# A Framework for Optimal Battery Management for Wireless Nodes

Maria Adamou[1] and Saswati Sarkar[2],
[1]Department of Computer and Information Science,
[2]Department of Electrical Engineering,
University of Pennsylvania Philadelphia, PA 19104
{adamou@gradient.cis,swati@ee}.upenn.edu

August 14, 2001

**Abstract**

The focus of this paper is to extend the lifetime of a battery powered node in wireless context. The lifetime of a battery depends on both the manner of discharge and the transmission power requirements. We present a framework for computing the optimal discharge strategy which maximizes the lifetime of a node by exploiting the battery characteristics and adapting to the varying power requirements for wireless operations. The complexity of the optimal computation is linear in the number of system states. However, since the number of states can be large, the optimal strategy can only be computed offline and executed via a table-lookup. We also present a simple discharge strategy which can be executed online without any table lookup, and attains near maximum battery lifetime. Finally, we use state space reduction techniques to approximate the optimal computation in significantly lower complexity.

## 1 Introduction

One major issue in today's wireless networks is the limited energy of the communication devices. Wireless networks consist of small portable devices, such as PDAs, mobile phones, headsets, etc, which have limited processing power and battery energy. Thus, one of the most important challenges in the design of wireless networks is to provide power management techniques which are low cost and computationally simple.

A lot of research has been performed in this area, primarily aiming to reduce the energy consumption at the hardware level [8], and at different layers of the network stack [7]. Another aspect of power management is to increase the lifetime of the battery of a mobile node by using energy efficient battery management techniques. For example, as shown in [3], using a pulsed current discharge instead of a constant discharge can result in an improved battery performance. The energy efficiency of a battery can be further improved by using an efficient pulsed discharged technique. A battery consists of several electrochemical cells from which power needs to be drained when the node transmits a packet. When a cell is allowed to rest in between discharge periods, it is able to recover part of its charge, thanks to the diffusion mechanism [3], thus the total energy delivered is increased. A battery discharge policy decides which cells should serve the packet and which cells are allowed to rest. Analytical and simulation results in [1],[2] show that the discharge policies have significant impact on the battery lifetime. In addition, [2] states that it is possible to implement many different discharge strategies, using smart battery packages [9]. Our goal is to find an optimal battery management policy that maximizes the delivered energy by exploiting the recovery capability of the battery, and adapting to the varying power requirements of wireless transmissions.

In [1], a stochastic model of the battery, which takes into consideration the recovery capability of the cells, is introduced. The model is used to show the improvements in the capacity of the battery of a node when pulsed discharge is considered. Also, the maximum energy a single cell can deliver is computed, under different traffic arrival processes. In [2], simple scheduling algorithms for discharging the cells of a battery are applied, combined with traffic shaping techniques, and their performance is evaluated by analysis. Using the model presented in [1], the authors show analytically that the Round Robin scheduling scheme can significantly improve the battery lifetime, as the battery is able to deliver its maximum available energy. The authors mention that an optimal discharge policy for attaining the maximum energy from a battery is an interesting open problem in the wireless context.

In this paper, we investigate an optimal battery management strategy. The contribution of the paper can be summarized as follows. Firstly, we develop a methodology for obtaining the optimal policy for discharging the cells of a battery using stochastic dynamic programming. The cells are optimally scheduled to serve the packets and the recovery process is fully exploited. In general, the formalization of such systems can be very complex and involves the solution of a large number of linear equations. The overall complexity

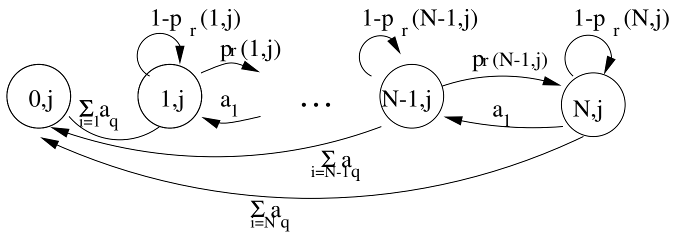


Figure 1: Stochastic model of a battery cell

is $O(M^4)$, where $M$ is the size of the state space of the system and this size is usually very large for real systems. Using the special properties of our system we develop a linear complexity ($O(M)$) algorithm for computing the optimal. By applying this algorithm, the optimal policy can be computed offline executed using table-lookup. Furthermore, the knowledge of the optimal can be used to evaluate the performance of online scheduling policies. Secondly, we propose a simple online scheduling policy which can be used without any table-lookup. We show analytically and by simulation that our simple policy performs close to the optimal and considerably improves over the Round Robin policy proposed in [2]. The improvement is around 20% in general and in some cases even higher. Thirdly, since the size of the state space $M$ of the system can be large for real scenarios, the computation of the optimal may still be complex, even though it is linear in $M$. Hence, we also propose an approximation strategy for computing the optimal policy for larger systems and we show that it gives results close to the optimal while having significantly less complexity.

The rest of the paper is organized as follows. In Section 2 we describe the battery model and the discharge procedure. In Section 3 we present the general framework for computing the optimal battery management policy. In Section 4 we give our linear complexity computation technique. In Section 5 we describe the proposed suboptimal policy and how it can be evaluated using the general framework. We also evaluate its performance by analysis and simulation, for different values of battery capacity and traffic models. In Section 6 we describe our approximation technique and compare its performance with the optimal. Finally, our conclusion and future work are discussed in Section 7.

## 2 System Assumptions and Objectives

We are focusing on the lifetime of a single battery-powered node. In this section, we describe the battery model and the discharge procedure. Our model is similar to that of [2]. A battery is defined as a group of $L$ electrochemical cells electrically connected in a serial and/or parallel arrangement. The "theoretical capacity" ($C$) of a cell is the maximum energy it can deliver. A cell can deliver $C$ units only if all the available active material of the cell is used. The "nominal capacity" of a cell ($N$) is the total energy it can deliver under a constant current discharge. When a packet needs to be transmitted by the mobile node, a certain number of charge units need to be discharged from the battery, from one or more of its cells. When a cell is not being drained it can recover one charge unit with a certain probability, due to the diffusion process [5]. As a result, the actual energy delivered by a cell, during its lifetime, is between $N$ and $C$ charge units. It delivers $N$ charge units if it does not recover any charge, while it delivers $C$ units under maximum possible recovery. The problem we investigate is how to efficiently assign the packets to the cells. The objective is to optimize the charge recovery process and thus maximize the total energy delivered by all the cells. This in turn maximizes the battery lifetime.

We assume that a cell is modeled by a stochastic process with $N+1$ states $(x_0 \dots x_N)$, as shown in Figure 1. For each state of the cell $i$, denoted by $(n_i, c_i)$, we define $n_i$ to be the remaining charge and $c_i$ the remaining capacity left in the cell. In other words, $c_i$ is the difference between the maximum theoretical capacity $C$ and the total charge units discharged so far from the cell. Initially, each cell is fully charged with $N$ charge units, and the remaining capacity equals $C$. Thus the initial state of a cell is $(N, C)$.

The "size" of a packet is the number of charge units required to transmit the packet, and the required charge depends on the power level of the transmission. This in turn depends on the transmission conditions and the distance of the destination. For example, when the transmission conditions are poor, or the next hop node is farther away, the packet needs to be transmitted at higher power, and as such it has a larger "size." We consider slotted time. We assume that a node needs to transmit at most one packet each time slot. The "size" of such a packet is $q$, $q \geq 0$, with probability $a_q$. If $q = 0$, then there is no packet to be transmitted in the slot. We assume that $a_0 < 1$. We also assume that the time needed for serving a packet is one time slot.

When a cell $i$ in state $(n_i, c_i)$ delivers $q$ units of charge, it moves to the state $(n_i - q,\ c_i - q)$. A cell $i$ is fully discharged (or "inactive") when its charge or its capacity becomes zero ($n_i = 0$, or $q_i = 0$). A battery expires when all its cells are completely discharged. A cell that is in state $(n_i, c_i)$, $0 < n_i < N, 0 < c_i < C$, in a slot, and is not serving a packet, may recover one charge unit and move to the state $(n_i + 1, c_i)$ with probability $p_r(n_i, c_i)$, where

$$p_r(n_i, c_i) = \begin{cases} e^{-g(N-n_i)-\phi(c_i)}, & 1 \leq n_i < N, 1 \leq c_i < C \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In this equation $g$ is a constant that depends on the discharge process of the cell and $\phi(j)$ is a step function which decreases as the remaining capacity of the cell increases. For example, if $C = 200$, $\phi(j) = 15.6$, for $0 \leq j \leq 5$, $\phi(j) = 0.8$, for $5 \leq j < 100$,$\phi(j) = 0.0025$, for $100 \leq j < 195$. Equation (1) denotes that the recovery capability of a cell decreases exponentially as more charge units are drained from the cell and the remaining charge and the capacity decrease.

The battery discharge policy decides which cells should be drained to serve an incoming packet. We consider only work-conserving discharge policies, which always serve an incoming packet, as long as there is an active cell in the system. A packet can be served by one or several cells. The authors in [2] showed that the lifetime of the battery significantly improves if each packet is served by only one cell, while the other cells recover. Thus, we assume that for each packet transmission the necessary current is drained from just one cell. The discharge policy considerably affects the total number of packets that can be transmitted during the node's lifetime. Consider a battery with only two cells and a packet of size 1 arriving in every time slot. A possible discharge policy is to assign all the packets to one cell until it is completely discharged and then use the second cell. In this case, only a total of $2N$ packets will be transmitted before the battery expires since the cells do not recover any charge, while the maximum limit is $2C$. On the other hand, a policy which uses the cell that has the larger remaining charge allows both cells to recover charges, and thus the total energy delivered will be close to $2C$ in this case. Intuitively, an efficient battery management policy should take into account the recovery probability of each cell, which depends on both its remaining charge and capacity, as Equation (1) shows, so as to fully exploit the recovery mechanism of the battery. Our objective is to provide an optimal policy that efficiently selects a cell for an incoming packet so as to maximize the total energy delivered by the battery before all the cells are completely discharged.

## 3 A Framework for Optimal Battery Management

We will present a framework for computing the optimal battery management policy using the theory of Markov decision processes (MDP)[4]. More specifically, we will use the theory of "stochastic shortest path" problem, presented in [4]. We give an overview of the related theory and computational techniques in the appendix. In this paper, we show that the optimal battery management problem falls within the purview of the stochastic shortest path problem and as such the general optimal policy formalization for stochastic shortest paths holds in this case as well.

### 3.1 Mathematical Formulation of System Evolution

We will use the system descriptions and assumptions introduced in Section 2 here. We represent the state of the system at time $k$ as a $2L$-tuple $x_k = (n_{1k}, c_{1k}, \ldots, n_{Lk}, c_{Lk})$, where $n_{uk} = 0, \ldots, N, c_{uk} = 0, \ldots, C$ are the remaining charge and capacity for cell $u$, at time $k$. Then, the system has a total of $M$ possible states, where $M = ((N+1)(C+1))^L$. The initial state is $x_0 = (N, C, \ldots, N, C)$. At each time $k$ the system chooses a cell to serve a packet of size $q_k$, where $Pr(q_k = i) = a_i$ for all slots $k$ and nonnegative integers $i$ and $q_1, q_2, \ldots$ are mutually independent. The cell chosen in slot $k$ is $u_k$, and $u_k$ belongs to the set of active cells, $U_k \subset \{1, \ldots, L\}$. A battery management policy is a rule which in every slot $k$ chooses the cell for serving a packet as a function of the system state $x_k$. The next state $x_{k+1}$ of the system depends on the size of the packet $q_k$, the chosen cell $u_k$ and the recovery process for all the cells. The amount of charge recovered by cell $r$ in slot $k$ is $a_{rk}$. Note that $a_{rk}$ can be either 0 or 1, $a_{rk} = 0$ if cell $r$ serves a packet in slot $k$, otherwise $a_{rk}$ is 1 or 0 depending on whether or not the cell recovers a charge unit. The transition probability $p_{xy}(u)$ from state $x$ to state $y$ under cell selection $u$ depends on the recovery probability $p_r(n_i, c_i)$ defined in Equation (1) for every cell $i$, and the probability distribution for the size of the packet $\{a_q\}$. We introduce some notations for describing the transition probabilities $p_{xy}(u)$. Let $x = (n_{1x}, c_{1x}, \ldots, n_{ux}, c_{ux}, \ldots, n_{Lx}, c_{Lx})$, $y = (n_{1y}, c_{1y}, \ldots, n_{uy}, c_{uy}, \ldots, n_{Ly}, c_{Ly})$. Let $n_{iy} \in \{n_{ix}, n_{ix}+1\}$ and $c_{iy} = c_{ix}, i \neq u$. Let $A_{xy}(u) = \{i : i \neq u, n_{iy} = n_{ix}+1\}$ and $B_{xy}(u) = \{i : i \neq u, n_{iy} = n_{ix}\}$. Thus $A_{xy}(u)$ is the set of cells which recover charge, and $B_{xy}(u)$ is the set of cells which do not recover charge. For example, consider a 3-cell system and $x = (1, 2, 2, 3, 4, 5)$, $y = (2, 2, 2, 3, 4, 5)$ and $u = 2$. Then,

$A_{xy} = \{1\}$ and $B_{xy} = \{3\}$.

$$p_{xy}(u) = \begin{cases} a_l \Pi_{i \in A_{xy}(u)} p_r\,(n_{ix}, c_{ix}) \\ \Pi_{i \in B_{xy}(u)}\,(1 - p_r(n_{ix}, c_{ix})) \\ \qquad \text{if } n_{uy} = n_{ux} - l, c_{uy} = c_{ux} - l, l > 0, \\ \qquad n_{uy} > 0, c_{uy} > 0 \qquad\qquad (a) \\ a_0\,(1 - p_r(n_{ux}, c_{ux}))\,\Pi_{i \in A_{xy}(u)} p_r\,(n_{ix}, c_{ix}) \\ \Pi_{i \in B_{xy}(u)}\,(1 - p_r(n_{ix}, c_{ix})) \\ \qquad \text{if } n_{ux} = n_{uy}, c_{ux} = c_{uy} \qquad\qquad (b) \\ a_0 p_r(n_{ux}, c_{ux}) \Pi_{i \in A_{xy}(u)} p_r\,(n_{ix}, c_{ix}) \\ \Pi_{i \in B_{xy}(u)}\,(1 - p_r(n_{ix}, c_{ix})) \\ \qquad \text{if } n_{uy} = n_{ux} + 1, c_{ux} = c_{uy}, \qquad (c) \end{cases} \tag{2}$$

Also, $p_{xy}(u) = 0$, for any other state $y$. The transition probabilities can be explained as follows. Since cell $u$ is selected for discharging, $n_{uy} = n_{ux} - l$, $c_{uy} = c_{ux} - l$ if a packet of size $l$ arrives (with probability $a_l$). Any other cell $i$ does not lose any remaining charge or capacity, (thus $c_{iy} = c_{ix}$) and may or may not recover. In the first case, $n_{iy} = n_{ix} + 1$, (w.p. $p_r(n_{ix}, n_{iy})$) and in the latter case $n_{iy} = n_{ix}$(w.p. $1 - p_r(n_{ix}, n_{iy})$) (case (a)). However, if no packet arrives (w.p. $a_0$), then cell $u$ can also recover and the recovery event is similar to that of the others (cases (b) and (c)).

The energy delivered by the battery at time $k$ is equal to the minimum of the packet size and the remaining charge and capacity of the scheduled cell. Thus, the average energy delivered in state $x$ under cell selection $u$, $\bar{q}(x, u)$ is given by:

$$\bar{q}(x, u) = \sum_{i=1}^{\min(n_u, c_u)} i a_i + \min(n_u, c_u) \left(1 - \sum_{i=1}^{\min(n_u, c_u)} a_i\right) \tag{3}$$

The objective is to choose the cell at each slot such that the expected cumulative energy is maximized. The choice of the cell depends on the state of the system. Let $J^*(x)$ denote the optimal energy if the system starts from state $x$. The objective is to compute the optimal cell selection $\mu^*$ which attains the energy $J^*(x)$ for each $x$. We first illustrate the state evolution with an example. In case of a two cell battery, we can represent the state of the system as $x_k = (n_1, c_1, n_2, c_2)$. Assume that a packet of size $q_k > 0$ arrives and the first cell is chosen to serve this packet ($u_k = 1$). Thus, the state $x_{k+1}$ will be given by the following relation: $x_{k+1} = (n_1 - q_k, c_1 - q_k, n_2 + a_{2k}, c_2)$. If $u_k = 2$, the next state is $x_{k+1} = (n_1 + a_{1k}, c_1, n_2 - q_k, c_2 - q_k)$. The energy obtained equals $q_k$ in either case.

## 3.2 Justification for using Stochastic Shortest Path Problem

We will now justify that the optimal battery management problem falls within the purview of the stochastic shortest path problem. The first observation is that *the total number of possible system states $M$ is finite.* Next, *given the current state and the cell selection, the future state is independent of the past states and cell selections.* This follows from the system evolution and the fact that the packet sizes are independent from slot to slot. The system terminates when the battery expires, and this happens when all the cells are fully discharged. Recall that a cell is fully discharged if the remaining charge or remaining capacity is 0. Thus any state $(n_1, c_1, \ldots, n_L, c_L)$ where either $n_i = 0$ or $c_i = 0$ for each $i$ is a termination state. Let $T$ denote the set of termination states. Note that once the battery reaches a state $x \in T$, it remains there and can not deliver any more energy. We argue that *the system reaches $T$ with probability 1 under any discharge policy.* This is because of the following reasons: (a) We consider only work conserving policies here, (b) a work conserving policy always serves a packet as long as the battery has not expired (c) a cell can deliver at most $C$ units of charge ($c_i$ can not increase in subsequent slots as per the system evolution) and thus the battery can deliver at most $LC$ units of charge (d) there is a nonzero probability of packet arrival in every slot ($a_0 < 1$) and each packet consumes at least one unit of charge. Thus the battery management problem satisfies all the characteristics of a stochastic shortest path problem as described in [4].

## 3.3 Formalization of the Optimal Solution

Since the battery management problem belongs to the broad class of stochastic shortest path problems, the optimal energy and cell selection can be obtained by solving Bellmans Equation given in [4].

***Proposition 1*** *The optimal energy for a state $x$, $J^*(x)$ satisfies Bellmans Equation given below.*

$$J^*(x) = \max_{u \in U(x)} \left[ \bar{q}(x,u) + \sum_{y=1}^{M} p_{xy}(u) J^*(y) \right], \tag{4}$$

*where $U(x)$ is the set of active cells in state $x$, and $\bar{q}(x,u)$ is given in (3).*

*A cell selection function $\mu^*(x)$ is optimal if and only if*

$$\mu^*(x) = \arg\max_{u \in U(x)} \left[ \bar{q}(x,u) + \sum_{y=1}^{M} p_{xy}(u) J^*(y) \right], \tag{5}$$

Now, standard techniques like value iteration and policy iteration may be used to solve the Bellmans Equation, as described in [4]. However, these general methods are not suitable for systems with a large state space, such as in our case. For example, when $C = 20, N = 10, L = 2$, $M \approx 40000$. The value iteration method is an iterative procedure which may need a large number of iterations to converge (potentially infinite), depending on the initial choices for the iterates. On the other hand, the policy iteration method involves $K$ iterations in the worst case, where $K$ is the total number of possible policies ($K$ is $O(ML)$) and each iteration requires the solution of a total of $M$ linear equations with $M$ variables $O(M^3)$ [10]. Thus, the overall complexity is $O(M^4L)$, which is large in general, , even for small values of N, C. In the next section we show how to solve Bellmans Equation (4) (and thereby obtain the optimal policy) in $O(M)$, exploiting the specific characteristics of the battery management system.

# 4 A Linear Complexity Algorithm for Computing the Optimal Strategy

In this section, we design a simplified computation scheme which obtains the optimal strategy in linear complexity ($O(M)$) (subsection 4.1). Subsequently, in subsection 4.2 we will discuss some salient features of the computational framework.

## 4.1 Design of the linear complexity computation technique

We make two significant observations which will help us develop our technique. Firstly, $J^*(n_1, c_1, \ldots, n_i, c_i, \ldots, n_L, c_L) = J^*(n_1, c_1, \ldots, c_i, c_i, \ldots, n_L, c_L)$, if $n_i > c_i$ for any cell $i$. For example, $J^*(3,2,3,4) = J^*(2,2,3,4)$. A cell in state $(n_i, c_i)$, $n_i > c_i$, cannot deliver more than $c_i$ charge units. The reasons are: (a) the remaining capacity $c_i$ decreases when a cell is being discharged, (b)$c_i$ can never increase and (c) the cell is fully discharged when $c_i = 0$. Thus, one need not consider states with $n_i > c_i$ for any cell $i$ for the purpose of optimal computation. This is equivalent to the condition that the recovery probability $p_r(n_i, c_i) = 0$ if $n_i = c_i$. Thus the new recovery probability for a cell $i$ is given by:

$$p_r(n_i, c_i) = \begin{cases} e^{-g(N-n_i)-\phi(c_i)}, & 1 \le n_i < N, \\ & n_i < c_i, 1 \le c_i < C \\ 0, & \text{otherwise} \end{cases} \qquad (6)$$

The second observation is that when the system is in state $x$ there is only a limited number of states that the system can move to from state $x$, under any cell selection. Note that the system may remain in the same state $x$ with some probability. Let $S_u(x)$ denote the set of next states the system can move to from state $x$, except $x$, if cell $u$ is selected, i.e., $S_u(x) = \{y : y \neq x, p_{xy}(u) > 0\}$. Note that $p_{xy}(u)$ can be computed as in Equation (2) except that now the recovery probability is given by Equation (6). Since $p_{xy}(u) = 0$ for all $y \notin S_u(x)$, Bellmans equation (equation (4)) can be rewritten as:

$$\begin{aligned} J^*(x) &= \max_{u \in U(x)} \left[ \bar{q}(x,u) + p_{xx}(u)J^*(x) + \sum_{y \in S_u(x)} p_{xy}(u)J^*(y) \right], \\ &= a_0 \Pi_{l=1}^{M} \left(1 - pr(n_{lx}, c_{lx})\right) J^*(x) + \max_{u \in U(x)} \left[ \bar{q}(x,u) + \sum_{y \in S_u(x)} p_{xy}(u)J^*(y) \right] \end{aligned}$$

since $p_{xx}(u) = a_0 \Pi_{l=1}^{M} \left(1 - pr(n_{lx}, c_{lx})\right) \forall u$

The important observation here is that a system remains in the same state only if there is no new arrival in a slot (i.e., $q_i = 0$ ), and this event is independent of the cell selection. Thus $p_{xx}(u)$ is independent of $u$.

$$\begin{aligned} \text{Thus, } J^*(x) &= \frac{\max_{u \in U(x)} J_u(x)}{1 - a_0 \Pi_{l=1} \left(1 - pr(n_{lx}, c_{lx})\right)} && (7) \\ \text{where } J_u(x) &= \bar{q}(x,u) + \sum_{y \in S_u(x)} p_{xy}(u)J^*(y) && (8) \end{aligned}$$

(Intuitively $J_u(x)$ is the energy delivered if cell $u$ is chosen when the system is in state $x$.)

We define $S(x) = \cup_{u \in U(x)} S_u(x)$. According to Equation (7) the optimal energy $J^*(x)$ can be computed if we know $J^*(y)$ for all $y \in S(x)$. Using specific properties of the battery management problem and Equation (7), we will present an algorithm which computes $J^*(x)$ sequentially such that $J^*(y)$ is already computed for all $y \in S(x)$ before computing $J^*(x)$.

We now describe the set $S_u(x)$ using the transition probabilities given in equation (2). Let $x = (n_{1x}, c_{1x}, \ldots, n_{Lx}, c_{Lx})$, then a state $y = (n_{1y}, c_{1y}, \ldots,$

$n_{Ly}, c_{Ly})$ is in $S_u(x)$ if and only if: (a) $y \neq x$, (b)$c_{iy} = c_{ix}$, $n_{iy} \in \{n_{ix}, n_{ix} + 1,\}$ $i \neq u$, and (c) $(n_{uy}, c_{uy}) \in \{(n_{ux} - q, c_{ux} - q), q > 0,\ (n_{ux} + 1, c_{ux})\}$. For example, in a 2-cell system, if $x = (1,2,1,2)$ then $S_1(x) = \{(2,2,1,2), (2,2,2,2), (1,2,2,2), (0,1,1,2),\ (0,1,2,2)\}$.

We denote as $diff_i(x)$ the difference $c_i - n_i$ between the remaining capacity and the charge of a cell $i$, when the system is in state $x$. For example, for a 2-cell system, $\text{diff}_1(1,2,3,5) = 1$, and $\text{diff}_2(1,2,3,5) = 2$. The key point to observe is that for all cells $i$ and states $y$ in $S(x)$, either $\text{diff}_i(y) = \text{diff}_i(x)$ or $\text{diff}_i(y) = \text{diff}_i(x) - 1$. Since $J^*(x)$ can be computed using (7), only if $J^*(y)$ is known for all $y$ in $S(x)$, our approach is to initially compute the $J^*(x)$ for the states $x$ which have lower values of these "diff" functions, and subsequently move to the states with higher values of these functions.

We also use the following additional properties of the optimal energy functions $J^*(x)$ at different stages of the computation.

1. From symmetry, $J^*(n_1, c_1, \ldots, n_i, c_i, \ldots, n_j, c_j, \ldots, n_L, c_L) = J^*(n_1, c_1, \ldots, n_j, c_j, \ldots, n_i, c_i, \ldots, n_L, c_L)$. For example, $J^*(1,2,3,4) = J^*(3,4,1,2)$. This symmetry reduces the number of states for which we need to compute the optimal energies by a factor of $L!$.

2. Also, $J^*(c_1, c_1, \ldots, c_L, c_L) = \sum_{i=1}^{L} c_i$. For example, $J^*(3,3,4,4) = 7$. This follows from the observation that a battery in state $(n_1, c_1, \ldots, n_L, c_L)$ can deliver at least $\sum_{i=1}^{L} n_i$ units and at most $\sum_{i=1}^{L} c_i$ units. Note that $\text{diff}_i(c_1, c_1, \ldots, c_L, c_L) = 0$ for all cells $i$. Thus, we know the optimal energies for the states with zero values of the "diff" functions without any computing, and we use these known values to compute the optimal energies of other states.

3. We also know that $J^*(x) = 0$, for any termination state $x \in T$. Also, $J^*(n_1, c_1, \ldots, 0, c_i, \ldots, n_L, c_L) = J^*(n_1, c_1, \ldots, 0, 0, \ldots, n_L, c_L)$. This again follows since a cell with 0 remaining charge can not serve any further packet.

We present our computation technique in Figure 2. For simplicity we describe the technique for the 2−cell case only, i.e., $L = 2$. This is for ease of presentation, and the generalization for the multiple cell case is straight forward. In fact in Section 5 we also give the maximum energies for a larger number of cells.

In Fig. 2 the terms $J_1(i,j,k,l)$ and $J_2(i,j,k,l)$ are the individual terms in the maximization in the right hand side of Equation (7) and can be computed using Equation (8).

```
Procedure Optimal_Energy()
begin
  for diff_1 = 1 to C-N do
    for i=1 to N do
      j=i+diff1;
      for diff_2 = 0 to diff_1 do
        if (diff_2 < diff_1) then
          last=N;
        else
          last=i;
        for k=0 to last do
          l=k+diff_2;
          /*START OF BLOCK 1 */
          J*(i,j,k,l) = max{J_1(i,j,k,l), J_2(i,j,k,l)} / (1 - a_0 Π_{l=1}^{M} (1 - pr(n_{li}, c_{li}))), where J_1, J_2 are given in Equation 8
          if J_1(i,j,k,l) > J_2(i,j,k,l) then
            μ*(i,j,k,l) = 1
          else
            μ*(i,j,k,l) = 2
            /* END OF BLOCK 1 */
  J* = J(N,C,N,C);
end
```

Figure 2: Computation of Optimal energy and cell selection procedure for a system with two cells.

**Illustrative Example:** We now illustrate the operation of our technique. We consider a battery with 2 cells, nominal capacity $N = 2$ and theoretical capacity $C = 4$. This example will show the sequence of computation, and demonstrate that when the algorithm computes $J^*(x)$, $J^*(y)$ for all states $y \in S(x)$ is already known. The overall sequence of computation is $(1,2,0,0), (1,2,1,1), (1,2,2,2), (1,2,0,1), (1,2,1,2), (2,3,0,0), (2,3,1,1),$ $(2,3,2,2), (2,3,0,1), (2,3,1,2), (1,3,0,0), (1,3,1,1), (1,3,2,2), (1,3,0,1),$ $(1,3,1,2), (1,3,2,3), (1,3,0,2), (1,3,1,3)(2,4,0,0), (2,4,1,1), (2,4,2,2),$ $(2,4,0,1), (2,4,1,2), (2,4,2,3), (2,4,0,2), (2,4,1,3), (2,4,2,4)$.

The optimal energy is first computed for the state $x = (1,2,0,0)$ ($\text{diff}_1(1,2,0,0) = 1$, $i = 1$, $\text{diff}_2(1,2,0,0) = 0$, $k = 0$). Here $U(x) = \{1\}$, as any work conserving policy uses cell 1 to serve a packet. Now, $S_1(x) = \{(2,2,0,0), (0,1,0,0)\}$ and $J^*(x) = J_1(x)$. Now, $J_1(x)$ can be computed as the optimal energies for all states in $S_1(x)$ are known, $J^*(2,2,0,0) = 2, J^*(0,1,0,0) = 0$, according to properties (2),(3). In the second iteration $J^*(1,2,1,1)$ is computed. Now, $U(x) = \{1,2\}$. We have $S_1(x) = \{(2,2,1,1), (0,1,1,1)\}$, and $S_2(x) = \{(2,2,1,1), (1,2,0,0), (2,2,0,0)\}$. Note that $J^*(y)$ is known for all $y$ in $S_1(x), S_2(x)$ from the previous iteration and properties (2) and (3) of the function $J^*(x)$. Similarly, the optimal energies for the rest of the states are computed. For example, when we compute the energies for the state $x = (1,3,1,3)$, we have $S_1(x) = \{(2,3,2,3),(2,3,1,3),(1,3,2,3),(0,2,1,3),(0,2,2,3)\}$ and $S_2(x) = \{(2,3,2,3),(2,3,1,3),(1,3,2,3),(1,3,0,2),\ (2,3,0,2)\}$. The optimal energies for all the above states are known from the previous iterations (refer to the sequence of computations given above) and by using the symmetry property (1) of $J^*(x)$.

**Proof of correctness of the technique:** We need to show that the technique given in Figure 2 computes the optimal energies for each state $x = (n_1, c_1, n_2, c_2)$. Note that the algorithm computes the energies for states $x$ with $\text{diff}_1(x) > \text{diff}_2(x) \geq 0$ and $\text{diff}_1(x) = \text{diff}_2(x) > 0$ if $n_2 \leq n_1$. The first question is whether the optimal energies of all other states can be computed using these values. This follows from properties (1) to (3) of $J^*(x)$ and the fact that $\text{diff}_i(x) \geq 0$ for all cells $i$. The argument is as follows. The algorithm does not compute $J^*(x)$ if (a)$\text{diff}_1(x) < \text{diff}_2(x)$ or if (b)$\text{diff}_1(x) = \text{diff}_2(x)$ and $n_1 < n_2$ or $n_1 = c_1$, and $n_2 = c_2$. Consider case (a) first. Let $z = (n_2, c_2, n_1, c_1)$. Note that $\text{diff}_1(z) > \text{diff}_2(z)$. Thus the algorithm computes $J^*(z)$, and we know that $J^*(x) = J^*(z)$ from symmetry. Now consider case (b). Let $n_1 < n_2$. Again, the algorithm computes $J^*(z)$, and thus $J^*(x)$ is obtained from symmetry. Finally, if $x = (c_1, c_1, c_2, c_2)$ $J^*(x) = c_1 + c_2$ from condition (2), and need not be computed separately.

Next, we need to show that the energies computed by the algorithm are the optimal energies $J^*(x)$ which satisfy Bellmans equation (4). Note that the computation of $J^*(x)$ in Figure 2 follows relation (7). Thus, we only need to show that $J^*(y)$ for all $y \in S(x)$ is computed before $J^*(x)$. We show this by induction. The base case is $(1,2,0,0)$ and the result holds for $(1,2,0,0)$ as argued in the previous example. We assume that all states considered before $x$ satisfy this property. We show that the result holds in the induction case for state $x$ by addressing several subcases separately. Note that $\text{diff}_i(y) \in \{\text{diff}_i(x), \text{diff}_i(x) - 1\}$, $i = 1,2$ for any $y \in S(x)$. Thus, the subcases we have to consider are: (a) $\text{diff}_1(y) = \text{diff}_1(x), \text{diff}_2(y) = \text{diff}_2(x) - 1$, (b)$\text{diff}_i(y) = \text{diff}_i(x), i = 1,2$ (c) $\text{diff}_1(y) = \text{diff}_1(x) - 1$, $\text{diff}_2(y) = \text{diff}_2(x)$ and (d)$\text{diff}_1(y) = \text{diff}_1(x) - 1$, $\text{diff}_2(y) = \text{diff}_2(x) - 1$.

Let $x = (n_1, c_1, n_2, c_2)$. Thus for the first subcase $y = (n_1 - d, c_1 - d, n_2 + 1, c_2), d \geq 0$. Note that $n_2 < c_2$ as a cell can not recover charge otherwise from (6). From the computation sequence of the algorithm, before any state $(n_1, c_1, n_2, c_2)$ is considered, all states $(n_1 - d, c_1 - d, k, l)$ are considered if $l - k < c_1 - n_1$. Since the algorithm is trying to compute $J^*(x)$ for state $x$, $c_2 - n_2 \leq c_1 - n_1$. As such, $c_2 - n_2 - 1 < c_1 - n_1$. Thus the state $(n_1 - d, c_1 - d, n_2 + 1, c_2)$ has been considered. Thus the result follows.

We consider the second subcase now. Let $x = (n_1, c_1, n_2, c_2)$. Here, $y = (n_1, c_1, n_2 - d, c_2 - d)$, or $y = (n_1 - d, c_1 - d, n_2, c_2)$ for some $d > 0$. (Note that $d \neq 0$ as $y \neq x$ for any $y \in S(x)$). For the first case, $J^*(y)$ is computed before, as $\text{diff}_i(y) = \text{diff}_i(x)$, $i = 1, 2$ and $n_2 - d < n_2$. Consider the second case. Again if $c_2 - n_2 < c_1 - n_1$, $J^*(y)$ is computed before as $\text{diff}_i(y) = \text{diff}_i(x)$, $i = 1, 2$ and $n_1 - d < n_1$. Let $c_2 - n_2 = c_1 - n_1$ (Note that $c_2 - n_2 \leq c_1 - n_1$ since $x$ is being considered). If $n_2 \leq n_1 - d$, $y$ is considered before $x$. Let $n_2 > n_1 - d$. Note that $n_2 \leq n_1$ since $c_2 - n_2 = c_1 - n_1$ and $x$ is being considered. Consider $z = (n_2, c_2, n_1 - d, c_1 - d)$. Note that $\text{diff}_2(z) = c_1 - n_1 = c_2 - n_2 = \text{diff}_2(x)$, and $\text{diff}_1(z) = c_2 - n_2 = c_1 - n_1 = \text{diff}_1(x)$. If $n_2 < n_1$, then since $\text{diff}_i(z) = \text{diff}_i(x)$, $i = 1, 2$ and $n_1 - d < n_2$ $J^*(z)$ is considered before. Also, $J^*(y) = J^*(z)$ from the symmetry property. Thus $J^*(y)$ is known. Now, let $n_2 = n_1$. Thus, $x = (n_1, c_1, n_1, c_1)$, $y = (n_1 - d, c_1 - d, n_1, c_1)$ and $z = (n_1, c_1, n_1 - d, c_1 - d)$. Clearly $z$ is considered before $x$ since $d > 0$. Thus, $J^*(z)$ and hence $J^*(y)$ is known by symmetry. Thus the result follows.

We consider the third subcase now. Let $x = (n_1, c_1, n_2, c_2)$. Now, $y = (n_1 + 1, c_1, n_2 - d, c_2 - d)$, $d \geq 0$. Let $c_2 - n_2 < c_1 - n_1 - 1$. In this case, $\text{diff}_2(y) < \text{diff}_1(y)$. Also, $\text{diff}_1(y) < \text{diff}_1(x)$. Thus $J^*(y)$ is computed before. Now let $c_2 - n_2 = c_1 - n_1 - 1$. Thus, $\text{diff}_2(y) = \text{diff}_1(y) < \text{diff}_1(x)$. If $\text{diff}_1(y) = \text{diff}_2(y) = 0$, $n_1 + 1 = c_1$ and $n_2 = c_2$. Thus, $y = (c_1, c_1, c_2 - d, c_2 - d)$, and $J^*(y)$ is known. Let, $\text{diff}_2(y) = \text{diff}_1(y) > 0$. If $n_2 - d \leq n_1 + 1$ $J^*(y)$ is computed before. Otherwise (i.e., if $n_2 - d > n_1 + 1$) $J^*(n_2 - d, c_2 - d, n_1 + 1, c_1)$ is computed before and $J^*(y) = J^*(n_2 - d, c_2 - d, n_1 + 1, c_1)$ from symmetry. Finally let $c_2 - n_2 = c_1 - n_1$ (Note that $c_2 - n_2 \leq c_1 - n_1$ as the algorithm is trying to compute $J^*(x)$). It follows that $n_2 \leq n_1$. Consider $z = (n_2 - d, c_2 - d, n_1 + 1, c_1)$. Thus, $\text{diff}_2(z) < \text{diff}_1(z) = \text{diff}_1(x)$, and $n_2 - d \leq n_2 \leq n_1$. Thus, $J^*(z)$ and hence $J^*(y)$ is known by symmetry. Thus the result follows.

We consider the fourth subcase now. Let $x = (n_1, c_1, n_2, c_2)$. Now, $y = (n_1 + 1, c_1, n_2 + 1, c_2)$. If $n_1 + 1 = c_1$ then $c_1 - n_1 = 1$ and thus $c_2 - n_2 \leq 1$ as $\text{diff}_2(x) \leq \text{diff}_1(x)$ (The last inequality holds as the state $x$ is being considered currently). It follows that $n_2 + 1 = c_2$ since $\text{diff}_i(y) \geq 0$ for all states $y$. Thus, $y = (c_1, c_1, c_2, c_2)$ and hence $J^*(y)$ is known. Now, let $n_1 + 1 < c_1$. Since $\text{diff}_i(y) = \text{diff}_i(x) - 1$, $i = 1, 2$ $J^*(y)$ is computed before $J^*(x)$, only if (a)$\text{diff}_2(y) < \text{diff}_1(y)$ or (b)$0 < \text{diff}_2(y) = \text{diff}_1(y)$ and $n_2 \leq n_1$. If $\text{diff}_2(x) < \text{diff}_1(x)$, then condition (a) holds as $\text{diff}_i(y) = \text{diff}_i(x) - 1$, $i = 1, 2$. Otherwise, (i.e., if $\text{diff}_2(x) = \text{diff}_1(x)$,) $\text{diff}_2(y) = \text{diff}_1(y)$ since $\text{diff}_i(y) = \text{diff}_i(x) - 1$, $i = 1, 2$. Now, $n_2 \leq n_1$ since $\text{diff}_2(x) = \text{diff}_1(x)$ and $J^*(x)$ is being computed. Since $n_1 + 1 < c_1$ by assumption, $\text{diff}_1(y) > 0$

and hence $\mathrm{diff}_2(y) > 0$. Thus condition (b) is satisfied in this case. Thus the result follows.

### 4.2 Discussion of Salient Features

Note that the algorithm "visits" every state $x$ at most once when it encounters state $x$ in Block 1 in Figure 2. The computation complexity of $J^*(x)$ depends on the size of $S(x)$. Note that $|S(x)|$ is $(p+2)^L$ if the maximum size of a packet is $p$. However, in a more efficient implementation, the computation of $J^*(x)$ can be performed in constant complexity. The idea is to use the already available computation results for other states. Thus, the complexity is linear in size of the state space $M$. Also, the storage required for this algorithm is $O(M)$. However, it is possible to reduce the storage substantially with certain observations (e.g., the 2-cell case needs a storage of $2N(C-N+1)(N+1)$ only, whereas $M = ((N+1)(C+1))^2$). We would like to point out that this strategy can be computed offline and thus a node can execute the optimal cell selection only by a table lookup procedure. The lookup table will need to store the optimal cell selection for $O(M)$ states (actually fewer than $M$ states need be stored because of state space reduction techniques discussed before) and the lookup complexity will also be $O(M)$. Now, $M$ can be large for real systems. Thus, we believe that the principal use of this optimum strategy will be as a "benchmark" for comparing the performance of online suboptimal strategies with the optimal energies. For example, we propose a simple suboptimal policy in Section 5 which can be used to choose the cell in an online fashion, and subsequently we use the computation presented here to show that the suboptimal policy delivers near-optimum energy.

Even though the computation complexity is linear in $M$, $M$ itself can be large for moderately large values of $C$ and $N$ ($M = ((N+1)(C+1))^L$). However, we could still compute the optimal strategy in order of minutes for $C = 200, N = 25, L = 2$ using an Ultra-SPARC SUN machine. The standard value and policy iteration techniques were consuming several hours (more than 10 hours) for the same numbers. We could also compute the optimal strategy for moderate values of $C$ and $N$ for $L = 4$. We present these computation results in Sections 5 and 6 (Figs. 3, 4 and 11).

The computational framework and the technique presented here make no assumption about the packet size distribution, except independence of the packet sizes from slot to slot. Thus, the computations can be used for a large number of different traffic models. We present results for two different size distributions in Section 5 and we observe that the optimal energy obtained can be quite different for different size distributions. Note that *the packet size depends on the transmission power which in turn depends on the transmission conditions, and the transmission conditions may have different distributions for different scenarios in the wireless case. Thus, it is important to accommodate different traffic distributions in the optimal framework.* Realistically, transmission conditions need not be independent in different slots. However, this technique can be generalized to capture Markovian dependencies in packet sizes. In this context, this strategy can also be used for different recovery probabilities.

Finally, the framework and the linear complexity computation technique presented here are not restricted to computation of the maximum energy. The same strategy can be used to obtain the energy attained by many other cell selection policies. We illustrate this in the next section.

# 5 Suboptimal policy

We consider a simple scheduling policy which aims to efficiently choose the cell to be discharged, so as to approximate the optimal. The choice of the cell depends on the remaining charge of all cells. More specifically, the incoming packet is assigned to the cell with the maximum remaining charge. It is possible to instantly monitor the level of charge in each cell using smart battery packages [3]. We denote this policy as the "Maximum Charge" (MC) policy.

We compute the total energy delivered by the MC strategy using the theory of stochastic dynamic programming once again. We first introduce the concept of stationary policies. A stationary policy is one in which the cell selection policy does not change with time, and the actual selection depends on time only through the state value, e.g., if the state is the same for two different slots, then the selection will also be the same for these slots under a stationary policy. Note that the optimal policy which satisfies equation (5) is stationary. From the stochastic shortest path framework [4], the energy obtained by a stationary policy $\mu$ starting from state $x$ is given by:

$$J_{\mu}(x) = \bar{q}\left(x, \mu(x)\right) + \sum_{y=1}^{M} p_{xy}\left(\mu(x)\right) J_{\mu}(y), \tag{9}$$

where $\bar{q}\left(x, \mu(x)\right)$ is given in Equation (3) above.

Next, arguing as in the derivation of Equation (7) from Bellmans Equation, we have

$$J_\mu(x) = \frac{\bar{q}(x,\mu(x)) + \sum_{y\in S_{\mu(x)}(x)} p_{xy}(\mu(x))J^*(y)}{1 - a_0\Pi_{l=1}^{M}(1 - p_r(n_{lx}, c_{lx}))} \tag{10}$$

This is similar to Equation (7). Thus we use a technique similar to Fig. 2 to compute $J_\mu(x)$, for any stationary policy $\mu$. The only modification is to replace block 1 by Equation (10). Observe that the MC policy is stationary and thus we can use this technique to compute the energy $J_{MC}(x)$ obtained by the system, starting from state $x$.

The idea behind MC is that it provides an efficient way to discharge the cells, since it allows the "most discharged" cells to recover. Intuitively, it should perform close to the optimal. We corroborate this observation with the numerical results presented in Figs. 3 and 4. Also, MC is easy to implement, as it does not need any table lookup as opposed to the optimal strategy. We also show by simulation that MC provides significant improvement in battery lifetime compared to the Round Robin policy proposed in [2].

We now describe the numerical and simulation performance evaluation of MC. We consider the performance metric $G$ which is the ratio between the total number of charge units delivered (A) and the maximum number of charge units that can be delivered by a battery of $L$ cells, $(L * C)$, i.e., $G = \frac{A}{L*C}$.

We first describe the different traffic models we will use. Firstly, we consider a Poisson traffic model. Here the probability that a packet of size $q$ arrives is $a_q = \frac{R^q e^{-R}}{q!}$, where $R$ is the average packet size. We also consider a different traffic model, where packets are normally of size 1 (with probability $a_1 = \alpha p$) but occasionally have a larger size $b$ ("burst") with probability $a_b = \alpha(1-p)$. Also, the probability of zero arrivals is $a_0 = 1 - \alpha$. This model corresponds to a realistic scenario where transmissions are usually good except occasionally, due to "fading". When the channel is good, only one charge unit is required to transmit a packet. During fading, the energy required to transmit a packet will be larger and equal to $b$.

In Fig. 3 we consider Poisson traffic. We compare the performance ratio $G$ for the optimal policy and the Maximum Charge policy as a function of the average packet size $R$, for the case of $L = 2$ cells. We choose the parameter values as $C = 200$ and $N = 25$, based on the parameter choices in [2]. We also give the results for $C = 200, N = 12$. As we can see in Fig. 3, MC closely follows the optimal. For very small $R$, since the cells are allowed to rest for longer periods, both policies perform well. For $R = 0.3$ to

0.6, MC differs from the optimal by at most 10%. In other ranges, they are very close. For larger $R$, the performance for both policies is significantly reduced. Especially after $R = 1$, $G$ is less than 0.5 for both policies. This is due to the fact that when a cell recovers it can only gain one charge unit, but when $R > 1$ more than one charge units are discharged for an average packet.

In Fig. 4 we plot $G$ as a function of the probability of zero packet arrival $a_0$ for the optimal and MC, for "bursty" traffic, where $p = 0.75$, $b = 3, 8$, $C = 200$, $N = 25$ and $L = 2$. MC is performing close to the optimal in general. Note that the performance of both MC and optimal are significantly worse for $b = 8$ as compared to $b = 3$. The performance is also worse compared to the Poisson traffic. For example, for average packet size 0.75 the optimal gives $G = 0.95$ for Poisson traffic, $G = 0.8$ and 0.6 for bursty traffic, for $b = 3$ and $b = 8$ respectively. This can be explained by the fact that the probability of large packets is higher for bursty traffic as compared to Poisson and it increases with $b$.

We now compare the performance of MC with RR presented in [2] using simulation. In Figs. 5, 6 the results for $L = 2$ and $L = 4$ are given for Poisson traffic, while Figs. 7-10 present the results for bursty traffic, for $b = 3$ and $b = 8$. In all cases we choose $C = 200$, $N = 12$ or $N = 25$, and we plot the percentage difference ($\frac{G_{MC} - G_{RR}}{G_{RR}} * 100$) as a function of the average load per cell ($R/L$) for Poisson traffic and as a function of $a_0$ for bursty traffic. When the load is low, the cell selection is not critical and several policies will perform well. On the other side, when the load is high then the battery lifetime will be low, irrespective of the cell scheduling. Thus, the critical region is for intermediate load, where the appropriate cell selection can make a difference in the attained energy. MC significantly outperforms RR in this region, for both traffic models. In case of Poisson traffic, the difference reaches high values for $C = 200, N = 12$, e.g., the difference is above 100% for $L = 4$ and load $R/L = 0.6$ (Fig. 5). The trends are similar for $N = 25$ (Fig. 6), though the percentage difference is smaller than in the case of $N = 12$.

In case of bursty traffic, the results in Figs. 7-10 show that MC attains an improvement of up to 35%when $b = 3$ and up to 165% when $b = 8$. When the $b$ is large, the performance of RR decreases rapidly while MC still performs well, even for heavy load. This can be explained by the fact that when a packet of larger size arrives the RR policy may assign the burst to a cell that is close to being completely discharged, thus quickly draining off all of its energy. However, MC carefully assigns the large burst to the cell with larger charge and thus it discharges the cells in a more fair manner,

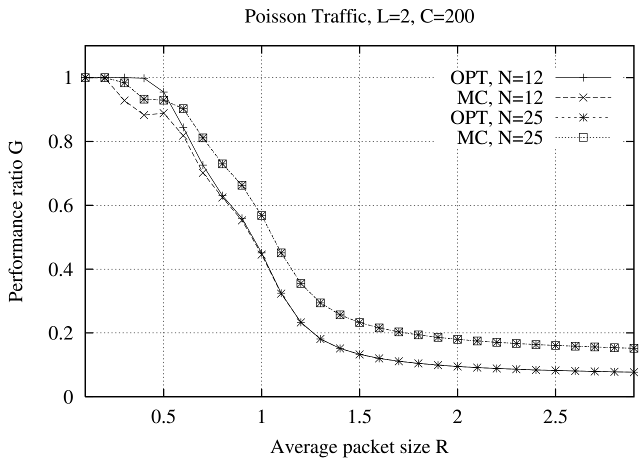


Figure 3: Performance ratio $G$ of Optimal and MC for $C = 200$ and $N = 12$, 25 as a function of the average packet size $R$, for Poisson Traffic

allowing them a longer period to recover.

# 6 Approximation Algorithm

We observed that the linear complexity algorithm of Section 4 for computing the optimal scheduling may become computationally intensive on account of the large state space. The objective of this section is to approximate the original system with a system with reduced state space, and compute the optimal energy and the scheduling in the new system using the technique of Section 4. The hope is that the optimal solutions for the two systems will be close to each other, and as such the low computational complexity solution of the latter will approximate the former.

We observe that the size of the state space of the system increases exponentially with the number of cells $L$, ($M = ((N+1)(C+1))^L$). Thus, our approach is to aggregate the cells in groups, and consider the optimal scheduling among the groups. Once a group is selected, a cell can be selected optimally within the group. If the cells are divided in $K$ groups of $L/K$ cells each, then the new system which considers groups only, has a state space of size $((LN/K+1)(LC/K+1))^K$. The state space of the system representing one group has $((N+1)(C+1))^{L/K}$ states. Thus the overall state space of the two tier system is $V = ((LN/K+1)(LC/K+1))^K + ((N+1)(C+1))^{L/K}$. If $K = \sqrt{L}$, the total number of states is $O\left((LNC)^{\sqrt{L}}\right)$.

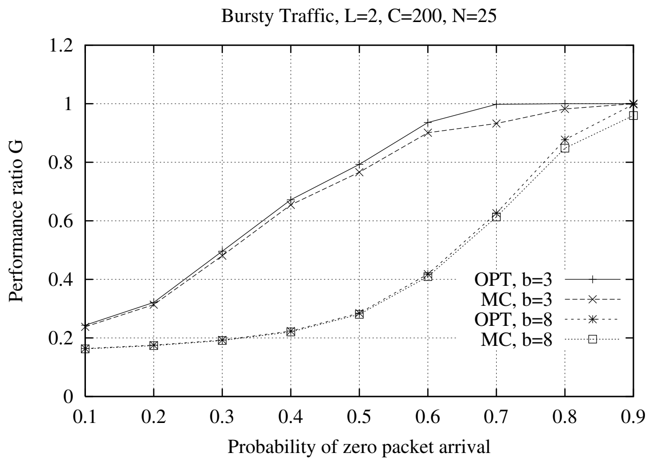


Figure 4: Performance ratio of Optimal and MC for $C = 200$,$N = 25$ and different values of burst size ($b$) as a function of the probability of zero packet arrivals, for Bursty Traffic

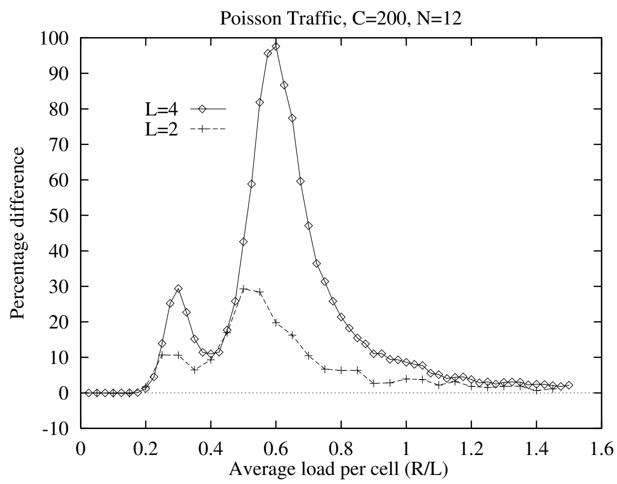


Figure 5: Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 12$, as a function of the average load per cell ($R/L$), for Poisson Traffic

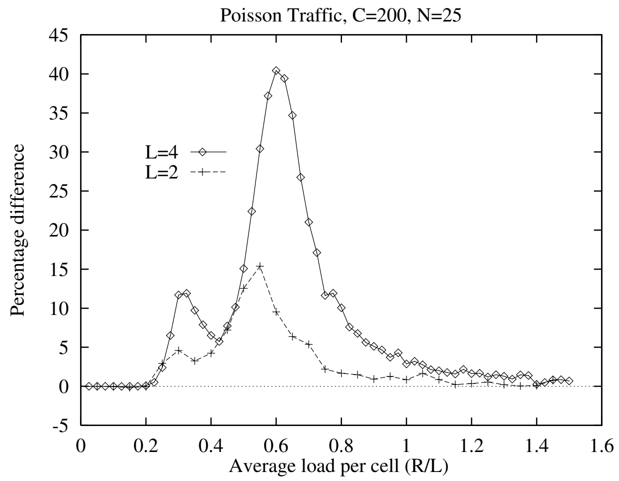


Figure 6: Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 25$, as a function of the average load per cell $(R/L)$, for Poisson Traffic

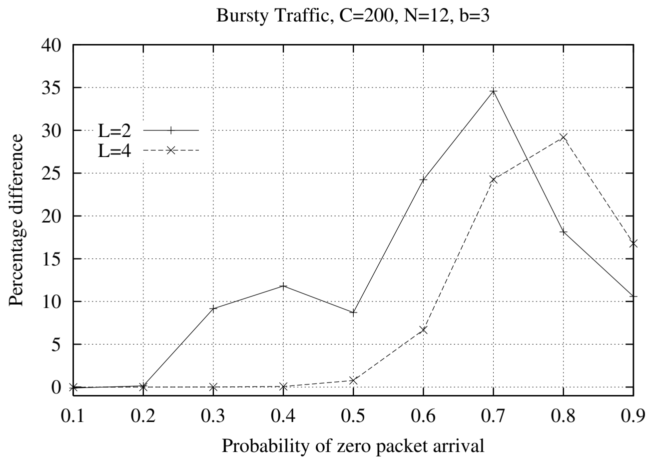


Figure 7: Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 12$, $b = 3$, as a function of the probability of zero packet arrivals, for Bursty Traffic

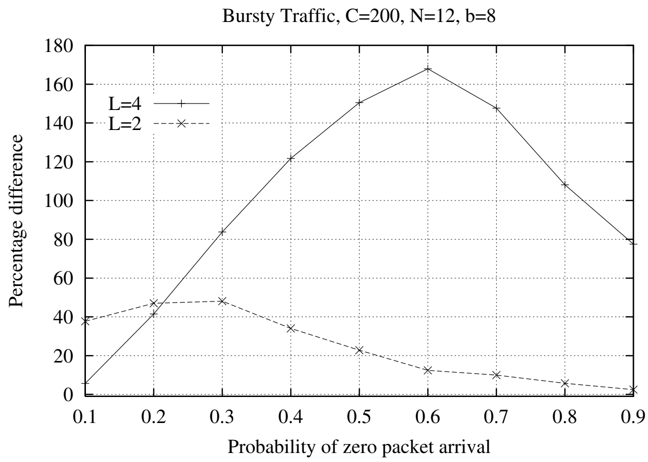


Figure 8: Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 12$, $b = 8$, as a function of the probability of zero packet arrivals, for Bursty Traffic

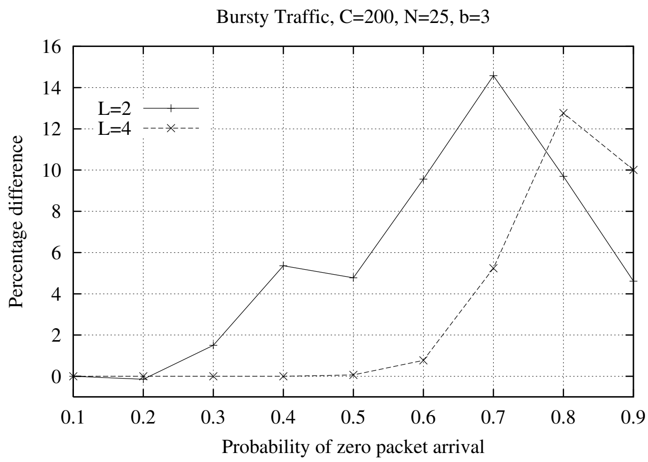


Figure 9: Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 25$, $b = 3$, as a function of the probability of zero packet arrivals, for Bursty Traffic

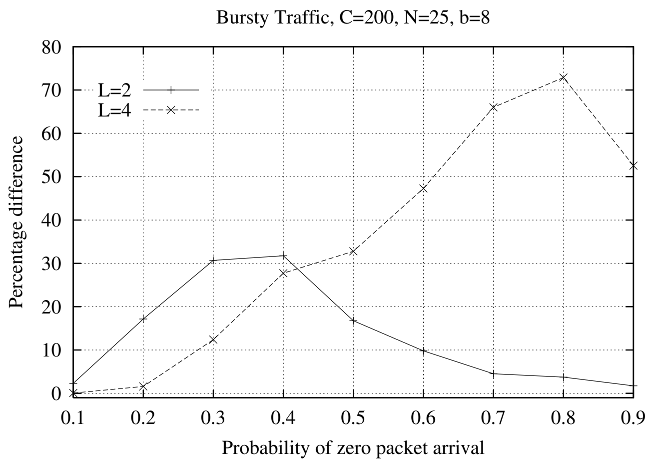


Figure 10: Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 25$, $b = 8$, as a function of the probability of zero packet arrivals, for Bursty Traffic

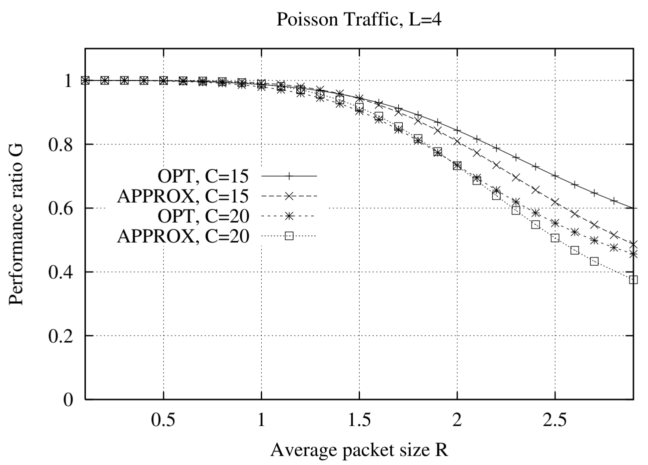


Figure 11: Performance ratio of Optimal and Approximation policy for $C = 15$, $C = 20$ and $N = 5$ as a function of average packet size $R$, for Poisson Traffic

Note that the exponent in the state space is decreased from $L$ to $\sqrt{L}$. A direct extension is to consider a multi-tier approach of first dividing the system in two groups, subsequently dividing these groups in subgroups, and considering their scheduling, etc.

We only describe the model in detail for the two tier approach. We divide the system in $K$ groups and consider the selection among the groups first, and subsequently consider the selection among the cells of each group. The state $(n, c)$ of each group is its remaining charge $n$ and capacity $c$. The remaining charge (capacity) is the sum of the remaining charges (capacities) of all cells in the group. The state of the system is a vector of the state of all the groups in the system. For example, the state of a system for 4 cells and 2 groups is $(n_1, c_1, n_2, c_2)$. We assume that the remaining charge and capacity of a group are equally divided between its cells, i.e., if a group of 2 cells has remaining charge of 4 units, then each cell has 2 units of charge. The assumption has been motivated by our observation that in general the optimal scheduling of Section 4 selects the cells so as to roughly equalize their charges and capacities.

When a group does not serve a packet, all its constituents can recover their charges. Thus if a $s-$cell group in state $(n, c)$ does not serve a packet, its recovery probability is binomial$(s, p_r(n/s, c/s))$ , where $p_r(n/s, c/s)$ is given in Equation (6) (we assume that $n$ and $c$ are multiples of $s$). Note that if $s = 1$ the recovery process is the same as for the battery model presented in Section 2. When a $s-$cell group in state $(n, c)$ serves a packet of size $q$, its charge and capacity decrease by $q$. However, we assume that only one cell serves the packet in the group, and as such the other $s-1$ cells can potentially recover. Thus, if a group with $s$ cells and state $(n, c)$ serves a packet, then its next state is $(n-Q+R, c-Q)$ where $Q$ and $R$ represent the arrival and charge recovery processes respectively. Here, distribution of $Q$ is given by $\{a_q\}$ and $R$ is binomial$(s-1, p_r(n/s, c/s))$ . Note that $Q$ and $R$ are independent, given that $Q > 0$. If $Q = 0$, then all $s$ cells can potentially recover, and the recovery process $R$ is binomial$(s, p_r(n/s, c/s))$ . The distributions for these random variables, and hence the state transition probabilities can be computed easily. The energy delivered by the system when choosing a group $u$ equals the minimum of the packet size, the remaining charge and capacity of the group. The average energy delivered, when a s-cell group in state $(n, c)$ is chosen, is given by

$$\bar{q}(x, u) = \sum_{i=1}^{\min(\frac{n_u}{s}, \frac{c_u}{s})} i a_i + \min(\frac{n_u}{s}, \frac{c_u}{s}) \left( 1 - \sum_{i=1}^{\min(\frac{n_u}{s}, \frac{c_u}{s})} a_i \right) \tag{11}$$

The objective is to select the groups so as to maximize the expected energy delivered.

The state evolution and the energy-reward structure of the new system falls in the stochastic shortest path framework given in [4]. It can also be shown that the computational technique of Section 4 applies to this system as well. Thus, the optimal energy obtained for this system can be computed in $O(V)$ where $V$ is the total number of states of this system, and $V << M$. This is because $V \approx O\left((LNC)^{\sqrt{L}}\right)$, $M \approx O\left((NC)^{L}\right)$, if the system is divided into $\sqrt{L}$ groups of $\sqrt{L}$ cells each, and $L << NC$.

We would like to mention in this context that the approximation via reduction of state space is a topic of ongoing research. We report the results here, as the numerical computations show that the optimum energy delivered by the two-tier system is close to that of the actual system for several choices of parameters. The intuition behind the proximity of the results is that the state evolution of the two systems are similar. We present the results for a system with 4 cells aggregated in 2 groups for the approximation. Here, $N = 5$ and $C = 20$ or 15. Thus $M \approx 13 * 10^9$ and $V \approx 16 * 10^5$. In Figure 11 we compare the performance of the approximation and the optimal for Poisson traffic, in case $L = 4$. It can be seen that the reduced system approximates the optimal closely in general. The performance metric $G$ differs by at most 0.1, which happens for $R > 2$. The discrepancy can be explained by the fact that for larger packet sizes the states of charge for the cells in a group are not equal in the actual system, while the reduced system still assumes equality. The results are encouraging considering the significant reduction in the state space. We plan to investigate different grouping systems and different recovery procedures for the reduced system so as to improve the approximation.

# 7 Conclusion

In this paper, we obtain an optimal battery discharge policy, for maximizing the lifetime of the power-limited wireless nodes. We use general results from stochastic dynamic programming framework and also exploit specific characteristics of the battery management problem to design the optimal solution. Even though the computation complexity of the optimal is linear in the number of system states, the computation may become prohibitive on account of the large size of the state space. Next, we design a computationally simple discharge strategy and show that the lifetime attained by this policy is close to that of the optimal. Finally, we approximate the computation of the optimal using state space reduction techniques. As future work, we plan to explore the approximation strategy further, and investigate the effects of the proposed battery management strategies in a network scenario.

| Symbol | Meaning |
|---|---|
| C | theoretical capacity |
| N | nominal capacity |
| L | number of cells |
| M | total number of states $M = ((N+1)(C+1))^L$ |
| $n_u$ | remaining charge of cell $u$ |
| $t_u$ | remaining capacity of cell $u$ |
| $p_r(n_u, t_u)$ | recovery prob. of cell $u$ in state $(n_u,\, t_u)$ |
| $J^*(x)$ | optimal energy of the system when in state $x$ |
| $\mu^*$ | optimal cell selection |
| $a_d$ | prob. of arrival of a packet of size $d$ |
| $p_{xy}(u)$ | transition prob. from state $x$ to state $y$ under cell selection $u$ |
| $T$ | set of termination states |
| $\bar{q}(x,u)$ | avg energy obtained in state $x$ under cell selection $u$ |

Table 1: Notation

## References


[1] C. F. Chiasserini and R. R. Rao, *A traffic control scheme to optimize the battery pulsed discharge*, Proc. of Milcom99, Atlantic City, NJ, Nov. 1999.

[2] C. F. Chiasserini and R. R. Rao, *Energy Efficient Battery Management*, Proc. of Infocom 2000, Tel Aviv, Israel, March 2000.

[3] C. F. Chiasserini and R. R. Rao, *Pulsed Battery discharge in communication devices*, MobiCom99, Seattle, pp. 88-95, August 1999.

[4] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control* Vol. I, 2nd Edition, published Nov.2000

[5] H. D. Linden, *Handbook of batteries,* 2nd edition, New York, McGraw-Hill, 1995

[6] T.F. Fuller, M. Doyle, and J.S. Newman, *Relaxation phenomena in lithium-ion-insertion cells*, J. Electrochem. Soc., vol. 141, no. 4, pp.982-990, Apr. 1994.

[7] Gray Girling, Jennifer Li Kam Wa, Paul Osborn, Radina Stefanova *The Design and Implementation of a Low Power Ad Hoc Protocol Stack* Presented at IEEE Wireless Communications and Networking Conference, September 2000, Chicago

[8] Tajana Simunic, Haris Vikalo, Peter Glynn and Giovanni De Micheli *Energy efficient design of portable wireless systems,* Proceedings of the 2000 international symposium on Low power electronics and design, July 25 - 27, 2000, Rapallo Italy

[9] Cadex Electronics Inc., *Intelligent Batteries,* http://www.cadex.com/cfm.index.

[10] Conte and deBoor, *Elementary Numerical Analysis: An algorithmic Approach,* McGraw-Hill, 1980

# A Overview of Stochastic Shortest Path Problem

In this section we give a brief overview of the stochastic shortest path problem. We will first present general characteristics which identify such a system, and subsequently present computational techniques for solving such a problem. Refer to [4] for more details.

## A.1 System Description and Identifying Characteristics:

The stochastic shortest path problem considers a discrete time dynamic process. The state of the system at time $k$ is represented by $x_k$, and $x_k$ can have one of $M$ possible values, where $M$ is a system parameter. The system chooses a control decision $u_k$ at each time $k$ from a finite state of all possible decisions, $U(x_k)$ (the set of allowable decisions $U(x_k)$ may depend on $x_k$). From each state $x$, under decision $u$ the system moves to the next state $y$ with transition probability $p_{xy}(u)$. *The next state is independent of the past states and controls given the current state and control decision.*

A reward $g(x_k, y_k, u_k)$ is associated with the current state $x_k$, the next state $y_k$ and the decision $u_k$ made at time $k$. The expected reward $g(i,u)$ for state $i$, under control decision $u$ is then $g(x_i,u) = \sum_j p_{ij}(u)\overline{g}(i,u,j)$. The reward accumulates over time and the goal is to choose the control policy so as to maximize the total expected reward.

The system can control the decision $u_k$ made at each state $k$ and thereby control the next state it will move to. This affects the reward. A *policy* $\pi = \{\mu_0(x_0), \mu_1(x_1), \ldots,\}$ is a sequence of decision rules or functions $\mu_k(x_k)$

which maps the state $x_k$ to the control decision $u_k = \mu_k(x_k)$ taken at time $k$. In other words, the policy gives a rule for choosing the control as a function of the current state. The policy affects the evolution of the system states and the total reward obtained. The total expected reward accumulated under a policy $\pi$ and initial state $x_0$ is

$$J_\pi(x_0) = \lim_{N \to \infty} E\left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k)) \middle| x_0 \right\} \tag{12}$$

The stochastic shortest path systems have certain states with specific properties, known as the "termination states." As the name indicates, these states have the property that when the system reaches one of the termination states, it stays there without any further reward, and the process is said to terminate. In addition, *termination happens with probability one under all policies.* Thus, the duration of such a process is random but finite with probability one and depends on the control decisions used. The goal of such a process is to choose the policy so as to reach the termination state with the maximum expected reward.

The optimal expected reward from state $x_i$ is denoted $J^*(x_i)$. An optimal policy $\pi^*$ is one which maximizes the expected reward starting from any initial state. Thus the maximum expected reward, under initial state $x_0$ is $J^*(x_0) = J_{\pi^*}(x_0) = max_{\pi \in \Pi} J_\pi(x_0)$, where $\Pi$ is the set of all policies. The optimal rewards can be computed using some general purpose results for this class of problems. We state the general results next.

## A.2 General Computation Strategies for Optimum Solution:

**Proposition 2** *The optimal rewards* $(J^*(1), J*(2), \ldots J^*(M))$ *satisfy the Bellmans equation given below.*

$$J^*(i) = \max_{u \in U(i)} \left[ g(i,u) + \sum_{j=1}^{M} p_{ij}(u) J^*(j) \right], \tag{13}$$

*A policy* $\mu(i)$ *is optimal if and only if*

$$\mu(i) = \arg\max_{u \in U(i)} \left[ g(i,u) + \sum_{j=1}^{M} p_{ij}(u) J^*(j) \right], \tag{14}$$

The above proposition tells us that the optimal policy can be computed if the optimal rewards $J^*(j)$ and the transition probabilities $p_{ij}(u)$ are known for all states $i, j$ and control $u$. The optimal rewards can be computed using one of the following methods, the *value iteration* or the *policy iteration* method [4].

In case of the *value iteration* method, the optimal rewards can be iteratively computed as follows.

**Proposition 3** *Given any initial conditions $J_0(1), \ldots, J_0(M)$, the sequence $J_k(i)$ generated by the DP iteration*

$$J_{k+1}(i) = \max_{u \in U(i)} g(i,u) + \sum_{j=1}^{N} p_{ij}(u) J_k(j), \qquad i = 1, \ldots, N \tag{15}$$

*converges to the optimal reward $J^*(i)$ for each state $i$.*

This method may require an infinite number of iterations to converge.

The other method, *policy iteration* always terminates in a finite number of iterations. The scheme can be described as follows.

**Proposition 4** *The optimal policy is computed by the following sequence of iterations starting from any initial policy $(\mu^0(1), \mu^0(2), \ldots, \mu^0(M))$.*

$$J_{k+1}(i) = g(i, \mu^k(i)) + \sum_{j=1}^{n} p_{ij}(u) J_{k+1}(j), \qquad i = 1, \ldots, M \tag{16}$$

$$\textit{and } \mu^{k+1}(i) = argmax_{u \in U(i)} \left[ g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_{\mu^k}(j) \right], \qquad i = 1, \ldots, M \tag{17}$$

*The process terminates when $\mu^{k+1} = \mu^k$. If the process terminates in $k$ iterations, then $J_k(1), \ldots J_k(M)$ are the optimal rewards and $\mu^k$ is the optimal policy.*

It can be shown that this procedure yields the optimal solution in a finite number of iterations. However, policy iteration involves solving $M$ linear equations (step (16)) with $M$ variables. The linear system can be solved using standard methods, such as Gaussian elimination. However, when the size of the state space $M$ is large, the computation can consume significant time and memory.

### A.3 General Computation Strategies for Computing Total Reward:

Similar methods can be used for computing the total reward attained by any "stationary" policy. We first need to introduce the concept of stationary policies for presenting these results. A stationary policy is one in which the rule does not change with time, and the actual control action depends on time only through the state value, e.g., if the state is the same for two different slots, then the control action will also be the same for these slots under a stationary policy. Note that the optimal policy which satisfies equation (14) is stationary. We now state the general results regarding the computation of the rewards attained by any arbitrary stationary policy, $\mu$, $(J_\mu(1), \ldots, J_\mu(M))$.

**Proposition 5** *The optimal rewards* $(J_\mu(1), J_\mu(2), \ldots J_\mu(M))$ *satisfy the Bellmans equation given below.*

$$J_\mu(i) = \left[ g\left(i, \mu(i)\right) + \sum_{j=1}^{M} p_{ij}\left(\mu(i)\right) J_\mu(j) \right], \tag{18}$$

The result is similar to Proposition 5. Thus the rewards $(J_\mu(1), \ldots, J_\mu(M))$ under a stationary policy $\mu$ can be computed using value-iteration and policy-iteration based techniques.

Incidentally, the stochastic shortest path problem has been described in [4] in terms of minimization of costs rather than the maximization of rewards. However, the results hold for the current framework of maximization of rewards as well.