, for a given initial state x_0, is the policy that allows to obtain the optimal cost $J^*(x)$. Moreover, if the policy is stationary, we can write that π is optimal if $J_\pi(x) = J^*(x),\ \forall\ x$, independently of the initial state. Note that, this and the following definitions are rather a system of equations, one for each state, and so, for each state we want to find the optimal cost.

For any function $J : \mathcal{X} \to \mathbb{R}$, we consider now the mapping to J, $T : \mathcal{X} \to \mathcal{X}$, defined as

$$(TJ)(x) = \min_{a\in\mathcal{A}(x)} \left[\gamma(a, x) + \alpha \sum_{y=1}^{n} p_{xy}(a) J(y) \right] \qquad x \in \mathcal{X}, \tag{4.3}$$

where n is the number of states, recalling that the set $\mathcal{X}$ is a countable finite set. We observe that, $(TJ)(\cdot)$ is itself a function and indicates the optimal cost function for the one-stage problem with immediate cost γ and terminal cost αJ. Associating the policy π, for any funtion $J : \mathcal{X} \to \mathbb{R}$ and for any policy π, the referring mapping T_π for the one-stage problem is

$$(T_\pi J)(x) = \left[\gamma(\pi(x), x) + \alpha \sum_{y=1}^{n} p_{xy}(\pi(x)) J(y) \right] \qquad x \in \mathcal{X} \tag{4.4}$$

We can define also the optimal cost for the k-stage, $(T^k J)(x)$, and the cost of a given policy π, $(T_\pi^k J)(x)$, namely

$$(T^k J)(x) = (T(T^{k-1} J))(x) \qquad \text{with} \quad (T^0 J)(x) = J(x) \quad x \in \mathcal{X} \tag{4.5}$$

$$(T_\pi^k J)(x) = (T_\pi(T_\pi^{k-1} J))(x) \qquad \text{with} \quad (T_\pi^0 J)(x) = J(x) \quad x \in \mathcal{X} \tag{4.6}$$

A fundamental result is the following *Monotonicity Lemma*, reported by [9] as Lemma 1.1.1.

Lemma 1. *For any functions $J : \mathcal{X} \to \mathbb{R}$ and $J' : \mathcal{X} \to \mathbb{R}$ such that*

$$J(x) \leq J'(x) \quad \forall\, x \in \mathcal{X}$$

and for any stationary policy $\pi : \mathcal{X} \to \mathcal{A}$, we have

$$(T^k J)(x) \leq (T^k J')(x) \quad \forall\, x \in \mathcal{X}, k = 1, 2, \ldots$$

$$(T_\pi^k J)(x) \leq (T_\pi^k J')(x) \quad \forall\, x \in \mathcal{X}, k = 1, 2, \ldots$$

Moreover, if the cost per stage γ is bounded, i.e. satisfies, for a scalar M,

$$|\gamma(a, x)| \leq M \quad \forall\, (a, x) \in \mathcal{A} \times \mathcal{X}$$

the DP algorithm converges to the optimal cost function J^*, given an arbitrary bounded starting function J. This result is called *Convergence of the DP algorithm* (Proposition 1.2.1 in [9]) and says that

Theorem 2. *For any bounded function $J : \mathcal{X} \to \mathbb{R}$, the optimal cost function satisfies*

$$J^*(x) = \lim_{N \to \infty} (T^N J)(x) \quad \forall\, x \in \mathcal{X},$$

and this is true also for every stationary policy π, whose associated cost function satisfies

$$J_\pi^*(x) = \lim_{N \to \infty} (T_\pi^N J)(x) \quad \forall\, x \in \mathcal{X}.$$

To compute the optimal cost function $J^*(x)$, we must apply the *Bellmann's equation*

$$J^*(x) = \min_{a \in \mathcal{A}(x)} \left[\gamma(a, x) + \alpha \sum_{y=1}^{n} p_{xy}(a) J^*(y) \right] \quad \forall\, x \in \mathcal{X}, \tag{4.7}$$

that is equivalent to

$$J^* = T J^*.$$

Moreover, J^* is the unique solution of the Bellmann's equation in the class of bounded functions. As a corollary, for every stationary policy π, the associated cost function is the unique solution of the Bellmann's equation, that is

$$J_\pi(x) = \left[\gamma(\pi(x), x) + \alpha \sum_{y=1}^{n} p_{xy}(\pi(x)) J_\pi(y) \right] \quad \forall\, x \in \mathcal{X}, \tag{4.8}$$

or, equivalently,

$$J_\pi = T_\pi J_\pi.$$

Then, a stationary policy π is optimal if and only if, $\forall\ x \in \mathcal{X}$, we obtain the minimum in the Bellmann's equation, using that policy $\pi(x)$, i.e.

$$TJ^* = T_\pi J^*,$$

and this is a necessary and sufficient condition for optimality.

To numerically solve the solution of the discounted problem with bounded cost per stage, we apply the *Value Iteration*, that takes the Bellmann's equation and iteratively computes its value to find the optimal cost function, in order to obtain it

$$\lim_{k\to\infty} (T^k J)(x) = J^*(x). \tag{4.9}$$

The value iteration algorithm works as follows

$$(T^{k+1}J)(x) = \min_{a\in\mathcal{A}(x)} \left[\gamma(a,x) + \alpha \sum_{y=1}^{n} p_{xy}(a)(T^k J)(y)\right] \quad \forall\ x \in \mathcal{X}, \tag{4.10}$$

until it converges to the optimal cost function $J^*(x)\ \forall x$, starting from arbitrary initial conditions $J_0(x)$.

To improve the algorithm, we add an *error bounds* condition, such that when the error $|(T^k J)(x) - J^*(x)|$ is small enough, according to a given target ϵ, for each state $x \in \mathcal{X}$, the iteration ends. For this purpose, we define the minimum and the maximum error values as

$$\underline{c}_k = \frac{\alpha}{1-\alpha} \min_{x\in\mathcal{X}} \left[(T^k J)(x) - (T^{k-1}J)(x)\right] \tag{4.11}$$

$$\overline{c}_k = \frac{\alpha}{1-\alpha} \max_{x\in\mathcal{X}} \left[(T^k J)(x) - (T^{k-1}J)(x)\right], \tag{4.12}$$

such that $\underline{c}_k \leq J^*(x) - (T^k J)(x) \leq \overline{c}_k$. If $\Delta_k = \overline{c}_k - \underline{c}_k < \epsilon$, then the algorithm can stop and $J^*(x),\ \forall\ x$, is the optimal cost for the discounted problem.

4.1.2 Relation between discounted and average cost problems

In many situations, it is more practical and useful to consider the *average* cost problem rather than the *discounted* one; however, we can prove that, there is a connection between them, [9]. Starting from the *Laurent series expansion*, for any stationary policy π and a discount factor $\alpha \in (0,1)$, we can connect the cost of π for a α-discounted problem, $J_{\alpha,\pi}$, with the average cost of π, J_π, via the equivalence

$$J_{\alpha,\pi} = (1-\alpha)^{-1} J_\pi + g_\pi + O(|1-\alpha|), \tag{4.13}$$

where g_π can be viewed as a relative cost, i.e., the difference of the total cost of π and the total cost that would be incurred if the cost per stage were the average J_π; and $O(|1-\alpha|)$ is an infinitesimal with respect to $\alpha \to 1$. We can rewrite the Laurent series expansion in terms of J_π

$$J_\pi = (1-\alpha)J_{\alpha,\pi} - (1-\alpha)g_\pi + O(|1-\alpha|^2), \tag{4.14}$$

observing that, the term $(1-\alpha)J_{\alpha,\pi}$ tends to dominate for $\alpha \approx 1$. A stationary policy π is said to be *Blackwell optimal* if it is simultaneously optimal for all the $\alpha-$discounted problems with α in a interval $(\bar{\alpha}, 1)$, where $\bar{\alpha}$ is some scalar with $0 < \bar{\alpha} < 1$. A Blackwell optimal policy is optimal over all policies and, furthermore, it minimizes both the average cost per stage and the $\alpha-$discounted cost for $\alpha \approx 1$. Additionally, for any 2 different Blackwell optimal policies π and π', we have the same total and relative costs

$$J_\pi = J_{\pi'} \qquad g_\pi = g_{\pi'}.$$

Finally, given a Blackwell optimal policy π^*, the couple of costs J^* and g^* satisfies the following pair of equations

$$J^*(x) = \min_{a \in \mathcal{A}(x)} \sum_{y=1}^{n} p_{xy}(a) J^*(y) \quad x \in \mathcal{X} \tag{4.15}$$

$$J^*(x) + g^*(x) = \min_{a \in \mathcal{A}(x)} \left[\gamma(a,x) + \sum_{y=1}^{n} p_{xy}(a) g^*(y) \right] \quad x \in \mathcal{X}, \tag{4.16}$$

and, in particular, J^* is the optimal average cost vector, that includes the optimal average costs for each state of $\mathcal{X}$.

4.1.3 Average problem

As defined in [9] and in [21], given an initial state x_0, the average cost per stage is

$$J_\pi(x_0) = \lim_{N \to \infty} \frac{1}{N} E \left[\sum_{k=0}^{N-1} \gamma(\pi_k(x_k), x_k) \right] \tag{4.17}$$

but, as in discounted problems, the average cost per stage of a policy and the optimal average cost per stage are independent of the initial state. Moreover, for a stationary policy π, it no longer depends of the instant k but it is the same $\forall\, k$.

Then, we can reduce the pair of equations for the Blackwell optimal policies in a single equation, that is called the *Bellmann's equation*: if a scalar λ and a function g satisfy the Bellmann's equation

$$\lambda + g(x) = \min_{a \in \mathcal{A}(x)} \left[\gamma(a,x) + \sum_{y=1}^{n} p_{xy}(a) g(y) \right] \quad x \in \mathcal{X}, \tag{4.18}$$

then λ is the optimal average cost $J^*(x)$

$$\lambda = \min_{\pi} J_\pi(x) = J^*(x).$$

Moreover, if for the policy $\pi^*(x)$ we obtain the minimum of Bellmann's equation $\forall x \in \mathcal{X}$, this stationary policy π is the optimal. On the other hand, considering a single stationary policy π, if a scalar λ_π and a function g satisfy

$$\lambda_\pi + g(x) = \gamma(\pi(x), x) + \sum_{y=1}^{n} p_{xy}(\pi(x))g(y) \quad x \in \mathcal{X}$$

then $\lambda_\pi = J_\pi(x)\ \forall\ x$. As before, we define also the mapping $T : \mathcal{X} \to \mathcal{X}$ for the relative cost g

$$(Tg)(x) = \min_{a \in \mathcal{A}(x)} \left[\gamma(a, x) + \sum_{y=1}^{n} p_{xy}(a)g(y) \right] \quad x \in \mathcal{X}, \tag{4.19}$$

so that, $\forall\ x$, we have $(Tg)(x) = J^* + g(x)$. For a stationary policy π, the referring mapping T_π is

$$(T_\pi g)(x) = \left[\gamma(\pi x, x) + \sum_{y=1}^{n} p_{xy}(\pi(x))g(y) \right] \quad x \in \mathcal{X} \tag{4.20}$$

and satisfies the Bellmann's equation $(T_\pi g)(x) = J_\pi + g(x), \forall\ x$.

In order to identify the value iteration algorithm for the average cost problem, we must introduce the definition of *weak accessibility (WA)* and, for this purpose, we can consider 2 different classes of models:

- *Single-Chain class*, if there are a closed set of states, where each state is reachable from all the other states of the set under some stationary policy, and a possible set of transient states;
- *Multi-Chain class*, if the stationary policy contains 2 or more closed recurrent classes.

Starting from this classification, the Single-Chain class satisfies the WA condition, which says that the set of states can be partitioned into 2 subsets, $\mathcal{S}_t$ and $\mathcal{S}_c$ such that:

- all states in $\mathcal{S}_t$ are transient under every stationary policy;
- for every 2 states x and y in $\mathcal{S}_c$, y is accessible from x.

Then, when the WA condition holds, the optimal average cost is the same for all initial states. In the following, we consider only this case, that is the more reasonable case for our purpose, and so, we can describe the *Value Iteration algorithm*. First, to avoid that some components of $(T^k g)$ diverge to ∞, so that we are not able to find the asymptotical

result, we subtract a fixed value, called δ^k, from all $(T^k g)(x)$, $x \in \mathcal{X}$, every instant k. We take $\delta^k = (T^k g)(t)$, where t indicates a particular fixed state, and this variant on the value iteration method is called *relative value iteration*. Chosen an arbitrary terminal cost function g^0, we can calculate by recursion

$$g^{k+1}(x) = (Tg^k)(x) - (Tg^k)(t) \quad \forall x \in \mathcal{X}, \tag{4.21}$$

so that, as the time horizon grows to infinity, the value iteration algorithm leads up to the optimal average cost $J^* = (Tg^*)(t)$,

$$(Tg^*)(t) = (Tg^*)(x) - g^*(x) \quad \forall x \in \mathcal{X}. \tag{4.22}$$

Finally, as in the discounted problem, we add an *error bounds* condition and we define the minimum and the maximum error values:

$$\underline{c}_k = \min_{x \in \mathcal{X}} \left[(Tg^k)(x) - g^k(x) \right] \tag{4.23}$$

$$\overline{c}_k = \max_{x \in \mathcal{X}} \left[(Tg^k)(x) - g^k(x) \right], \tag{4.24}$$

such that $\underline{c}_k \leq J^*(x) \leq \overline{c}_k$. If $\Delta_k = \overline{c}_k - \underline{c}_k < \epsilon$, for an arbitrary $\epsilon > 0$ then the algorithm can stop and $J^*(x) = \lambda, \ \forall \ x$, is the optimal cost for the average cost problem.

4.2 Problem formulation and policy definitions

We are able now to describe how the model evolves among the time slots and to characterize it by the definitions of state and action.

At each epoch k, as shown in Fig. 4.1, the energy stored in the battery satisfies

$$b_{k+1} = \min\{b_k - p_k + h_k, \overline{b}\} \qquad \forall \ k, \tag{4.25}$$

because the battery has a finite capacity, so if both a large amount of energy arrives and the energy consumption is limited, only $\overline{b}$ units of energy can be accepted. Moreover, there is another constraint because the consumed energy for transmission, sensing and dissipation at slot k can not exceed the available energy b_k, been left from the previous slot, that is,

$$p_k \leq b_k \qquad \forall \ k \tag{4.26}$$

On the other hand, also for the amount of data in the buffer there is a bound,

$$q_{k+1} = \min\{q_k - t_k + d_k, \overline{q}\} \qquad \forall \ k, \tag{4.27}$$

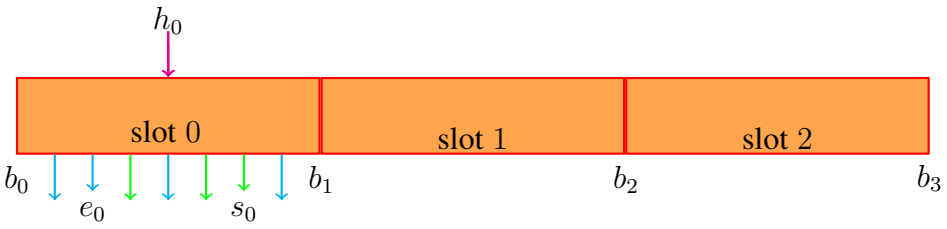


Figure 4.1: Energy arrivals

implying that, if the number of bits that can be transmitted is greater than the number of available bits in the buffer at that slot, due to the large amount of available energy, the buffer empties because the data, that are arriving in it, can't be used until the next epoch. Alternatively, we can set

$$\begin{aligned} q_{k+1} &= q_k - t_k + d_k \qquad \forall\, k \\ &\text{subject to} \\ d_k &\leq \overline{q} - q_k + t_k; \end{aligned} \tag{4.28}$$

where the constraint allows not to waste data, because sensed data can't exceed the available space of the data buffer, represented by the free space at the beginning of the slot k, $\overline{q} - q_k$, and the transmitted data t_k during slot k. From (4.28), we obtain another constraint for the transmitted data, i.e.,

$$t_k \leq q_k \qquad \forall\, k, \tag{4.29}$$

because the node isn't able to transmit more data than those that are stored in the buffer. Finally, for both buffers, we assume to have an initial amount of energy b_0 and data q_0, respectively, that is arbitrary because the cost function for a MDP is independent of the initial state. Given the update fuctions, we continue characterizing states and actions of our system.

The state $\mathbf{x}_k$ is identified by those quantities that are fundamental to describe the node at slot k and that are available during slot k. Therefore, $\mathbf{x}_k$ is denoted as $\mathbf{x}_k = (b_k, q_k, \alpha_k, h_k)$ because

- with b_k we take into account the actual stored energy that depends on the quantities achieved until instant kT, as written in (4.25);
- with q_k we take into account the amount of data stored in the buffer at instant kT that depends on the quantities achieved until it, as written in (4.27);
- with α_k we allow for the characteristics of the fading channel.

- with h_k we include the energy harvested in the current slot k, added in the buffer and available starting from slot $k+1$;

Then the knowledge of h_k is determined by the Poisson process and α_k depends on the case we consider: for case (a), we know it and it is a constant for the entire slot k; otherwise, for case (b), it refers to the value of the previous slot and we must chosen α_{k+1} accordingly the transition probability $P(\alpha_{k+1}|\alpha_k)$. Regarding action $\mathbf{a}_k$, that is taken by the system during slot k, it is established observing the current state $\mathbf{x}_k$ and describes how the available energy can be divided for data sensing and transmission, so $\mathbf{a}_k = (t_k, d_k)$.

The second step is to find the state transition probabilities $P\left[\mathbf{x}_{k+1}|\mathbf{a}_k, \mathbf{x}_k\right] = p_{\mathbf{x}_k,\mathbf{x}_{k+1}}(\mathbf{a}_k)$, which describes the probability to go into a specific state at slot $k+1$ from state $\mathbf{x}_k$, given that the action $\mathbf{a}_k$ is been chosen at slot k. Then, we proceed with the definition of this probability to find out a simpler expression.

$$p_{\mathbf{x}_k,\mathbf{x}_{k+1}}(\mathbf{a}_k) = P\left[\mathbf{x}_{k+1}|\mathbf{a}_k, \mathbf{x}_k\right] = P\left[b_{k+1}, q_{k+1}, \alpha_{k+1}, h_{k+1}|t_k, d_k, b_k, q_k, \alpha_k, h_k\right]$$

First of all, we note that all quantities at slot $k+1$ are independent from each other, given the state and the action at slot k; so,

$$p_{\mathbf{x}_k,\mathbf{x}_{k+1}}(\mathbf{a}_k) = P\left[b_{k+1}|\mathbf{a}_k, \mathbf{x}_k\right] P\left[q_{k+1}|\mathbf{a}_k, \mathbf{x}_k\right] P\left[\alpha_{k+1}|\mathbf{a}_k, \mathbf{x}_k\right] P\left[h_{k+1}|\mathbf{a}_k, \mathbf{x}_k\right].$$

Then, looking at the definitions of those quantities, we can simplify again, removing the variables of slot k, from which they are independent; therefore the final expression is

$$p_{\mathbf{x}_k,\mathbf{x}_{k+1}}(\mathbf{a}_k) = P\left[b_{k+1}|b_k, t_k, d_k, \alpha_k, h_k\right] P\left[q_{k+1}|q_k, t_k, d_k\right] P\left[\alpha_{k+1}|\alpha_k\right] P\left[h_{k+1}|h_k\right]. \tag{4.30}$$

At this point, given the PDFs of α_k and h_k, we know $P\left[\alpha_{k+1}|\alpha_k\right]$ and $P\left[h_{k+1}|h_k\right]$; for the other two probabilities we can observe that they can be easily written as

$$P\left[b_{k+1}|b_k, t_k, d_k, \alpha_k, h_k\right] = \begin{cases} 1 & \text{if (4.25) is satisfied} \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

$$P\left[q_{k+1}|q_k, t_k, d_k\right] = \begin{cases} 1 & \text{if (4.28) is satisfied} \\ 0 & \text{otherwise} \end{cases} \tag{4.32}$$

because we know all the conditioning quantities, since they come from the previous slot, and if they satisfy equations (4.25) and (4.28) respectively, it means that it is possible to continue, otherwise, if something doesn't respect its bounds, the system must stop. Combining all

these results, we have

$$p_{\mathbf{x}_k,\mathbf{x}_{k+1}}(\mathbf{a}_k) = \delta_{b_{k+1},\min\{b_k-\tilde{f}(t_k,d_k,\alpha_k)+h_k,\bar{b}\}}\delta_{q_{k+1},q_k-t_k+d_k} \cdot P\left[\alpha_{k+1}|\alpha_k\right] P\left[h_{k+1}|h_k\right]. \tag{4.33}$$

To obtain the Bellmann's equation (4.18), we need to define the immediate cost function $\gamma(\mathbf{a}_k, \mathbf{x}_k), \forall\ k$, as a function of the current state and the action. The cost function describes the instantaneous reward of the system in terms of total number of bits really transmitted; in other words, what we want to maximize is the instantaneous rate, that depends on the energy allocated for the transmission and the channel gain of the considered slot, so that the immediate cost function is equal to

$$\gamma(\mathbf{a}_k, \mathbf{x}_k) = \gamma(t_k, d_k, b_k, q_k, \alpha_k, h_k) = t_k. \tag{4.34}$$

The last important step to define our policy is to find the set $\mathcal{A}(\mathbf{x}_k)$ of all possible actions that can be taken starting from state $\mathbf{x}_k$ at slot k. Looking at the definition of action $\mathbf{a}_k = (t_k, d_k)$, we observe that t_k and d_k can assume values limited by intervals $[0, \overline{t}]$ and $[0, \overline{d}]$. Moreover, they are correlated since their respective energy quantities are correlated through their sum, being the available energy in the battery distributed among sensing and transmission; therefore, we need to take the intersection between their possible values. Finally, since t_k depends on the data stored in the buffer at slot k, i.e. q_k, we need to take into account this limit too. Then

$$\mathcal{A}(\mathbf{x}_k) = \Big\{(t_k, d_k) \;\Big|\; 0 \le t_k \le \min\{q_k, \overline{t}\}, 0 \le d_k \le \min\{\overline{q} - q_k + t_k, \overline{d}\}, \frac{N_0 W T \Delta_\Gamma}{(\eta\alpha_k)^2}(2^{\frac{t_k}{TW}} - 1) + \frac{d_k}{\beta\eta} + \xi 1(t_k + d_k) \le b_k\Big\}. \tag{4.35}$$

In particular, we can consider two cases, so that the resulting region $\mathcal{A}(\mathbf{x}_k)$ is the union of two sets;

- if $b_k \le \xi$, then we can't trasmit or receive any data, so $t_k = 0$ and $d_k = 0$, obtaining

$$\mathcal{A}_1(\mathbf{x}_k) = \{(0, 0)\}; \tag{4.36}$$

- otherwise, if $b_k > \xi$, then $d_k \le h(t_k, b_k, \alpha_k)$, where $h(t_k, b_k, \alpha_k)$ is a concave function in t_k,

$$h(t_k, b_k, \alpha_k) = \beta\eta\Big(b_k - \xi - \frac{N_0 T W \Delta_\Gamma}{(\eta\alpha_k)^2}(2^{\frac{t_k}{TW}} - 1)\Big), \tag{4.37}$$

obtaining

$$\mathcal{A}_2(\mathbf{x}_k) = \Big\{(t_k, d_k): \quad t_k \leq \min\{q_k, \overline{t}\}, \quad d_k \leq \min\{\overline{q} - q_k + t_k, \overline{d}, h(t_k, b_k, \alpha_k)\}\Big\}. \tag{4.38}$$

Therefore, as illustrated in Fig. 4.2, the resulting region (4.38) is convex. However, we postpone the theoretical proof to Section 5.3.

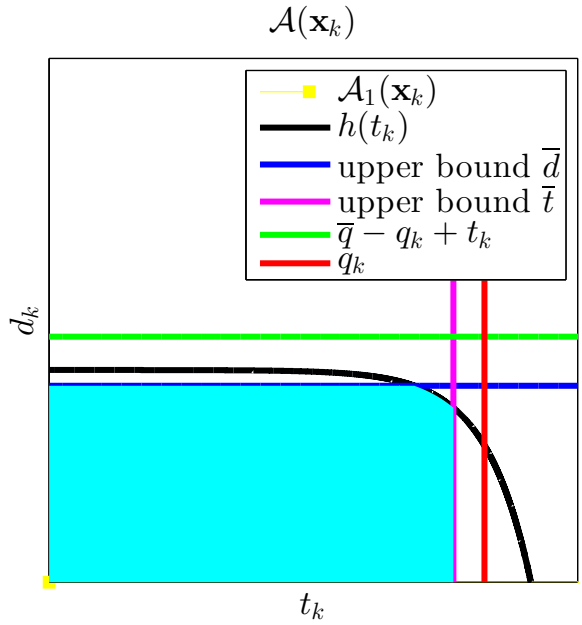


Figure 4.2: Sets that compose $\mathcal{A}(\mathbf{x})$

Finally, after we have defined all the necessary quantities, the Bellmann's equation of our policy is

$$g(\mathbf{x}_k) = \max_{\mathbf{a}_k \in \mathcal{A}(\mathbf{x}_k)} \left[\gamma(\mathbf{a}_k, \mathbf{x}_k) + \sum_{\mathbf{x}_{k+1} \in \mathcal{X}} p_{\mathbf{x}_k, \mathbf{x}_{k+1}}(\mathbf{a}_k) g(\mathbf{x}_{k+1})\right] \qquad \mathbf{x}_k \in \mathcal{X}. \tag{4.39}$$

Chapter 5

Policy properties

In this chapter, we give some theoretical results about the properties of our policy, in order to make the analysis easier and computationally less complex. We first recap some useful definitions given above and then we prove the monotonicity and the concavity of our optimization problem.

5.1 Summary of useful variables and functions

First of all, the state at slot k is characterized by four variables,

- b energy buffer level;
- q data queue level;
- α channel gain;
- h energy arrivals;

so the global state is $\mathbf{x} = [b, q, \alpha, h]$. We can divide it into two parts:

- internal state $\mathbf{x}_i = [b, q]$;
- external state $\mathbf{x}_e = [\alpha, h]$;

The action is represented by data that can be transmitted and data that can be received by the system at slot k, $\mathbf{a} = [t, d]$. Knowing the action $\mathbf{a}$ and the channel gain α, the power used for transmission and sensing is given by (2.5), that is a convex function in $\mathbf{a}$ for a fixed α, as represented in Fig. 5.1.

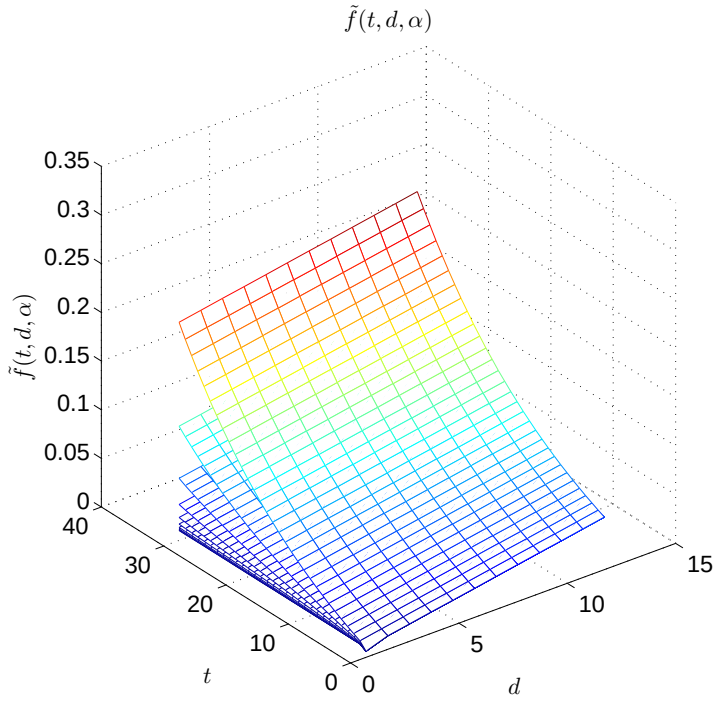


Figure 5.1: $\tilde{f}(t,d,\alpha)$ for any possible value α

The action set contains all possible actions, i.e. all possible pairs (t,d), that can be applied in order to satisfy the bounds imposed by the system and the choosen application, so it is

$$\mathcal{A}(\mathbf{x}) = \Big\{(t,d) \;\Big|\; 0 \le t \le \min\{q,\overline{t}\}, 0 \le d \le \min\{\overline{q}-q+t,\overline{d}\}, \\ \tilde{f}(t,d,\alpha) \le b\Big\}. \tag{5.1}$$

The action set can be written as the union of two sets, as we have already seen,

- $\mathcal{A}_1(\mathbf{x}) = \{(0,0)\}$ if $b \le \xi$;
- $\mathcal{A}_2(\mathbf{x}) = \Big\{(t,d): \quad t \le \min\{q,\overline{t}\}, \quad d \le \min\{\overline{q}-q+t,\overline{d},h(t,b,\alpha)\}\Big\}$ otherwise;

and, by construction, $\mathcal{A}_2(\mathbf{x})$ is a convex set.

The purpose of this analysis is to maximize the transmitted data, so the reward is simply

$$\gamma(\mathbf{a},\mathbf{x}) = t. \tag{5.2}$$

Then, given an action map $\mathbf{a} = \pi(\mathbf{x}) \in \mathcal{A}(\mathbf{x})$, the state update function at slot k is expressed by

$$\begin{aligned} [t, d] &= \pi(b, q, \alpha, h) \\ \begin{bmatrix} \tilde{b} \\ \\ \tilde{q} \end{bmatrix} &= f(t, d, b, q, \alpha, h) \end{aligned} \tag{5.3}$$

where $[\tilde{b}, \tilde{q}]^T = \mathbf{y}_i$ at slot $k+1$ and $f(\mathbf{a}, \mathbf{x})$ is

$$f(\mathbf{a}, \mathbf{x}) = \begin{bmatrix} \min(b - \tilde{f}(\mathbf{a}, \alpha) + h, \overline{b}) \\ \\ q - t + d \end{bmatrix} \tag{5.4}$$

Defined all these quantities, we go on with the value iteration for selecting the optimum action at slot k. Calling $\mathbf{y} = [\mathbf{y}_i, \mathbf{y}_e] = [\tilde{b}, \tilde{q}, \tilde{\alpha}, \tilde{h}]$ the state at slot $k+1$, the steps to execute are

$$\begin{aligned} \mathcal{G}(\mathbf{x}) &= \int g_k(\mathbf{x}_i, \mathbf{y}_e) p(\mathbf{y}_e | \mathbf{x}_e) dy_e \\ U(\mathbf{a}, \mathbf{x}) &= \mathcal{G}(f(\mathbf{a}, \mathbf{x}), \mathbf{x}_e) \end{aligned} \tag{5.5}$$

Then, the corresponding Bellmann's equation is

$$\tilde{g}_{k+1} = \max_{\mathbf{a} \in \mathcal{A}(\mathbf{x})} \gamma(\mathbf{a}) + U(\mathbf{a}, \mathbf{x}), \tag{5.6}$$

and, normalizing it for example with its value at state $\mathbf{0}$, we obtain

$$g_{k+1}(\mathbf{x}) = \tilde{g}_{k+1}(\mathbf{x}) - \tilde{g}_{k+1}(\mathbf{0}) \tag{5.7}$$

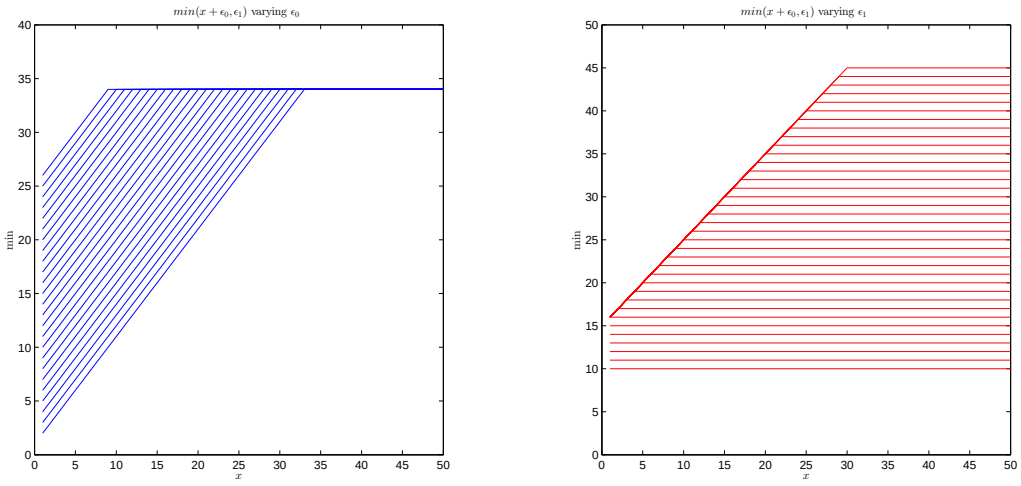
starting from $g_0(\mathbf{x}) = 0$. At the same time, the optimum action at slot k is given by

$$\pi^*(\mathbf{x}) = \arg \max_{\mathbf{a} \in \mathcal{A}(\mathbf{x})} \gamma(\mathbf{a}) + U(\mathbf{a}, \mathbf{x}). \tag{5.8}$$

For the details of how we have found (5.5) and following, we postpone to Appendix B.1.

5.2 Properties of $f(\mathbf{a}, \mathbf{x})$

Lemma 3. *The mapping function $f(\mathbf{a}, \mathbf{x})$, (5.4), is non decreasing in $\mathbf{x}_i$ for fixed $[\mathbf{a}, \mathbf{x}_e]$ and is a concave function in $[\mathbf{a}, \mathbf{x}_i]$ for fixed $\mathbf{x}_e$.*



(a) *varying* ϵ_0 *for fixed* ϵ_1. (b) *varying* ϵ_1 *for fixed* ϵ_0.

Figure 5.2: $min(x+\epsilon_0, \epsilon_1)$

Proof. We can easily prove the monoticity property from the fact that function $\min(x+\epsilon_0, \epsilon_1)$ is concave and non decreasing for any choice of ϵ_0 and ϵ_1. In effect, taking the first entry of function $f(\mathbf{a}, \mathbf{x})$, that we call $m_{\tilde{b}}(b, t, d, \alpha, h)$, and fixing a value for $\mathbf{x}_e$, we can observe its properties from Fig. 5.2. On the other hand, to prove the concavity property, we first define $\check{x} = ux_1 + (1-u)x_2$, with $0 \leq u \leq 1$ and x any variable, that is the convex combination. Moreover, we note that the second entry is linear, hence is straightforwardly concave. Then, we have to prove concavity of f by only proving that it holds for its first entry. Since $\tilde{f}(\mathbf{a}, \alpha)$ is a convex function in $\mathbf{a}$ for fixed α, it verifies

$$\tilde{f}(\check{\mathbf{a}}, \alpha) = \tilde{f}(u\mathbf{a}_1 + (1-u)\mathbf{a}_2, \alpha) \leq u\tilde{f}(\mathbf{a}_1, \alpha) + (1-u)\tilde{f}(\mathbf{a}_2, \alpha), \tag{5.9}$$

$\forall\ \mathbf{a}_1, \mathbf{a}_2 \in \mathcal{A}(\mathbf{x})$ and $\forall\ u \in [0,1]$. At the same time, we recall that $m_{\tilde{b}}(b, t, d, \alpha, h) = w(\tau) = \min(\tau + h, \bar{b})$ is concave and non decreasing in $[b, \mathbf{a}]$ for fixed $\mathbf{x}_e$. Moreover, we take two values $\check{\tau}$ and $\hat{\tau}$ such that

$$\begin{aligned} \check{\tau} = \check{b} - \tilde{f}(\check{\mathbf{a}}, \alpha) &= ub_1 + (1-u)b_2 - \tilde{f}(\check{\mathbf{a}}, \alpha) \\ &\geq u(b_1 - \tilde{f}(\mathbf{a}_1, \alpha)) + (1-u)(b_2 - \tilde{f}(\mathbf{a}_2, \alpha)) = \hat{\tau}, \end{aligned} \tag{5.10}$$

where the inequality is due to the convexity on $\tilde{f}$. Therefore, since $w(\tau)$ is non decreasing,

we also have $w(\check{\tau}) \geq w(\hat{\tau})$. Applying these results, we have for the first entry of f

$$\begin{aligned} f_1(\check{\mathbf{a}}, \check{\mathbf{x}}_i, \mathbf{x}_e) &= w(\check{\tau} + h) = w(\check{b} - \tilde{f}(\check{\mathbf{a}}, \alpha)) = w(ub_1 + (1-u)b_2 - \tilde{f}(\check{\mathbf{a}}, \alpha) + h) \\ &\geq w(u(b_1 - \tilde{f}(\mathbf{a}_1, \alpha)) + (1-u)(b_2 - \tilde{f}(\mathbf{a}_2, \alpha)) + h) \\ &\geq uw(b_1 - \tilde{f}(\mathbf{a}_1, \alpha) + h) + (1-u)w(b_2 - \tilde{f}(\mathbf{a}_2, \alpha) + h) \\ &= uf_1(\mathbf{a}_1, \mathbf{x}_{i1}, \mathbf{x}_e) + (1-u)f_1(\mathbf{a}_2, \mathbf{x}_{i2}, \mathbf{x}_e), \end{aligned} \tag{5.11}$$

where the first inequality is given by the convexity of $\tilde{f}(\mathbf{a}, \alpha)$, proved in (5.9), and by the monotonicity of $w(\tau)$; the second inequality is due to the concavity of $w(\tau)$. From (5.11), we prove the concavity of f. □

5.3 Properties of $\mathcal{A}(\mathbf{x})$

For the action set $\mathcal{A}(\mathbf{x})$ we are able to prove the following results. First of all

Lemma 4. *Let $\mathbf{x}_1$, $\mathbf{x}_2$ be the global states that differ in the value of internal state $\mathbf{x}_i$. Choosen any admissible action pair $(\mathbf{a}_1, \mathbf{a}_2)$, such that $\mathbf{a}_1 \in \mathcal{A}(\mathbf{x}_1)$ and $\mathbf{a}_2 \in \mathcal{A}(\mathbf{x}_2)$, then it is*

$$\check{\mathbf{a}} \in \mathcal{A}(\check{\mathbf{x}}). \tag{5.12}$$

Proof. To prove this statement, we take advantage of the concavity and monotonicity of function $w(\tau)$ and of the convexity of function $\tilde{f}$ in $\mathbf{a}$ for fixed α. Hence, by observing (5.1), we separately analyze its three upper bounds.

- about the first bound, we have

$$\begin{aligned} \min(\check{q}, \overline{t}) &= \min(uq_1 + (1-u)q_2, \overline{t}) \\ &\geq u\min(q_1, \overline{t}) + (1-u)\min(q_2, \overline{t}) \\ &\geq ut_1 + (1-u)t_2 = \check{t} \geq 0, \end{aligned}$$

  where the first inequality is due to the concavity of minimum function, and the second is given by the upper bound of t in the action set definition.

- about the second bound, we have

$$\begin{aligned} \min(\overline{q} - \check{q} + \check{t}, \overline{d}) &= \min(\overline{q} - uq_1 - (1-u)q_2 + ut_1 + (1-u)t_2, \overline{d}) \\ &\geq u\min(\overline{q} - q_1 + t_1, \overline{d}) + (1-u)\min(\overline{q} - q_2 + t_2, \overline{d}) \\ &\geq ud_1 + (1-u)d_2 = \check{d} \geq 0, \end{aligned}$$

  where, again, the first inequality is due to the concavity of minimum function, and the second is given by the upper bound of d in the action set definition.

- finally, about the third bound, we have

$$\begin{aligned}\check{b} - \tilde{f}(\check{\mathbf{a}}, \alpha) &= ub_1 + (1-u)b_2 - \tilde{f}(u\mathbf{a}_1 + (1-u)\mathbf{a}_2, \alpha) \\ &\geq u(b_1 - \tilde{f}(\mathbf{a}_1, \alpha)) + (1-u)(b_2 - \tilde{f}(\mathbf{a}_2, \alpha)) \geq 0,\end{aligned}$$

  where the inequality derives from the convexity of $\tilde{f}$, and so the concavity of $-\tilde{f}$.

Therefore, the convexity of $\mathcal{A}_2(\mathbf{x})$ is proved, observing all the inequalities from rigth to left, and consequently the statement of Lemma 4 □

About the monotonicity,

Lemma 5. *Let $\mathbf{x}_1$, $\mathbf{x}_2$ be the global states that differ only in the value of b. By assuming $b_1 \leq b_2$, then*

$$\mathcal{A}(\mathbf{x}_1) \subset \mathcal{A}(\mathbf{x}_2). \tag{5.13}$$

Proof. We exploit once again the fact that function $w(\tau)$ is concave and non decreasing. Then, we take an action $\mathbf{a} \in \mathcal{A}(\mathbf{x}_1)$; since $\mathbf{a}$ is an admissible action for state $\mathbf{x}_1$, we have from (5.1)

$$\tilde{f}(\mathbf{a}, \alpha) \leq b_1.$$

Moreover, by assumption we know that $b_1 \leq b_2$, therefore

$$\tilde{f}(\mathbf{a}, \alpha) \leq b_2$$

and $\mathbf{a} \in \mathcal{A}(\mathbf{x})$. □

To summarize the results of this section, the action set $\mathcal{A}(\mathbf{x})$ is non decreasing in b for fixed $\mathbf{x}_e$ and, in particular, $\mathcal{A}_2(\mathbf{x})$ is also convex in $\mathbf{x}_i$ for fixed $\mathbf{x}_e$.

5.4 Properties of $g(\mathbf{x})$

Theorem 6. *The function $g_k(\mathbf{x})$ is concave in $\mathbf{x}_i$ for fixed $\mathbf{x}_e$ and it is non decreasing in b for fixed $[q, \mathbf{x}_e]$.*

Proof. We proceed by induction. Since the statement is true for g_0, we prove that, if it holds for g_k, it holds for g_{k+1} too. In other words, we assume that concavity and monoticity hold for g_k. Thanks to this, also $\mathcal{G}(\mathbf{x})$ i a concave and non decreasing function because integration is applied on $\mathbf{x}_e$ and maintains the same properties.

We now prove that also $U(\mathbf{a}, \mathbf{x})$ is a concave function in $[\mathbf{a}, \mathbf{x}_i]$ for fixed $\mathbf{x}_e$ and a non decreasing function in b for fixed $[\mathbf{a}, q, \mathbf{x}_e]$. Beginning from the non decreasing property for

$U(\mathbf{a}, \mathbf{x})$, it derives from the non decreasing property of f, that we have proved in Lemma 3 and for which $f_1(\mathbf{a}, b_1, q, \mathbf{x}_e) \leq f_1(\mathbf{a}, b_2, q, \mathbf{x}_e)$ for $b_1 \leq b_2$. Hence, we find

$$U(\mathbf{a}, b_1, q, \mathbf{x}_e) = \mathcal{G}(f(\mathbf{a}, b_1, q, \mathbf{x}_e), \mathbf{x}_e) \leq \mathcal{G}(f(\mathbf{a}, b_2, q, \mathbf{x}_e), \mathbf{x}e) = U(\mathbf{a}, b_2, q, \mathbf{x}_e), \quad (5.14)$$

where the inequality derives from the fact that $\mathcal{G}(\mathbf{x})$ is non decreasing and so preserves the non decreasing property of f. To derive the concavity of U, we now need to recall the concavity of f in $[\mathbf{a}, \mathbf{x}_i]$ for fixed $\mathbf{x}_e$. Therefore,

$$\begin{aligned} U(\check{\mathbf{a}}, \check{\mathbf{x}}_i, \mathbf{x}_e) &= \mathcal{G}(f(\check{\mathbf{a}}, \check{\mathbf{x}}_i, \mathbf{x}_e), \mathbf{x}_e) = \mathcal{G}(f(u\mathbf{a}_1 + (1-u)\mathbf{a}_2, u\mathbf{x}_{i1} + (1-u)\mathbf{x}_{i2}, \mathbf{x}_e), \mathbf{x}_e) \\ &\geq \mathcal{G}(uf(\mathbf{a}_1, \mathbf{x}_{i1}, \mathbf{x}_e) + (1-u)f(\mathbf{a}_2, \mathbf{x}_{i2}, \mathbf{x}_e), \mathbf{x}_e) \\ &\geq uU(\mathbf{a}_1, \mathbf{x}_{i1}, \mathbf{x}_e) + (1-u)U(\mathbf{a}_2, \mathbf{x}_{i2}, \mathbf{x}_e), \end{aligned} \quad (5.15)$$

where the first inequality is due to the concavity of f, and the second is given by the concavity of $\mathcal{G}$. As a result, U is concave in $[\mathbf{a}, \mathbf{x}_i]$ for fixed $\mathbf{x}_e$.

Then, we need to study the properties of $\tilde{g}_{k+1}$, which correspond to those of g_{k+1} because they differ from each other only by a summation of a constant value for normalization. First of all, we denote

- $\mathbf{a}^* = \pi^*(\check{\mathbf{x}}_i, \mathbf{x}_e)$ the optimum action for state $\check{\mathbf{x}} = [\check{\mathbf{x}}_i, \mathbf{x}_e]$, where $\check{\mathbf{x}}_i$ is the convex combination of $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$;
- $\mathbf{a}_1 = \pi^*(\mathbf{x}_{i1}, \mathbf{x}_e)$ the optimum action for state $\mathbf{x}_1 = [\mathbf{x}_{i1}, \mathbf{x}_e]$;
- $\mathbf{a}_2 = \pi^*(\mathbf{x}_{i2}, \mathbf{x}_e)$ the optimum action for state $\mathbf{x}_2 = [\mathbf{x}_{i2}, \mathbf{x}_e]$;
- $\check{\mathbf{a}} = u\mathbf{a}_1 + (1-u)\mathbf{a}_2$ the convex combination of $\mathbf{a}_1$ and $\mathbf{a}_2$, but not necessarly the optimum action for state $\check{\mathbf{x}}$.

The fact that, in general, $\check{\mathbf{a}} \neq \mathbf{a}^*$ is due to the non linearity of the action map π. Recalling the result of Lemma 4, we have

$$\begin{aligned} \tilde{g}_{k+1}(\check{\mathbf{x}}_i, \mathbf{x}_e) &= \gamma(\mathbf{a}^*) + U(\mathbf{a}^*, \check{\mathbf{x}}_i, \mathbf{x}_e) \\ &\geq \gamma(\check{\mathbf{a}}) + U(\mathbf{a}^*, \check{\mathbf{x}}_i, \mathbf{x}_e) \\ &= \gamma(u\mathbf{a}_1 + (1-u)\mathbf{a}_2) + U(u\mathbf{a}_1 + (1-u)\mathbf{a}_2, u\mathbf{x}_{i1} + (1-u)\mathbf{x}_{i2}, \mathbf{x}_e) \\ &\geq u(\gamma(\mathbf{a}_1) + U(\mathbf{a}_1, \mathbf{x}_{i1}, \mathbf{x}_e)) + (1-u)(\gamma(\mathbf{a}_2) + U(\mathbf{a}_2, \mathbf{x}_{i2}, \mathbf{x}_e)) \\ &= u\tilde{g}_{k+1}(\mathbf{x}_{i1}, \mathbf{x}_e) + (1-u)\tilde{g}_{k+1}(\mathbf{x}_{i2}, \mathbf{x}_e); \end{aligned} \quad (5.16)$$

where the first inequality derives from the optimum action definition, and the second is due to the concavity of U and the linearity of γ. To prove the monotonicity of $\tilde{g}_{k+1}$ in b, we take b_1, b_2 such that $b_1 \leq b_2$ and then we have

$$\begin{aligned} \tilde{g}_{k+1}(b_1, q, \mathbf{x}_e) &= \max_{\mathbf{a} \in ; \mathcal{A}(\mathbf{x}_1)} \gamma(\mathbf{a}) + U(\mathbf{a}, \mathbf{x}_{i1}, \mathbf{x}_e) \\ &\leq \max_{\mathbf{a} \in \mathcal{A}(\mathbf{x}_1)} \gamma(\mathbf{a}) + U(\mathbf{a}, \mathbf{x}_{i2}, \mathbf{x}_e) \\ &\leq \max_{\mathbf{a} \in \mathcal{A}(\mathbf{x}_2)} \gamma(\mathbf{a}) + U(\mathbf{a}, \mathbf{x}_{i2}, \mathbf{x}_e) \\ &= \tilde{g}_{k+1}(b_2, q, \mathbf{x}_e), \end{aligned} \tag{5.17}$$

where the first inequality derives from the monotonicity of U and the second is due to Lemma 5. □

Corollary 7. *The function $U(\mathbf{a}, \mathbf{x})$ is a concave function in $[\mathbf{a}, \mathbf{x}_i]$ for fixed $\mathbf{x}_e$ and it is non decreasing in b for fixed $[\mathbf{a}, q, \mathbf{x}_e]$.*

Chapter 6

Numerical results

6.1 Numerical parameters

Before we go on with the simulation results, a fundamental point is to set up the numerical parameters that characterize the system in order to have a more realistic scenario. Therefore, we focus on a Universal Mobile Telecommunications System (UMTS) technology applied in a latest generation smartphone. A smartphone can be considered as a small sensor node, of dimensions $12 \times 5 \ \mathrm{cm}^2$ (typical of a display of 4 inchs), characterized by a battery of $1500 \ \mathrm{mAh}$ capacity and of $3.5 \ \mathrm{V}$ voltage. About the connectivity, we concentrate on a UMTS network with $5 \ \mathrm{MHz}$ bandwidth, whose maximum uplink transmission rate is fixed to $11 \ \mathrm{Mbit/s}$. On such a device, an application of weather forecast through a webcam is applied; we consider, for example as in [22] and [23], a webcam composed by a mobile camera with image resolution of 160×120 pixel and $24 \ \mathrm{bpp}$, i.e. $\simeq 460 \ \mathrm{Kbit/image}$, and compression format JPEG, so that a compressed image is about $40 \ \mathrm{Kbit}$.

To find the suitable numerical parameters for our model, we start from the solar irradiance, that is a measure of the irradiance produced by the sun in the form of electromagnetic radiation. We consider a constant and fixed value for the solar irradiance, $300 \ \mathrm{W/m}^2$, that is the maximum solar irradiance at midday. However a solar panel has an efficiency of 20%, therefore the real solar irradiance on it is 'only' of $60 \ \mathrm{W/m}^2$. Then we examine a device of dimensions $12 \times 5 \ \mathrm{cm}^2$, as introduced before, so that the real maximum power that it can receive is $P = 60 \cdot 12 \times 5 \cdot 10^{-4} = 0.36$ W. From this result, we are able to fix one by one the other parameters.

First, we set up the time slot T to $10 \ \mathrm{ms}$, that is a reasonable coherence time in the presence of a Rayleigh channel, then we can derive the rate of energy arrivals in a slot into

the system battery,

$$\bar{\lambda_h} = P \cdot T = 0.36 \text{ J/s} \cdot 10^{-2} \text{ s/slot} = 3.6 \cdot 10^{-3} \text{ J/slot}.$$

Moreover, as introduced in the previous section, we set the energy arrival in a slot as a sum of some $\delta-$units of energy that arrive to the system according to a Poisson process of rate λ_h. For this purpose, we take $\delta = 3.6 \cdot 10^{-3}$ J and we can derive the average number of $\delta-$unit energy arrivals into the system as

$$\lambda_h = \frac{\bar{\lambda_h}}{\delta} = \frac{3.6 \cdot 10^{-3} \text{ J/slot}}{3.6 \cdot 10^{-3} \text{ J}} = 1 \text{ arrival/slot}.$$

A fundamental parameter that characterizes the system model and conditions the system performance is ξ. ξ is the amount of energy that is dissipated in a slot of 10 ms, everytime some energy for data sensing and/or transmission is sent from the battery. To set up its value, we consider the battery capacity of 1500 mAh, that typically lasts 24 h; then, taking into account the standard voltage 3.5 V, we find the average current that flows on the battery and the average consumed power.

$$\bar{I} = \frac{1500 \text{ mAh}}{24 \text{ h}} = 62.5 \text{ mA} \qquad \bar{P_c} = 3.5 \text{ V} \cdot 62.5 \text{ mA} = 0.21875 \text{ W}.$$

Knowing that [W] = [J/s], the average consumed energy in a slot T is

$$\bar{p_c} = \bar{P_c} T = 0.21875 \text{ J/s} \cdot 10^{-2} \text{ s} = 2.1875 \cdot 10^{-3} \text{ J}.$$

Therefore, it is reasonable to fix $\xi = \delta$.

As done for energy, we must give a δ_d-unit for the amount of data, that is thesize of one packet, that we assume to be constant and known; for example, in order to have simple calculation, we set $\delta_d = 3.6$ Kbit, that is a reasonable parameter too, because usually the Maximum Transmission Unit (MTU) of an IP packet that can be transmitted without fragmentation is 1500 byte $= 12$ Kbit over Internet. This number is comparable to the size of an image taken by a webcam (for example to monitor the weather in a particular mountain place), with the characteristics given above, which generates about $11\delta_d$ bit per image.

Regarding the value of β, that connects data and energy needed for the data sensing, first, we assume that the relation between sensing energy and data is a linear relation managed by β, $d_k = \beta s_k, \forall\ k$. We now suppose that in a slot of 10 ms a single energy $\delta-$unit arrives into the system; in this case, if the battery is empty and this energy amount is dedicated to the data sensing, only one δ_d-unit of data can arrive, that is only one packet. So, we are able to find the correct β,

$$\beta = \frac{\delta_d}{\delta} = \frac{3.6 \cdot 10^3 \text{ bit}}{3.6 \cdot 10^{-3} \text{ J}} = 1 \text{ Mbit/J}.$$

This result can be proved also as follows: to read 10 bits on average we need $1\ \mathrm{mW}$ in a slot T, therefore we need $1\ \mathrm{mW} \cdot 10^{-2}\ \mathrm{s} = 10^{-5}\ \mathrm{J}$. Then the number of bits arriving into the system when a Joule of energy is available is exactly β, that is $10\ \mathrm{bit}/10^{-5}\ \mathrm{J} = 1\ \mathrm{Mbit}$.

Starting from the above considerations, another important energy parameter is the capacity of the battery $\overline{b}$; it tells us the maximum amount of energy that it is able to support in a time slot because every 10 ms the battery state b_k changes acconding to the energy arrivals and departures, as given in (4.25). Therefore $\overline{b}$ must be compatible with the Poisson arrivals rate λ_h, that we have found, and with the actions that can be taken during a slot T. For this purpose, in general, we need:

- 1δ of energy to store δ_d data that are sensed;
- 1δ of dissipated energy;
- 1δ of energy to transmit δ_d data;

in other words, we need on average 3δ every slot to read and transmit data. Given this result, it is sufficient to have a battery capacity of $\overline{b} = 50\delta$, to be sure to store enough energy.

Then, we choose an appropriate value for $\overline{q}$, the maximum possible amount of bits that can be stored into the data buffer every time slot of $10\ \mathrm{ms}$. In order to find it, we can fix a greater bound, considering the maximum battery bound $\overline{b}$ and the value of β, that relate them; therefore,

$$\overline{q} = \beta\overline{b} = 1\ \mathrm{Mbit/J} \cdot 50\delta = 180\ \mathrm{Kbit} = 50\delta_d.$$

We can find also an upper bound for the sensing data, $\overline{d}$; to set up it, we observe the chosen application: on average, the system can receive and store one δ_d-packet and, at the same time, can transmit one δ_d-packet. However, the complete JPEG-image takes about $40\ \mathrm{Kbit}$, i.e. $\simeq 11\delta_d$, so the upper bound for the sensed data can be set up to $\overline{d} = 11\delta_d$.

Finally, we look for an upper bound also for the trasmitted data, i.e., $\overline{t}$; for this purpose, we consider the maximum transmission rate for uplink $v = 11\ \mathrm{Mbit/s}$ of a UMTS technology, so that the maximum amount of bits that can be transmitted from the system in a slot T is

$$\overline{t} = v \cdot T = 11\ \mathrm{Mbit/s} \cdot 10^{-2}\ \mathrm{s} = 110\ \mathrm{Kbit/slot} \simeq 30\delta_d.$$

To resume all the numerical parameters needed for the analysis, we can look at the following table.

Variable	Meaning	Value
T	time slot duration	10 ms
δ	energy unit	$3.6 \cdot 10^{-3}$ J
δ_d	packet length	3600 bit
λ_h	energy arrival rate	$1\delta/\text{slot}$
$\overline{b}$	battery capacity	50δ
ξ	dissipated energy	δ
β	linear coefficient for data sensing	1 Mbit/J
$\overline{q}$	data buffer size	$50\delta_d$
η	efficiency	90%
N_0	power spectral density	10^{-8} W/Hz
W	bandwidth	5 MHz
Δ_Γ	SNR gap	1
$\overline{t}$	maximum trasmission energy	$30\delta_d$
$\overline{d}$	maximum sensing data	$11\delta_d$

Table 6.1: Summary of numerical parameters

6.2 Simulations for the optimal action

In this section, we figure the obtained results of our proposed policy in order to graphically prove the properties stated in Chapter 5. In effect, the following figures show the optimal actions, in terms of transmitted and sensed data, on varying the energy arrivals and the channel state and as functions of the battery and data buffer states. To perform them, we have implemented a Matlab code, reported in Appendix B.2, in which we compute the reward and the optimal action of our policy using a mex function to implement the value iteration, until the reward converges. In all the following figures, we denote X axis with the percentage of the energy battery or the data buffer; Y axis with the number of δ_d-units of data and the curves represent the optimal action for different values of q or b, with respect to the X axis.

From Fig. 6.1, where we plot the optimal transmitted data action as a function of the energy battery state, we observe that, the transmitted data increase as the channel gain becomes higher, i.e., the channel becomes better. On the other hand, the increase is less perceptible, when the number of energy arrivals increases. Therefore, the transmitted data are independent of the energy arrivals because the useful energy for transmission depends on the energy already saved on the battery, while the energy arrivals become usable from next slot. Regarding channel state, when the channel gain is null (i.e. $-\infty$dB), no data can be transmitted, so the optimal action t is always null, even if the data buffer is full. Increasing the channel gain, data can be transmitted only when there is enough energy in the battery and enough data in the buffer. Finally, we can note that, when the channel gain reachs its maximum value (i.e. 10dB), the transmitted data can't exceed the upper bound $\bar{t}$, imposed by the choosen device, even if the battery and the data buffer are full.

In Fig. 6.2 we plot the optimal sensed data action as a function of the data buffer state, again for different values of energy arrivals and channel state. We observe that, the sensed data too don't depend on the energy arrivals for the same reason. Moreover, the optimal action d is quite independent of the channel gain because sensed data are first stored in the data buffer, so they don't pass through the channel. Increasing the data buffer state q, the curves follow now a decreasing trend, in order to not waste data, because when data buffer is quite full, no other data are accepted and this is due to the constraint in (4.28). Another time, we can note that the maximum number of sensed data units that arrive into the system is that imposed by the choosen application.

It is interesting to observe also the sum of the optimal transmitted data and sensed data action, i.e. the amount of data operating in a time slot inside the system. For this purpose, Fig. 6.3 shows the optimal sum as a function of the energy battery state. We can note that,

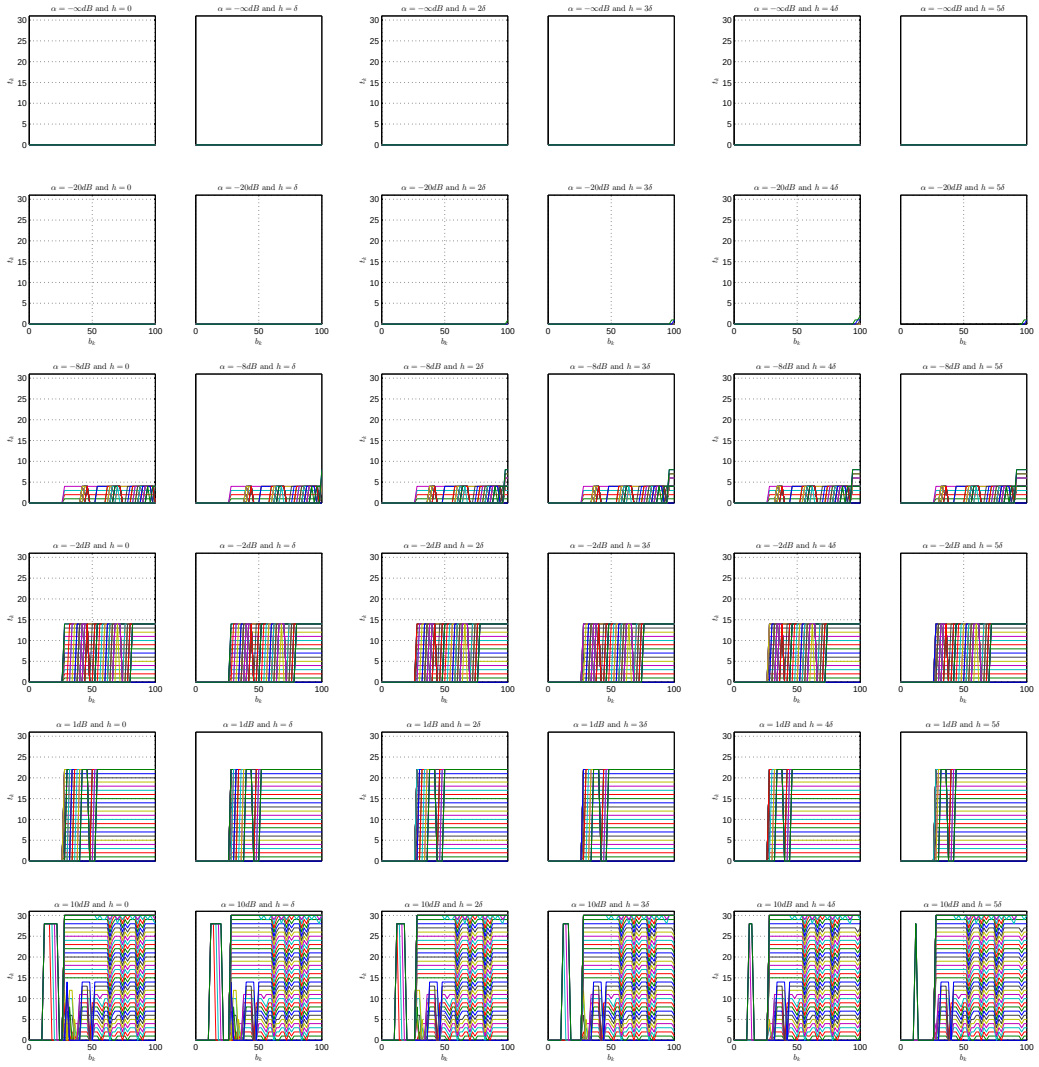


Figure 6.1: Optimal action t as a function of b

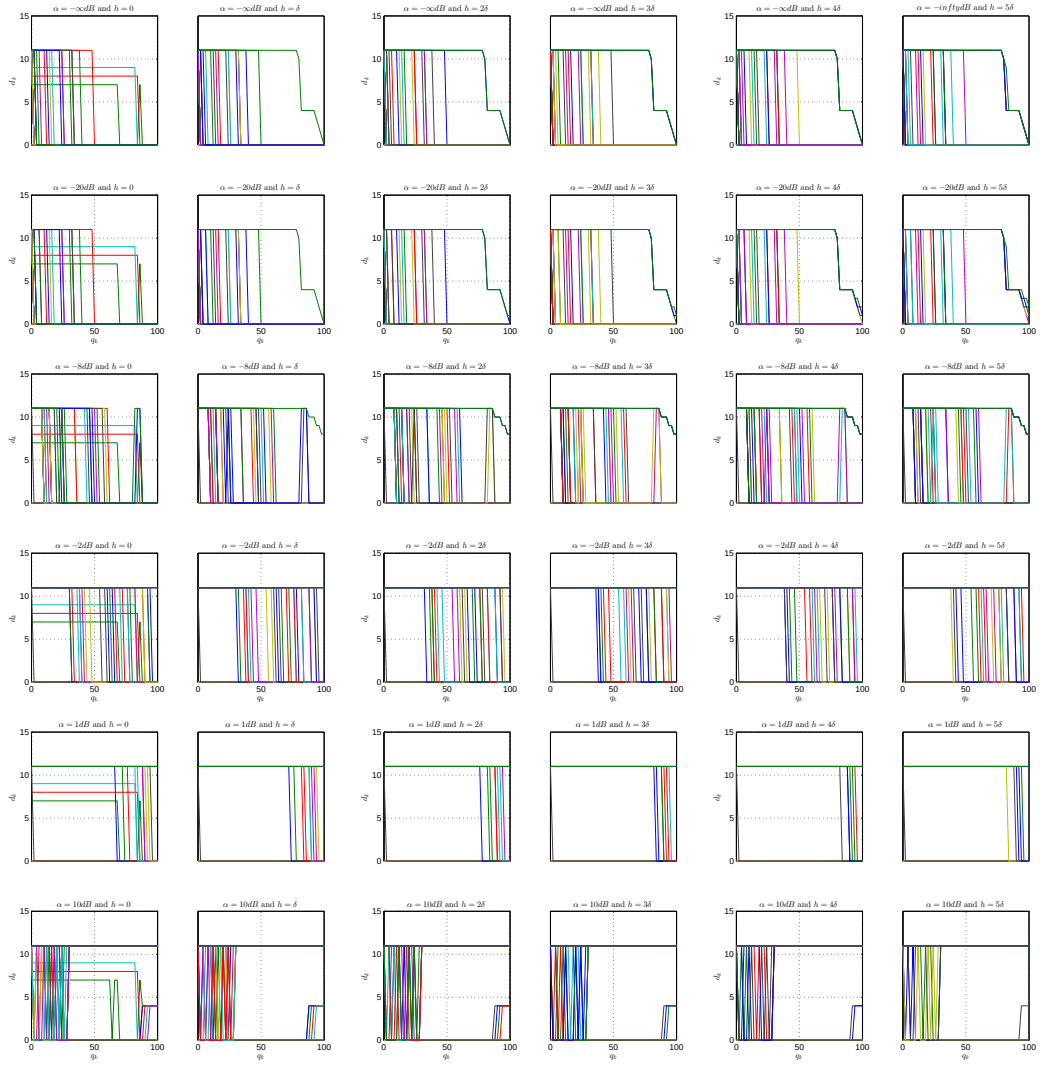


Figure 6.2: Optimal action d as a function of q

when the channel is null, only sensing is allowed, and starting from a sufficient level of energy battery, that is approximately 25%; moreover, according to the data buffer state, the sensed data are almost always the maximum possible because energy is exclusively dedicated to sensing. Then, increasing the channel gain, the optimal sum is non-decreasing as a function of b, reaching the upper bound $\overline{t} + \overline{d}$ when the channel gain takes its maximum value; however the upper bound is variable, depending on the constraints given by 4.29 and 4.28.

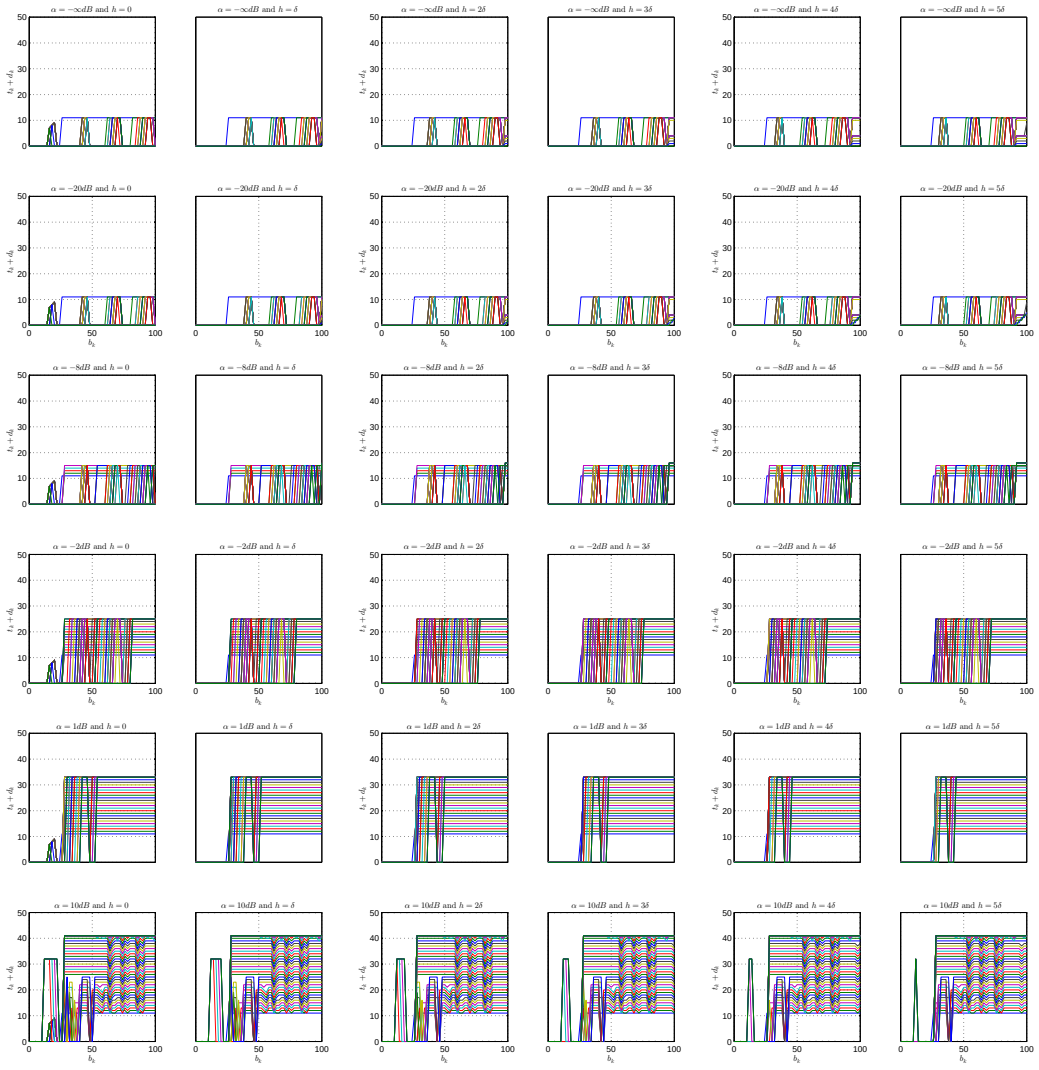


Figure 6.3: Optimal sum $t + d$ as a function of b

In the same way, we can observe Fig. 6.4, where the optimal sum of transmitted and sensed data is plotted as a function of the data buffer state q. In Fig. 6.4, there are two different trends: when the channel state is bad, there is a non-increasing trend; when the channel state becomes better, the curves are non-decreasing. In effect, when the channel state is bad, data can't be transmitted, they can be only sensed, so we find again the results

of Fig. 6.2. Increasing the channel gain, the receiver is able to receive data, so increasing the data buffer state allows to transmit more and more data up to the upper bound $\overline{t}$.

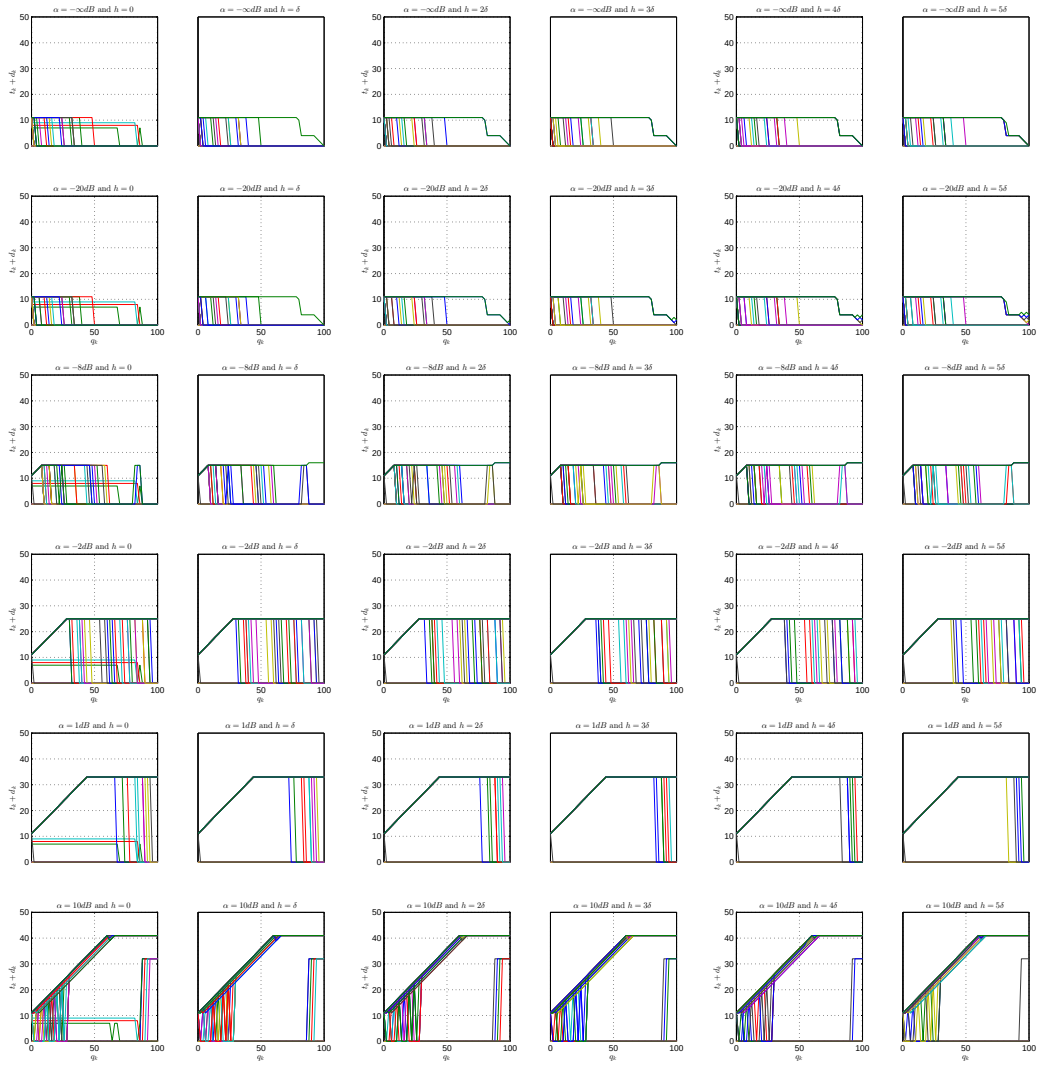


Figure 6.4: Optimal sum $t + d$ as a function of q

We can observe also the influence of the data-sensing efficiency parameter β on the performance of the system. Considering the linear relation between sensing data and sensing energy, i.e. $d_k = \beta s_k \ \forall \ k$, we have found in Section 6.1 that, if we have a single energy $\delta-$unit arrival and it is dedicated to sensing, only one δ_d-unit of data can arrive to the system and β is equal to 1Mbit/J. If we increase the value of β, keeping fixed δ and δ_d, we increase the number of data packets that can be sensed with a single $\delta-$unit energy; in other words, we impose to the system to allocate less energy for sensing, in favour of transmission. However, due to the finite size of data buffer, β can't grow to infinity, because no more than $\overline{q}$ data can be accepted. In addition, we have set also an upper bound for sensed data, $\overline{d}$, that is imposed by the choosen application. Therefore, in this particular setting, the maximum

admissible value of β is given by the ratio between $\overline{d} = 11\delta_d$ and 1δ of energy, that is the minimum quantity of energy; so

$$\beta_{\max} = 11 \text{ Mbit/J}.$$

On the other hand, we can also decrease the value of β, so that we need more $\delta-$units of energy for a single sensed δ_d packet. For example, if we choose $\beta = 0.1$ Mbit/J, then, for a single δ_d packet, we need

$$s = \frac{\delta_d}{\beta} = 10\delta.$$

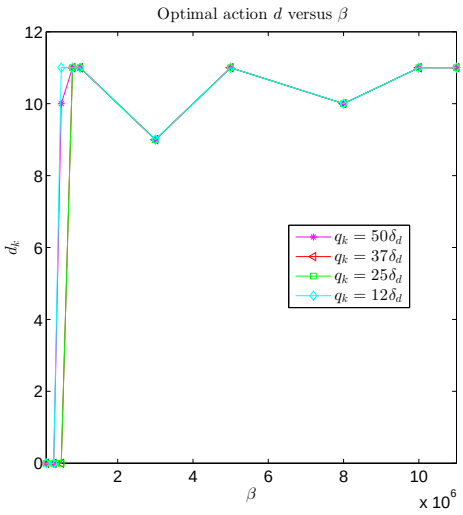
In this way, we are able to find the minimum admissible value of β, imposing that all the available energy into the battery, i.e. $\overline{b}$, is dedicated for sensing a single δ_d data packet; so we obtain

$$\beta_{\min} = 2 \cdot 10^4 \text{ bit/J},$$

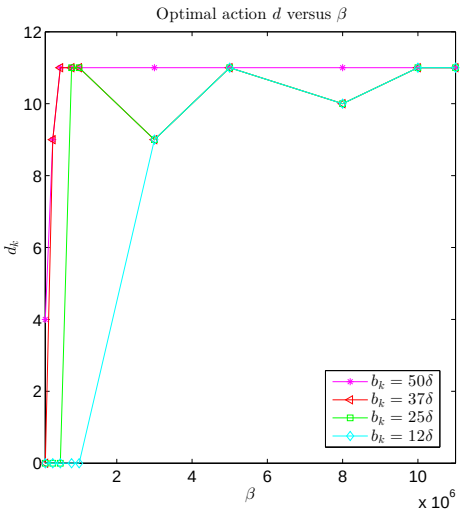
with which the reward, in terms of transmitted data, is null. In fact, using a $\beta < 10^6$ bit/J, the performance gets worse, because less data are available in the buffer and more energy is necessary to sense the same amount of data; in other words, sensing becomes more influential for the system performance and the reward is worse.

For this purpose, it is interesting to observe the optimal action d as a function of β values, for different values of data buffer state, as in Fig. 6.5 (a), and for different values of battery state, as in Fig. 6.5 (b), keeping fixed the channel gain and the energy arrivals states. In Fig. 6.5 (a), we note that, independently of the data buffer state, as much as β is less than 1 Mbit/J, the action d is null because there is no much energy to allocate for sensing (we have choosen $b = 25\delta$, i.e. half-full battery). When β increases, less energy is necessary for sensing the same amount of data, so the action d becomes different from 0; in particular, even if the battery is not full, the maximum amount of data is sensed for $\beta_{\max}$. In Fig. 6.5 (b), the results are a little different because, when the battery is full or three-fourths full, the action d is quite always the maximum because there is enough available energy, independently of the value of β; however, when the available energy decreases, the system is able to sense data starting from higher β values.

On the other hand, in Fig. 6.6 we plot the optimal action t as a function of battery state, for different values of β, keeping fixed the channel gain, the energy arrivals and the data buffer states. Focusing, for example, on the black curve for $\beta = 10^5$ bit/J, we observe that we are able to transmit something only when battery is quite full, so the system has enough energy for sensing and then transmitting. Increasing β, then more data are avilable, so we can already transmit starting from lower battery states; however, we must note that the upper bound for transmitted data is given by the data buffer state, that in this case is $25\delta_d$.



(a) For different values of data buffer state and $b = 25\delta$



(b) For different values of battery state and $q = 25\delta_d$

Figure 6.5: Optimal action d with $\alpha = 7dB$ and $h = 10\delta$

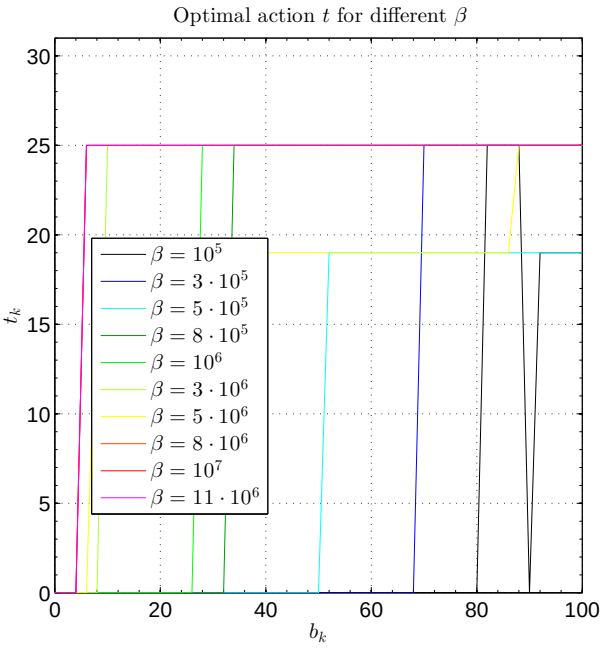


Figure 6.6: Optimal action t for different values of β with $q = 25\delta_d$, $\alpha = 7dB$, $h = 10\delta$

For the reward performance, we postpone to next section, in which we compare the optimal reward of our policy with the reward of other empirical policies.

6.3 Comparison with empirical policies

It is interesting and useful now to compare the results of our optimal policy with other empirical policies in order to prove the optimality of ours. For this purpose, we choose two different empirical policies, whose code is reported in Appendix B.2, which both use the same features of ours in terms of energy arrivals and channel states to obtain the same behaviour for all the three policies. Before showing the graphical results, we need to describe the operation of both the empirical policies.

Policy 1. The first policy decides slot by slot how much data the system is able to allocate for transmission, choosing the minimum among the data available in the buffer, the data that the system can transmit depending on how much energy there is in the battery and the maximum of transmitted data. After that, depending on the real energy used for transmission, the remaining available energy can be allocated for sensing. For this purpose, the sensed data action is taken looking for the minimum between the data that the system can accept, based on the available energy for sensing, the empty space available in the buffer and the maximum of sensed data. Finally, given the action (t, d) of current slot k, we are then able to compute the state update functions (4.25) and (4.27).

Policy 2. Contrary to the first, the second policy makes a decision based only on how much energy is available, because the aim is to transmit as much data as possible as soon as there is enough available energy in the battery. After that, depending on how much data can still be stored in the buffer, the system chooses to accept as much data as possible. Again, given the action (t, d) of current slot k, we are then able to compute the state update functions (4.25) and (4.27).

Obtained the actions for all the three policies, we evaluate the performance in terms of reward, i.e. average of transmitted data, computed in a long time period. First of all, we observe Fig. 6.7, in which we plot the normalized reward for all the three policies as a function of different energy battery capacities, keeping fixed the data buffer size at $50\delta_d$, as imposed in table 6.1. We first note that, for all the three policies, the reward already saturates with a small capacity of energy battery; so it is sufficient to have a smaller capacity to have good performance for the choosen application. Moreover, we prove the optimality of our policy against the other two proposed policies, because it achieves an higher reward. Policy 2 achieves a small reward because of its approach of transmitting as much as possible when

there is enough energy. In effect, Policy 2 has a more marked burstiness because, when there is enough energy into the battery, the system decides to use all of it in a slot, so that the following slots are necessary to harvest energy and no data are transmitted. The same behaviour is taken for sensing because sensing is also concentrated on a single slot, in which the data buffer is filled up. So, in its entirety, Policy 2 is absolutely not efficient because it is a too much aggressive policy, since it doesn't save energy for future slots. On the other hand, Policy 1 achieves a better reward because it evaluates how much energy and data can be allocated in a slot in order to respect all the bounds given by the system. However, its reward isn't the optimal one, because it has a too much conservative behaviour since, in every slot, it decides for the minimum in order to save as much energy and data as possible for next slots.

For these reasons, our policy achieves the optimal reward, because it reaches a compromise between a too much aggressive policy, as Policy 2, and a too much conservative policy, as Policy 1.

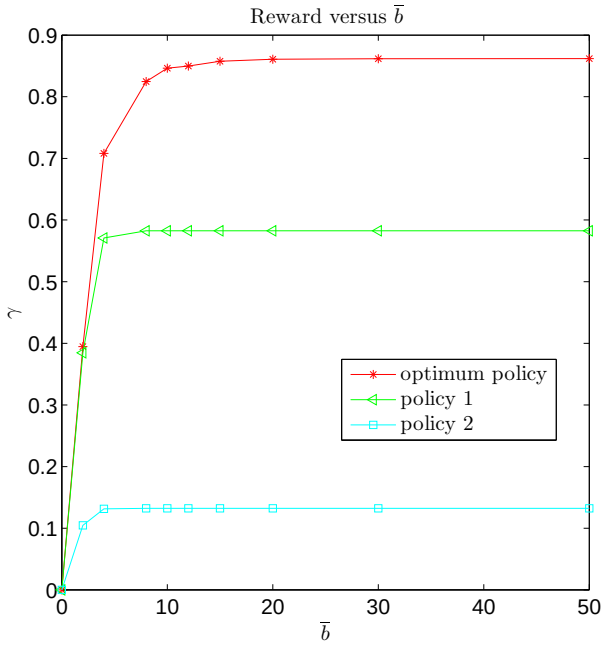


Figure 6.7: Normalized reward γ for different energy battery capacities $\overline{b}$

We find the same results also in Fig. 6.8, in which we plot the normalized reward as a function of different data buffer sizes, keeping fixed now the energy battery capacity to 50δ, as imposed in table 6.1. Once again, the reward saturates with a small size of data buffer and our policy performs the best reward against the other two empirical policies.

Finally, we plot in Fig. 6.9 the reward, simultaneously, for different values of $\overline{b}$ and $\overline{q}$. In particular, focusing on the first figure, that shows the reward of our policy, we observe that

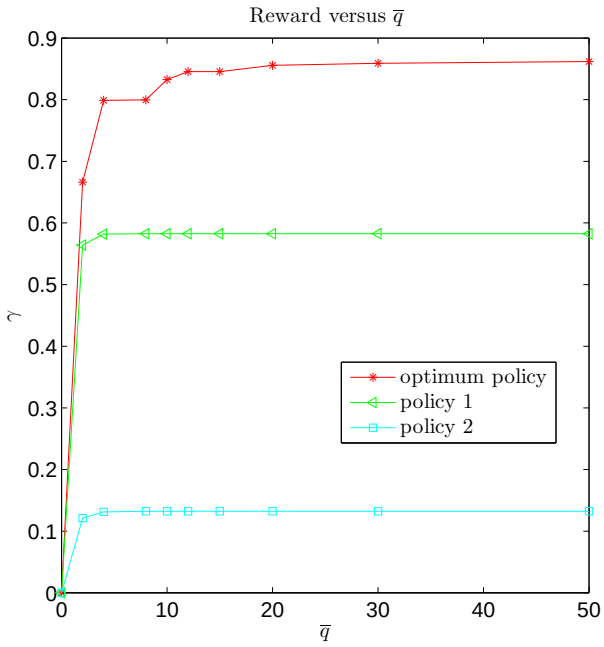


Figure 6.8: Normalized reward γ for different data buffer sizes $\overline{q}$

we can achieve the same best performance even with a smaller energy battery and a smaller data buffer, for example we could set $\overline{b} = 20\delta$ and $\overline{q} = 20\delta_d$.

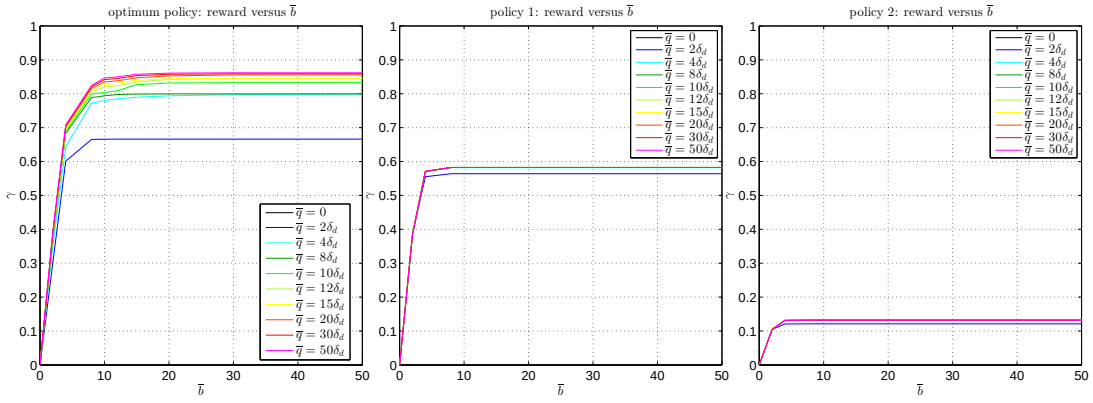


Figure 6.9: Reward for different values of $\overline{b}$ and $\overline{q}$

As introduced in the previous section, we want now to compare the reward performance as a function of the data-sensing efficiency parameter β for different policies. First of all, we describe another empirical policy, that we have implemented for this particular comparison.

Policy 3. The third policy decides to sense data only when the data buffer is empty and transmission isn't possible; while, when there is even only one δ_d-packet, only transmission is admissible, again, choosing the minimum among the data available in the buffer, the data that the system can transmit depending on how much energy there is in the battery and the

maximum of transmitted data. At the end, given the action (t, d) of current slot k, we are then able to compute the state update functions (4.25) and (4.27).

Looking at Fig. 6.10, the reward is shown as the number of δ_d-packets that are transmitted in a time slot. We observe that, increasing the value of β, the reward increases, except for policy 2, because more data are available in the buffer. However, while for small values of β, the first three curves have similar results; for high values of β, our policy achieves the best reward, distancing itself always more from the empirical curves. Moreover, policy 3 is worse than policy 1 because sensing is possible only when data buffer is empty and transmission is limited by data before stored. Regarding the performance of our policy, we can note that, even if the reward increases, increasing β, the best relation sensed data - transmitted data is obtained with $\beta = 1$ Mbit/J, because for one sensed δ_d-packet, about one δ_d-packet is transmitted.

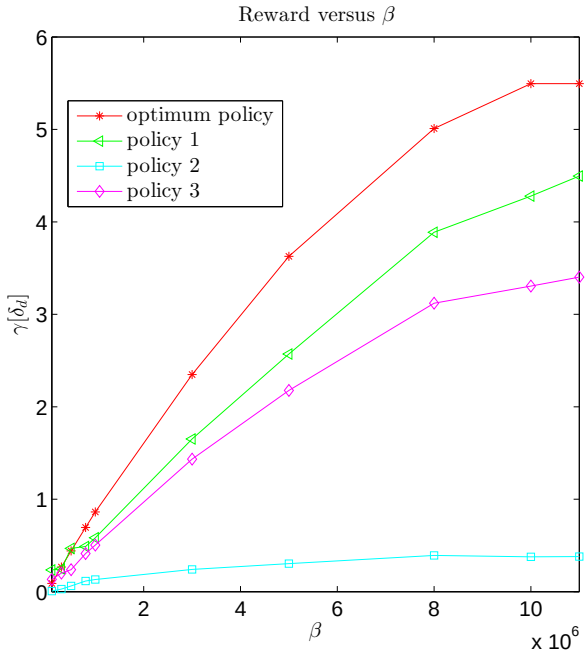


Figure 6.10: Normalized reward γ for different values of β

Chapter 7

Conclusions

In this work, we have studied an EH communication system, that transmits data over an AWGN fading channel over a slotted time. To be more realistic, we have considered a simplified model, in which we have taken into account the energy allocation for sensing data, a constant non-negligible processing energy cost and a constant efficiency parameter. Moreover, the energy arrivals process is a causal stochastic process with a Poisson distribution and the channel gain is modeled by a Rayleigh distribution function.

In this scenario, we have formulated the problem as a MDP and designed a policy to find the optimal action in terms of transmitted and sensed data, respecting all the constraints given by the finite capacity of the energy battery and the data buffer. The aim is to maximize the total transmitted data and, to implement the problem, we have based our algorithm on the value iteration, keeping the computational complexity low. Moreover, we have proved some important theoretical results, as the convexity of the possible actions set, the concavity of the optimal reward and the monotonicity of both of them. These results are useful in order to improve the algorithm and reduce the computational complexity because we are able to look for the optimal action around the previously found optimal action and no more on the entire set of possible actions, but this improvement is left to future works. Solving our problem with a Matlab code, we have plotted the optimal action in terms of transmitted and sensed data and their sum as functions of the energy battery state and the data buffer state for different values of energy arrivals and channel gains, proving the monotonicity of the solutions. Finally, we have choosen and implemented two different empirical policies, in order to compare them with our proposed policy and show the best results, in terms of reward, of this last one.

Appendix A

Proofs

A.1 Proof of relation between e_k and t_k

From the definition of instantaneous rate (2.2), that we report here,

$$t_k = TW \log_2 \left(1 + \frac{\alpha_k^2 \eta e_k}{N_0 W T \Delta_\Gamma}\right);$$

dividing both sides by TW and raising both to the power of 2, we obtain equation (2.3)

$$2^{\frac{t_k}{TW}} = 1 + \frac{\alpha_k^2 \eta e_k}{N_0 W T \Delta_\Gamma}$$

$$e_k = \frac{N_0 W T \Delta_\Gamma}{\alpha_k^2 \eta} \left(2^{\frac{t_k}{TW}} - 1\right),$$

where e_k is a function of t_k and α_k and we can refer to it through

$$e_k = \frac{\tilde{h}(t_k)}{\alpha_k^2}. \tag{A.1}$$

The obtained result means that e_k is the minimum energy required at slot k to transmit t_k bits, knowing that the channel gain is α_k. However this result is valid only if we know the channel gain α_k of slot k, i.e., the ideal case.

On the other hand, in the estimated case, α_k is referred to the channel gain of the previous slot $k-1$. Since, at the beginning of slot k, we don't have the value of α_{k+1}, but only α_k of the previous slot, we must express α_{k+1} in function of α_k through the transition probability $P[\alpha_{k+1}|\alpha_k]$. In order to obtain this expression, we have to rewrite the instantaneous rate as

$$r(e_k, \alpha_{k+1}) = TW \log_2 \left(1 + \frac{\alpha_{k+1}^2 \eta e_k}{N_0 W T \Delta_\Gamma}\right) \tag{A.2}$$

because the channel gain referred to slot k is α_{k+1}, and

$$e_k \geq \frac{\tilde{h}(t_k)}{\alpha_{k+1}^2}, \tag{A.3}$$

with the $\geq$-sign because we need to assure a minimum value of energy to successfully transmit. Then, to find out the energy necessary for data transmission, we condition (A.3) with respect to the known value of α_k of previous slot, i.e.,

$$P\left[e_k \geq \frac{\tilde{h}(t_k)}{\alpha_{k+1}^2}\Big|\alpha_k\right] = P\left[\alpha_{k+1} \geq \sqrt{\frac{\tilde{h}(t_k)}{e_k}}\Big|\alpha_k\right] \geq 1-\epsilon, \tag{A.4}$$

that is the success probability of transmitting a packet, and we want it to be high with respect to a target ϵ. At the same time, the probability of error, i.e. the probability to not transmit the packet, is

$$P\left[\alpha_{k+1} < \sqrt{\frac{\tilde{h}(t_k)}{e_k}}\Big|\alpha_k\right] \leq \epsilon. \tag{A.5}$$

Equivalently, to resolve (A.5), we compute the integral

$$\int_0^{\sqrt{\frac{\tilde{h}(t_k)}{e_k}}} P[\alpha_{k+1} = x|\alpha_k]dx = \epsilon. \tag{A.6}$$

Appendix B

Implementation with Matlab

B.1 Simplified expression of Bellmann's equation

In this section, we show how we can simplify the Bellmann's equation (4.39) to implement it in Matlab.

First of all, we recall the Bellmann's equation for our problem, expressed as

$$g(\mathbf{x}_k) = \max_{\mathbf{a}_k \in \mathcal{A}(\mathbf{x}_k)} \left[\gamma(\mathbf{a}_k, \mathbf{x}_k) + \sum_{\mathbf{x}_{k+1} \in \mathcal{X}} p_{\mathbf{x}_k, \mathbf{x}_{k+1}}(\mathbf{a}_k) g(\mathbf{x}_{k+1}) \right] \qquad \mathbf{x}_k \in \mathcal{X}. \tag{B.1}$$

We can work for a moment with the sum of the right part of (B.1) in order to simplify the expression, that depends on $\mathbf{a}_k$ and $\mathbf{x}_k$. Calling it $U(\mathbf{a}, \mathbf{x})$, where we omit the subscript k, we have

$$\begin{aligned} U(\mathbf{a}, \mathbf{x}) &= \sum_{\mathbf{y} \in \mathcal{X}} p_{\mathbf{x},\mathbf{y}}(\mathbf{a}) g(\mathbf{y}) \\ &= \sum_{[\tilde{b},\tilde{q},\tilde{\alpha},\tilde{h}]} P(\tilde{\alpha}|\alpha) \cdot P(\tilde{h}) \cdot \delta_{\tilde{b}, \underbrace{\min\{b - \tilde{f}(t,d,\alpha) + h, \bar{b}\}}_{m_{\tilde{b}}(b,t,d,\alpha,h)}} \cdot \delta_{\tilde{q}, \underbrace{q - t + d}_{m_{\tilde{q}}(q,t,d)}} \cdot g(\tilde{b}, \tilde{q}, \tilde{\alpha}, \tilde{h}) \\ &= \sum_{[\tilde{b},\tilde{q},\tilde{\alpha},\tilde{h}]} P(\tilde{\alpha}|\alpha) \cdot P(\tilde{h}) \cdot g(m_{\tilde{b}}(b,t,d,\alpha,h), m_{\tilde{q}}(q,t,d), \tilde{\alpha}, \tilde{h}). \end{aligned} \tag{B.2}$$

Defining

$$\bar{g}(\tilde{b}, \tilde{q}, \tilde{\alpha}) = \sum_{\tilde{h}} P(\tilde{h}) g(\tilde{b}, \tilde{q}, \tilde{\alpha}, \tilde{h}), \tag{B.3}$$

where we average the function $g(\tilde{b}, \tilde{q}, \tilde{\alpha}, \tilde{h})$ with the PDF of energy arrivals $\tilde{h}$; and defining

$$\hat{g}(\tilde{b}, \tilde{q}, \alpha) = \sum_{\tilde{\alpha}} P(\tilde{\alpha}|\alpha) \bar{g}(\tilde{b}, \tilde{q}, \tilde{\alpha}), \tag{B.4}$$

where, given a value of the channel gain α at slot k, we take the average again with respect to the transition matrix $P(\tilde{\alpha}|\alpha)$ for all $\tilde{\alpha}$ at slot $k+1$; at the end, we have

$$U(\mathbf{a},\mathbf{x}) = \hat{g}(m_{\tilde{b}}(b,t,d,\alpha,h), q-t+d, \alpha). \tag{B.5}$$

The new functions, defined through the previous steps, are obtained in Matlab by simple inner products between matrices and vectors, so that we are able to reduce the dimensions of matrices. Therefore, the Bellmann's equation becomes

$$g(b,q,\alpha,h) = \max_{(t,d)\in\,\mathcal{A}(\mathbf{x})} \left[t + \hat{g}(m_{\tilde{b}}(b,t,d,\alpha,h), q-t+d, \alpha)\right]. \tag{B.6}$$

Before going on with the resolution of (B.5), we need to define the range of all six variables and the functions that appear inside the cycles, i.e.

$$\begin{aligned}
&P(\tilde{h}) = \frac{\lambda_h^{\tilde{h}} \cdot e^{-\lambda_h}}{\tilde{h}!};\\
&\tilde{f}(t,d,\alpha) = \frac{N_0 W T \Delta_\Gamma}{(\eta\alpha)^2}\left(2^{\frac{t}{WT}} - 1\right) + \frac{d}{\beta\eta} + \xi 1(t+d);\\
&m_{\tilde{b}}(b,t,d,\alpha,h) = \min\{b - \tilde{f}(t,d,\alpha) + h, \overline{b}\} = \tilde{b};\\
&m_{\tilde{q}}(q,t,d) = q - t + d = \tilde{q}.
\end{aligned}$$

We start with the initial 6-dimensional matrix $U(\mathbf{a},\mathbf{x})$ of type $\frac{\overline{t}}{\delta_d} \times \frac{\overline{d}}{\delta_d} \times \frac{\overline{b}}{\delta} \times \frac{\overline{q}}{\delta_d} \times \frac{\lceil \max(\alpha) \rceil}{\text{step}} \times \frac{h_{max}-h_{min}}{\delta}$, where step is the quantization step used in order to find the transition matrix that characterizes the channel. Then we are able to compute $U(\mathbf{a},\mathbf{x})$ with the following steps:

1. first, we calculate $\bar{g}(\tilde{b},\tilde{q},\tilde{\alpha})$ through the inner product between the PDF of energy arrival $\tilde{h}$, for all possible values of $\tilde{h}$, and the 4-dimensional matrix $g(m_{\tilde{b}}, m_{\tilde{q}}, \tilde{\alpha}, \tilde{h})$; the result is a 3-dimensional matrix of type $\frac{\overline{b}}{\delta} \times \frac{\overline{q}}{\delta_d} \times \frac{\lceil \max(\alpha) \rceil}{\text{step}}$;

2. we simplify again, taking the inner product between a row of the transition matrix $P[\tilde{\alpha}|\alpha]$, corresponding to a given value for α at instant k, and the matrix $\bar{g}(\tilde{b},\tilde{q},\tilde{\alpha})$ along the third dimension; we obtain a new 3-dimensional matrix $\hat{g}(\tilde{b},\tilde{q},\alpha)$ of type $\frac{\overline{b}}{\delta} \times \frac{\overline{q}}{\delta_d} \times \frac{\lceil \max(\alpha) \rceil}{\text{step}}$, for all possible values α of slot k;

Continuing the algorithm from that point, we compute the Bellmann's equation (B.1):

1. we sum each value t with the corresponding element of matrix $\hat{g}(\tilde{b},\tilde{q},\alpha)$, obtained by the two functions for $\tilde{b}$ and $\tilde{q}$, that depend on that specific value of t;

2. we take the maximum with respect to all admissible pairs (t,d) (the admissibility is defined at the beginning of the cycles through the condition (4.26));

3. Obtained the maximum, we write it in the corresponding cell of 4-dimensional matrix $g(b, q, \alpha, h)$, of type $\frac{\bar{b}}{\delta} \times \frac{\bar{q}}{\delta_d} \times \frac{\lceil \max(\alpha) \rceil}{\text{step}} \times \frac{h_{max} - h_{min}}{\delta}$.

These steps characterize a single instant k, and we must repeat all the calculations for a long time interval.

B.2 Matlab code

B.2.1 The main code

```
function main_C()
slot=1000
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% main parameters

    T = 1e-2;                 % slot duration [s]
    W = 5e6;                  % bandwidth [Hz]
    TW = T*W;                 % constant
    N_0 = 1e-8;               % power spectral density [W/Hz]
    eta = 0.9;                % efficiency of the solar panel
    GammaL = 1;               % signal-to-noise ratio gap
    K = N_0*TW*GammaL/eta;    % constant

    delta = 0.0036;           % unit of energy [J]
    b_max = 50*delta;         % [J] maximum energy buffer capacity
    xi = delta;               % [J] dissipated energy
    h_max = 10*delta;         % [J] maximum energy at input
    lambda_h = 1;             % number of delta units of power

    delta_d = 3600;           % unit of data [bit]
    q_max = 50*delta_d;       % [bit] maximum data buffer size
    t_max = 30*delta_d;       % [bit] maximum transmitted bit
    d_max = 11*delta_d;       % [bit] maximum data for sensing
    beta = 1e6;               % [bit/J] fraction for data sensing

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% initialization

    b_int = floor(b_max/delta)+1;   % # of possible values of b
    h_int = floor(h_max/delta)+1;   % # of possible values of h
    q_int = floor(q_max/delta_d)+1; % # of possible values of q
    t_int = floor(t_max/delta_d)+1; % # of possible values of t
    d_int = floor(d_max/delta_d)+1; % # of possible values of d
    a_int = length(a);              % # of possible values of a

    b(:,1) = (0:b_int-1)*delta;     % vector of all possible values of b
    h(:,1) = (0:h_int-1)*delta;     % vector of all possible values of energy arrivals
    q(:,1) = (0:q_int-1)*delta_d;   % vector of possible values of q
    t(:,1) = (0:t_int-1)*delta_d;   % vector of all possible values of t
    d(:,1) = (0:d_int-1)*delta_d;   % vector of all possible values of d
    a      = a';                    % vector of all possible values of a
```

```

    % energy arrival probabilities
    P_h(:,1) = exp(-lambda_h)*lambda_h.^(0:h_int-1)./(factorial(0:h_int-1));
    P_h(end) = 1-sum(P_h(1:end-1));

    % channel transition probababilities (already taken from 'channel' file)
    p_kk1_alpha;

    % state update function f()
    [a1 b1 c1] = ndgrid(2.^(t/TW)-1,d/(beta*eta)+xi,K./(eta*a.^2));
    f_tilde = c1.*a1+b1;
    f_tilde(1,:,1) = b1(1,:,1); % when t=a=0 --> simplified expression
    f_tilde(1,1,:) = 0;          % when t=d=0 --> no transmission

    % g_x function
    g_x = zeros(b_int, q_int, a_int, h_int);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% main loop

for k = 1:slot
    tic
    % find g_bar
    tmp = reshape(g_x,b_int*q_int*a_int,h_int)*P_h;
    tmp = reshape(tmp,b_int,q_int,a_int);
    % find g_hat
    tmp = reshape(tmp,b_int*q_int,a_int)*p_kk1_alpha';
    tmp = reshape(tmp,b_int,q_int,a_int);
    % compute gx
    [tmp, action] = compute_gx(b,q,h,a,t,d,delta,delta_d,f_tilde,tmp,beta*eta,xi);
    tmp = tmp-tmp(1);
    diffe = norm(g_x(:)-tmp(:));
    g_x = tmp;
    % display
    disp(['End slot ', num2str(k), ', diff = ', num2str(diffe)])
    toc
    % exit if small diff
    if (diffe<1e-40)
        break;
    end
    % save once in a while
end
```

B.2.2 Function to compute the optimal action

```
#include "math.h"
#include "mex.h"
#include "matrix.h"
#include <stdlib.h>
#include <float.h>
```

```
int round(double number)
{
    return (number >= 0) ? (int)(number + 0.5) : (int)(number - 0.5);
}

/*****************************************************************
 * Main function
 *****************************************************************/
void mexFunction(
        int nlhs,               // number of outputs
        mxArray *plhs[],        // outputs vector
        int nrhs,               // number of inputs
        const mxArray *prhs[]   // inputs vector
        )
{
    double *b, *q, *h, *a, *t, *d, *f_tilde, *g_hat, *g_x, *action;
    double delta, delta_d, etabeta, xi, h_tk, q_t, y, max_v;
    double vb, vq, vt, vh, va, vd, q_max;
    int b_int, q_int, h_int, a_int, t_int, d_int, t_maxint, d_maxint;
    int max_t, max_d, dims[5], i, l, bk, qk, hk, ak, tk, dk, bk1, qk1,ind;

    /* 1. Check validity of expressions */
    // check input length
    if (nrhs !=12)
        mexErrMsgTxt("Twelve input arguments required");
    // check output length
    if (nlhs != 2)
        mexErrMsgTxt("Two output arguments required");

    /* 2. Read inputs */
    // input array
    b = mxGetPr(prhs[0]);
    q = mxGetPr(prhs[1]);
    h = mxGetPr(prhs[2]);
    a = mxGetPr(prhs[3]);
    t = mxGetPr(prhs[4]);
    d = mxGetPr(prhs[5]);
    // vector length
    b_int    = mxGetM(prhs[0]);
    q_int    = mxGetM(prhs[1]);
    h_int    = mxGetM(prhs[2]);
    a_int    = mxGetM(prhs[3]);
    t_maxint = mxGetM(prhs[4]);
    d_maxint = mxGetM(prhs[5]);
    // maximum values
    q_max = q[q_int-1];
    //units of energy and data
    delta   =  mxGetScalar(prhs[6]);
    delta_d =  mxGetScalar(prhs[7]);
    //auxiliary matrices
    f_tilde = mxGetPr(prhs[8]);
    g_hat   = mxGetPr(prhs[9]);
```

```
    //input constant
60  etabeta = mxGetScalar(prhs[10]);
    xi      = mxGetScalar(prhs[11]);

    /* 3. Prepare output vectors */
    dims[1] = b_int;
65  dims[2] = q_int;
    dims[3] = a_int;
    dims[4] = h_int;
    dims[0] = 2;
    plhs[0] = mxCreateNumericArray(4,&dims[1],mxDOUBLE_CLASS, mxREAL);
70  g_x     = mxGetPr(plhs[0]);
    plhs[1] = mxCreateNumericArray(5,dims,mxDOUBLE_CLASS, mxREAL);
    action  = mxGetPr(plhs[1]);

    /* 4. Run the algorithm */
75  for (bk=0; bk<b_int; bk++)
    {
        vb = b[bk];
        for (qk=0;qk<q_int; qk++)
        {
80          vq = q[qk];
            for (ak=0; ak<a_int; ak++)
            {
                va = a[ak];
                for (hk=0; hk<h_int; hk++)
85              {
                    vh = h[hk];
                    // prepare value of the zero action
                    max_v = g_hat[min(bk+hk,b_int-1)+b_int*(qk+q_int*ak)];
                    max_t = 0;
90                  max_d = 0;
                    // search for maximum
                    if (vb>xi) {
                        t_int = (ak==0?1:min(qk+1,t_maxint));
                        for (tk=0; tk<t_int; tk++)
95                      {
                            vt = t[tk];
                            d_int = round(etabeta*(double)(vb-f_tilde[tk+t_maxint*(0+
                              d_maxint*ak)]))/delta_d+1;
                            d_int = min(min(q_int-qk+tk,d_int),d_maxint);
                            for (dk=0; dk<d_int; dk++)
100                         {
                                vd = d[dk];
                                // search for maximum
                                bk1 = bk+hk-round((f_tilde[tk+t_maxint*(dk+d_maxint*ak)]/
                                  delta));
                                bk1 = min(max(0,bk1),b_int-1);
105                             qk1 = qk-tk+dk;
                                y = vt + g_hat[bk1+b_int*(qk1+q_int*ak)];
                                if (y>=max_v) {
                                    max_v = y;
```

```
                                    max_t = vt;
                                    max_d = vd;
                                }
                            }
                        }
                    }
                    // store maximum
                    ind = bk+b_int*(qk+q_int*(ak+a_int*hk));
                    g_x[ind] = max_v;
                    action[0+2*ind] = max_t;
                    action[1+2*ind] = max_d;
                }
            }
        }
    }
    /* 5. Exit */
    return;
}
```

B.2.3 The empirical policies code

```
time = 1e6;
%%%%%%%%%%%%%%%%%% policy1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial values
b_k = 0;
q_k = 0;
%known values of channel gain and energy arrivals
h_k = arrival;

%to find the linear values of channel gain
alpha_db = (qa-12)*3; %in dB
alpha= 10.^((alpha_db+10)/20); %linear
alpha(alpha<0.1) = 0;

for k=1:time
   %current state
    bk  = round(b_k/delta) +1;    %index of current battery state
    qk  = round(q_k/delta_d) +1;  %index of current data buffer state
    ak  = qa(1,k);                %index of current channel gain
    a_k = alpha(1,k);             %linear value of channel gain
    hk  = h_k(1,k) +1;
    %current action
    en_avail   = b_k-xi; %available energy
    data_avail = q_k;    %available data to transmit
    if en_avail>0        %if enough energy in battery
        if data_avail>0  %if there are data in buffer
            t_k = min(data_avail,min(TW*log2(1+(a_k^2*(en_avail)/K)),t_max));
            e_k = K/(a_k)^2*(2^(t_k/TW)-1); %really used energy for transmission
            s_k = max(en_avail-e_k,0);      %left energy for sensing
            d_k = min(beta*s_k, min(q_max-data_avail+t_k,d_max));
        else                     %if no data in buffer
```

```
            t_k = 0;
            d_k = min(en_avail*beta, d_max); %only sensing
        end
    else                                    %if no enough energy in battery
        t_k = 0;
        d_k = 0;
    end
    tk = round(t_k/delta_d) +1;   %index of current action t
    dk = round(d_k/delta_d) +1;   %index of current action d

    %following state and parameters
    f_tilde_k = f_tilde(tk,dk,ak);
    b_k = compute_btilde_val(b_k, f_tilde_k, (hk-1)*delta,b_max,delta);
    q_k = max(data_avail- t_k+ d_k,0);
    tx(1,k) = t_k; %transmitted data at slot k
end

re_1 = sum(tx(:))/(time*delta_d) %normalized reward

%%%%%%%%%%%%%%%%%%% policy2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial index of state
bk = 1;
qk = 1;
for k = 1:time
    ak = qa(k);   %index of current channel gain
    hk = h_k(k);  %index of current energy arrivals
    pk = round(f_tilde(1:min(t_int,qk),1,ak)/delta)+1;
    [~,tk] = max(pk.*(pk<=bk)); % transmitted bit
    pk = round(f_tilde(tk,1:min(d_int,q_int-qk+1),ak)/delta)+1;
    [~,dk] = max(pk.*(pk<=bk)); % sensed bit
    pk = pk(dk); % transmitted power
    % update state --> following indices of state
    bk = min(max(1,bk-pk+hk),b_int);
    qk = qk-tk+dk;
    % memory
    re(k) = tk-1;
end

re_2 = mean(re)          %normalized reward

%%%%%%%%%%%%%%%%%%% policy3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial values
b_k = 0;
q_k = 0;

for k=1:time
   %current state
    bk  = round(b_k/delta) +1;   %index of current battery state
    qk  = round(q_k/delta_d) +1; %index of current data buffer state
    ak  = qa(1,k);                    %index of current channel gain
    a_k = alpha(1,k);                 %linear value of channel
    hk  = h_k(1,k) +1;                %index of current energy arrivals
```

```
    %current action
    en_avail    = b_k-xi; %available energy
    data_avail = q_k;      %available data to transmit
    if en_avail>0          %if enough energy in battery
        if data_avail>0    %if data in buffer
            t_k = min(data_avail,min(TW*log2(1+(a_k^2*(en_avail)/K)),t_max));
            d_k = 0; %no sensing
        else
            t_k = 0;       %no transmission
            d_k = min(en_avail*beta, d_max);
        end
    else
        t_k = 0;
        d_k = 0;
    end
    tk = round(t_k/delta_d) +1;
    dk = round(d_k/delta_d) +1;

    %following state and parameters
    f_tilde_k = f_tilde(tk,dk,ak);
    b_k = compute_btilde_val(b_k, f_tilde_k, (hk-1)*delta,b_max,delta);
    q_k = max(data_avail- t_k+ d_k,0);
    tx(1,k) = t_k; %transmitted data at slot k
end

re_3 = sum(tx(:))/(time*delta_d) %normalized reward
```

Bibliography

[1] R. Rajesh, V. Sharma, and P. Viswanath, "Information capacity of energy harvesting sensor nodes," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on* (IEEE, ed.), pp. 2363–2367, 2011.

[2] R. Rajesh, V. Sharma, and P. Viswanath, "Capacity of gaussian channels with energy harvesting and processing cost," *Information Theory, IEEE Transactions on*, vol. 60, pp. 2563–2575, May 2014.

[3] S. Mao, M. H. Cheung, and V. W. Wong, "Joint energy allocation for sensing and transmission in rechargeable wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, 2014.

[4] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus, and A. Yener, "Transmission with energy harvesting nodes in fading wireless channels: Optimal policies," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 8, pp. 1732–1743, 2011.

[5] O. Orhan, D. Gunduz, and E. Erkip, "Optimal packet scheduling for an energy harvesting transmitter with processing cost," in *Communications (ICC), 2013 IEEE International Conference on*, pp. 3110–3114, IEEE, 2013.

[6] J. Xu and R. Zhang, "Throughput optimal policies for energy harvesting wireless transmitters with non-ideal circuit power," *Selected Areas in Communications, IEEE Journal on*, vol. 32, pp. 322–332, 2014.

[7] O. Ozel, K. Shahzad, and S. Ulukus, "Energy harvesting communications with hybrid energy storage and processing cost," in *Signals, Systems and Computers, 2013 Asilomar Conference on*, pp. 609–613, Nov 2013.

[8] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-constrained modulation optimization," *Wireless Communications, IEEE Transactions on*, vol. 4, no. 5, pp. 2349–2360, 2005.

[9] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena Scientific Belmont, MA, 1995.

[10] O. Orhan, D. Gunduz, and E. Erkip, "Throughput maximization for an energy harvesting communication system with processing cost," in *Information Theory Workshop (ITW), 2012 IEEE*, pp. 84–88, IEEE, 2012.

[11] S. Mao, M. H. Cheung, and V. W. Wong, "An optimal energy allocation algorithm for energy harvesting wireless sensor networks," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 265–270, IEEE, 2012.

[12] N. Bui and M. Rossi, "Staying alive: System design for self-sufficient sensor networks," *arXiv preprint arXiv:1310.7717*, 2013.

[13] M. Miozzo, D. Zordan, P. Dini, and M. Rossi, "Solarstat: Modeling photovoltaic sources through stochastic markov processes," *arXiv preprint arXiv:1311.1025*, 2013.

[14] A. Fu, E. Modiano, and J. N. Tsitsiklis, "Optimal transmission scheduling over a fading channel with energy and deadline constraints," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 3, pp. 630–641, 2006.

[15] X. Kang, Y.-K. Chia, C. K. Ho, and S. Sun, "Cost minimization for fading channels with energy harvesting and conventional energy," *Wireless Communications, IEEE Transactions on*, vol. 13, pp. 4586–4598, Aug 2014.

[16] S. Luo, R. Zhang, and T. J. Lim, "Optimal save-then-transmit protocol for energy harvesting wireless transmitters," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 3, pp. 1196–1207, 2013.

[17] O. Orhan, D. Gunduz, and E. Erkip, "Energy harvesting broadband communication systems with processing energy cost," *arXiv preprint arXiv:1312.0054*, 2013.

[18] P. Youssef-Massaad, L. Zheng, and M. Médard, "Bursty transmission and glue pouring: on wireless channels with overhead costs," *Wireless Communications, IEEE Transactions on*, vol. 7, no. 12, pp. 5188–5194, 2008.

[19] T. Erseghe, "On the evaluation of the polyanskiy-poor-verdu converse bound for finite blocklength coding in awgn," *arXiv preprint arXiv:1401.7169*, 2014.

[20] S. M. Ross, *Applied Probability Models with Optimization Applications*. Holden Day, San Francisco, 1970.

[21] T. Erseghe, A. Zanella, and C. G. Codemo, "Markov decision processes with threshold based piecewise linear optimal policies," *Wireless Communications Letters, IEEE*, vol. 2, no. 4, pp. 459–462, 2013.

[22] "http://www.alameteo.it/webcam.html," 2014.

[23] "http://www.arpa.veneto.it/arpavinforma/bollettini/meteo-e-neve-1/webcam/informazioni," 2014.