Optimal Dispatching Control for Elevator Systems During Uppeak Traffic

David L. Pepyne and Christos G. Cassandras, *Fellow, IEEE*

***Abstract*— In this paper we develop optimal dispatching controllers for elevator systems during uppeak traffic. An uppeak traffic period arises when the bulk of the passenger traffic is moving from the first floor up into the building (e.g., the start of a business day in an office building). The cars deliver the passengers and then return empty to the first floor to pick up more passengers. We show that the structure of the optimal dispatching policy minimizing the discounted or average passenger waiting time is a *threshold-based* policy. That is, the optimal policy is to dispatch an available car from the first floor when the number of passengers inside the car reaches or exceeds a threshold that depends on several factors including the passenger arrival rate, elevator performance capabilities, and the number of elevators available at the first floor. Since most elevator systems have sensors to determine the car locations and the number of passengers in each car, such a threshold policy is easily implemented. Our analysis is based on a Markov decision problem formulation with a batch service queueing model consisting of a single queue served by multiple finite-capacity bulk servers. We use dynamic programming techniques to obtain the structure of the optimal control policy and to derive some of its important properties. Several numerical examples are included to illustrate our results and to compare the optimal threshold policy to some known *ad hoc* approaches. Finally, since many transportation systems can be modeled as multiserver batch service queueing systems, we expect our results to be useful in controlling those systems as well.**

***Index Terms*— Bulk-service queueing networks, dynamic programming, Markov decision problems, optimal control, optimization problems, queueing theory, thresholds, transportation models.**

I. Introduction

ELEVATOR systems form a class of discrete-event systems (DES's) whose complexity makes them difficult to model, analyze, and optimize. In multiple-car elevator systems, particularly those designed to serve large buildings, a major challenge is that of developing a *dispatching control policy*, i.e., a scheme for systematically deciding *when* and *where* each car should move, stop, or switch direction based on the current state and available past history. While in general, the objective of an elevator dispatching policy depends on the particular building, for office buildings, the usual goal is to minimize the average passenger waiting time [20]. Achieving this objective is difficult for a number of reasons, including the need to: 1) coordinate multiple cars; 2) satisfy constraints on elevator movement (e.g., a car must stop at a floor where a passenger wants to exit); 3) operate with incomplete state information (e.g., while it is known whether an elevator has been called to a particular floor, it is generally not known how many passengers are waiting at that floor); 4) make decisions in the presence of uncertainty (e.g., passenger arrival (pa) times and destinations are uncertain); and 5) handle nonstationary passenger traffic (e.g., for an office building, passenger traffic varies continuously throughout the day, from morning up-traffic, to heavy two-way lunchtime traffic, to evening down-traffic). Even without difficulties 4) and 5), the dispatching control problem is combinatorially explosive due to the enormous size of the state space.

A systematic study of the elevator dispatching control problem begins by decomposing passenger traffic into four different situations: 1) *uppeak traffic*; 2) *lunchtime traffic*; 3) *downpeak traffic*; and 4) *interfloor traffic* [20]. The uppeak traffic situation arises when all passengers are moving up from the first floor (e.g., the start of the business day in an office building). Lunchtime traffic is a characterization in which passengers are going to and returning from the first floor (e.g., as they go to and return from lunch in an office building). The downpeak traffic situation is observed when all passengers are moving down to the first floor (e.g., the end of the business day when an office building is emptied). Finally, interfloor traffic is a characterization in which passengers are moving equally likely between floors.

In this paper we limit ourselves to the uppeak traffic situation and develop the theory for optimal dispatching. During uppeak, passengers arrive only at the first floor. The elevators carry them up to their requested destinations and then make an express run returning empty to the first floor to serve additional passengers. While the uppeak traffic situation is arguably the simplest one to model, it is the most difficult one for an elevator system to handle from the standpoint of passenger handling capacity [19]. Because passengers are arriving to a single floor during uppeak, it is possible for a very high pa intensity to cause cars to fill up and the lobby queue to grow unbounded. In fact, the anticipated pa intensity during uppeak is used in planning the size of the elevator system that will be needed to serve a building [4].

Historically, when the first passenger elevators were introduced in the 1890's, each car was individually controlled by an attendant riding the car. As building heights rose, however,

Manuscript received January 31, 1996; revised December 16, 1996. Recommended by Associate Editor, C. C. Lee. This work was supported in part by the National Science Foundation under Grants ECS-93-11776 and EID-92-12122, by AFOSR under Contract F49620-95-1-0131, and by a grant from United Technologies/OTIS Elevator.

D. L. Pepyne is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA.

C. G. Cassandras was with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA. He is now with the Department of Manufacturing Engineering, Boston University, Boston, MA 02215 USA.

Publisher Item Identifier S 1063-6536(97)07780-4.

1063–6536/97$10.00 © 1997 IEEE

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

so did the number and speed of the cars and it soon became impossible for the attendants to provide effective coordination and control. With the introduction of the first semiautomatic elevator controllers in the 1920's, the attendant's job was reduced to one of simply closing the doors and starting the car. By 1950, fully automated elevator controllers eliminated the attendant altogether. The first automated elevator controllers were simple electromechanical relay systems. By the 1970's, microprocessor based elevator controllers were in common use. Since that time, increases in processor speed and memory capacity have allowed for the implementation of increasingly complicated dispatching algorithms. Modern dispatching algorithms employ fuzzy logic [2], [14], expert systems [22], sophisticated rule-based and search-based strategies [3], [16], artificial intelligence with learning [13], dynamic programming [10], [15], and reinforcement learning [7], [17]. In general, different dispatching algorithms are used for the different traffic situations defined earlier. While modern dispatching algorithms give good performance, most are *ad hoc* and heuristic, designed using experience, intuition, and simulation, as opposed to formal techniques based on optimal control theory.

For the uppeak situation, the dispatching objective is reduced to the question of when to dispatch an elevator from the first floor. The simplest algorithm for uppeak dispatches an elevator as soon as the first passenger boards. Another, termed *half-capacity plus time-out*, dispatches an elevator whenever half its capacity is reached or when a timer, started when the first passenger enters the elevator, expires (usually a 20-s timer is used). The main contribution of this paper is to show that the structure of the optimal dispatching policy, minimizing the discounted or average passenger waiting time for uppeak traffic, is a *threshold-based* policy. That is, the controller should dispatch an elevator when the number of passengers inside a car reaches or exceeds a certain threshold. In practice the number of passengers in a car is estimated by an on-board scale measuring the total weight of passengers; more sophisticated systems use light beams to detect passengers entering and exiting the cars. Since most elevator systems have some method to determine the number of passengers in each car as well as sensors to determine the car locations, such a threshold policy is easily implemented. In contrast to the *ad hoc* dispatching algorithms described above, our analysis will show that the thresholds are not fixed, but depend on the pa rate, the performance capabilities of the elevators, and the number of elevators available at the first floor. Although our analysis does not provide explicit numerical values for the thresholds, parameterizing the control policy in terms of a few thresholds, allows us to use any of a number of recently developed methods based on perturbation analysis and sample path constructability techniques for DES [6], [9], [11] to determine them on-line from actual observable system data. A detailed scheme for determining the thresholds will be given in a follow-up paper, where a dispatching controller is developed with the ability to adapt the threshold values as operating conditions (such as the pa rate) in the system change.

Our analysis is based on modeling the dispatching problem in uppeak traffic as a Markov decision problem (MDP) [5], [18] and applying dynamic programming techniques to derive the structural properties of the optimal control policy. The basic model is that of a queueing system consisting of a single queue (representing the first-floor lobby where arriving passengers wait for a car) served by multiple, finite capacity bulk servers (representing the cars). In [1] a similar model was used to analyze the uppeak traffic situation. No control strategy, however, was developed. The optimality of a threshold policy has been shown for a single server batch service queueing system [8]. In this paper, however, we consider a multiple-server system in which each server is limited to a finite capacity. We note that since many transportation systems, in addition to the uppeak elevator dispatching problem, can be modeled as multiple bulk server queueing systems (for example, airport shuttle busses, shipment of parcels or military supplies, etc.) we expect our results to be useful in controlling those systems as well.

The remainder of this paper is organized as follows. In the next section, we formulate an MDP model for the uppeak elevator dispatching problem. In Section III, we consider the discounted cost problem and derive the associated dynamic programming equations. In Section IV we present properties of the value function and use these properties to show the optimality of a threshold policy for the discounted cost problem. In Section V we extend the optimality of a threshold policy to the average cost problem. Several examples are presented in Section VI to verify our analysis. Finally, we end in Section VII with a summary and discussion. Two appendixes provide proofs for the lemmas and corollaries used in the body of the paper.

II. Problem Formulation

In this section, we first present a queueing model for the uppeak traffic situation (Section II-A) and then develop a MDP for the corresponding dispatching control problem (Section II-B). We consider the case of an elevator system with two cars to keep the analysis manageable. As will be seen, however, extensions to the $N>2$ car case follow naturally and in a straightforward way.

A. The Queueing Model

For the elevator system we consider, we assume that the uppeak traffic originates from a single floor (the first floor), and that each elevator serves every floor (i.e., "zoning" [4] is not used). Then we can model the uppeak traffic situation as a single queue of infinite capacity (representing the first-floor lobby), served by two identical bulk servers (corresponding to two identical elevator cars), each with a finite capacity of C passengers. Fig. 1 illustrates this model, in which passengers arrive one at a time to the queue according to a Poisson process with intensity λ. Each pa generates a pa event. The passengers are admitted into the cars and the cars are dispatched by the dispatching control. Exactly how dispatching control is exercised in this model is discussed in the next section. The passengers are served by the cars in batches of size no greater than the car capacity C. The time for a car to serve a batch of passengers is exponentially distributed with parameter μ,

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

a constant, which is the same for each car, and independent of the state. After a car has delivered all of its passengers, it makes an immediate express run returning empty to the first floor lobby to serve more uppeak passengers. The completion of service generates a "car arrival" (ca) event indicating that one of the two elevators has become available for service. Since the two elevators are identical, there is no need to distinguish between them.

It is worth making a few remarks regarding this model. First, experience has shown the Poisson process to be a good model of pa's [12], particularly for the case of a medium-sized office building (one with 10–15 floors) in a suburban office park, where the building's occupants will typically drive to work alone. Next, modeling service times through an exponential distribution with a constant rate is intended to aggregate random effects due to travel time, number of stops, passenger unloading time, door opening and closing time, and the door holding time. Although it is possible to use a more elaborate service time model (e.g., using a distribution from the Erlang family or taking into account the destination floors already selected by passengers inside an elevator that is waiting to be dispatched), doing so greatly increases the complexity of the problem and makes analysis intractable. More importantly, from a practical standpoint, our purpose here is to derive the *structure* of an optimal dispatching policy, structure which can be used to develop dispatching controllers that do not depend on modeling assumptions regarding the distributions of the arrival and service processes.

B. The Markov Decision Problem

An MDP formulation is now introduced to rigorously define the uppeak dispatching problem (see [5], [6], and [18] for a general background on MDP's). The state-space X for the model is obtained by defining $y(t) \in \{0,1,\cdots\}$ to denote the queue length at the first floor lobby at time t and $z(t) \in \{0,1,2\}$ to denote the number of elevators available at the first floor at time t. Thus, $X = \{(y,z)\colon\ y = 0,1,\cdots, z = 0,1,2\}$. When needed, we will denote the state by $x = (y,z)$. State transitions in this model are the result of event occurrences; in particular, pa events or ca events. Control actions are taken only when any such event occurs and they define a set $U = \{0,1,2\}$ where

Action $u = 0$:	do nothing, hold all available cars at the first floor;
Action $u = 1$:	load one car and dispatch it;
Action $u = 2$:	allow both cars to load and dispatch them simultaneously.

Since cars returning to the lobby are assumed to be empty, each available car can serve up to C passengers from the lobby. Those passengers which would cause a car to overflow will remain at the lobby to wait for another one. If we define

$$[y-C]^+ = \max\{y-C,0\}$$
$$[y-2C]^+ = \max\{y-2C,0\} \tag{1}$$

then dispatching one elevator serves $\min\{y,C\}$ passengers and leaves behind a lobby queue of length $[y-C]^+$ passengers

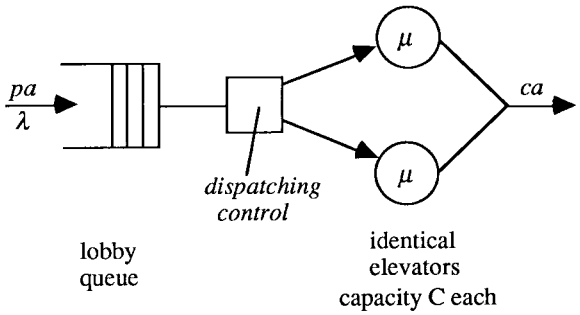

Fig. 1. Queueing model for the two-car uppeak dispatching problem.

(i.e., dispatching control only allows passengers to load into one car at a time), while dispatching both elevators serves $\min\{y,2C\}$ passengers and leaves behind a lobby queue of length $[y-2C]^+$ passengers.

Observe that not all actions are admissible at every state. In particular, let $U(y,z)$ (a subset of U) denote the set of admissible actions from the state $x = (y,z)$ and we have

$U(y,0) = \{0\}$:	dispatching is not allowed when no cars are available;
$U(y,1) = \{0,1\}$:	two cars cannot be dispatched when only one is available;
$U(y,2) = \{0,1,2\}$:	when both cars are available; all actions in U are admissible.

To implement the control action $u = 1$ when both cars are available implies the ability to load one car before loading the other car. This is typically implemented using the popular "next car" feature [4]. Since returning cars are empty, they do not need to open their doors when they reach the main lobby; thus, to force passengers to load one car at a time, only one car opens its doors. This car is referred to as the "next car" to be dispatched. Note, even when a car returns to the first floor with down passengers and must open its doors to discharge them, it is still possible to implement the "next car" feature by discouraging passengers from entering the car by dimming its lights and making it appear as if the car is out of service.

Next, we use the standard uniformization technique [5], [6] to convert the continuous-time MDP above into an equivalent discrete-time MDP. This is accomplished by choosing a uniform rate $\gamma = \lambda + 2\mu$, the total event rate in our two-car model (this obviously extends to $\gamma = \lambda + N\mu$ for an N-car model). In this uniformized model, fictitious ca events (causing no state change) are included to account for states where the feasible event rate is less than γ. Without loss of generality, we can assume the time scale has been normalized so that $\gamma = 1$. Control actions are taken at the beginning of each time step. Let $P_{ij}(u)$ denote the conditional probability that the state at the next time step is $j \in X$ given that the state at the current time step is $i \in X$ and the control action taken at the beginning of the current time step is $u \in U(i)$. These state transition probabilities are given by

$$P_{ij}(0) = \begin{cases} \lambda, & i = (y,0), j = (y+1,0) \\ 2\mu, & i = (y,0), j = (y,1) \end{cases} \tag{2a}$$

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

$$P_{ij}(0)=\begin{cases}\lambda, & i=(y,1), j=(y+1,1)\\ \mu, & i=(y,1), j=(y,2)\\ \mu, & i=j=(y,1)\end{cases} \tag{2b}$$

$$P_{ij}(0)=\begin{cases}\lambda, & i=(y,2), j=(y+1,2)\\ 2\mu, & i=j=(y,2)\end{cases} \tag{2c}$$

$$P_{ij}(1)=\begin{cases}\lambda, & i=(y,1), j=([y-C]^+ +1,0)\\ 2\mu, & i=(y,1), j=([y-C]^+,1)\end{cases} \tag{2d}$$

$$P_{ij}(1)=\begin{cases}\lambda, & i=(y,2), j=([y-C]^+ +1,1)\\ \mu, & i=(y,2), j=([y-C]^+,2)\\ \mu, & i=(y,2), j=([y-C]^+,1)\end{cases} \tag{2e}$$

$$P_{ij}(2)=\begin{cases}\lambda, & i=(y,2), j=([y-2C]^+ +1,0)\\ 2\mu, & i=(y,2), j=([y-2C]^+,1).\end{cases} \tag{2f}$$

For each (2a)–(2f), the first row corresponds to a state transition induced by a pa event. All remaining transitions are induced by a ca event, including fictitious ca events introduced by uniformization. In (2b), for example, the last ca event is fictitious because one of the cars is available at the lobby and, therefore, cannot generate an actual ca event. In (2c), both ca events are fictitious because both cars are available at the lobby. In (2e) the last ca event is fictitious. In this case, it may appear that this fictitious ca event causes a state change, which is not the case. Here, the state at the beginning of the time step is $i=(y,2)$. Taking the action $u=1$ at the beginning of the time step causes an immediate state change to an "intermediate" state $i'=([y-C]^+,1)$. When the fictitious ca event occurs, there is no state change with respect to the "intermediate" state, i.e., $j=i'$. It is the control action taken at the beginning of the time step that causes the state change, and not this fictitious ca event. In the remaining (2a), (2d), and (2f) all ca events are real.

Notice that although we are taking control actions at the beginning of each time step, we would never take an action (other than $u=0$) when a fictitious event occurs. This is because actions are only taken in response to state changes, and fictitious events do not cause state changes. Therefore, if a state is such that an action $u\neq 0$ should have been taken, the action would have been taken in response to the real event that caused the system to transition to the state, and not in response to fictitious events occurring at later time steps.

To complete the MDP formulation, we introduce the following cost structure. Let us denote the cost for the kth time step by $C(y_k,z_k,u_k)$ where u_k is the control action taken at the beginning of the time step when the state is (y_k,z_k). Our objective then is to obtain the optimal stationary policy π^* that minimizes the total discounted cost to be incurred over the infinite-horizon

$$V_{\pi^*}^{\alpha}(i)=\inf_{\pi}E_{\pi}\left[\sum_{k=0}^{\infty}\alpha^k C(y_k,z_k,u_k)|x_0=i\right] \tag{3}$$

where $E[\cdot]$ denotes the expectation operator, and $\alpha(0<a<1)$ is a given discount factor. Obtaining the optimal stationary policy ϕ^* minimizing the average cost

$$\overline{V}_{\phi^*}=\inf_{\phi}\left[\limsup_{N\to\infty}\frac{1}{N}E_{\phi}\left[\sum_{k=0}^{N-1}C(y_k,z_k,u_k)\right]\right] \tag{4}$$

will be considered later in Section V.

We will take the one-step cost to be proportional to the queue length resulting from the control action taken at the beginning of the time step. Letting β be some given positive and bounded holding cost, we have

$$C(y,0,u)=\beta y \tag{5a}$$

$$C(y,1,u)=\begin{cases}\beta[y-C]^+ & u=1\\ \beta y & u=0\end{cases} \tag{5b}$$

$$C(y,2,u)=\begin{cases}\beta[y-2C]^+ & u=2\\ \beta[y-C]^+ & u=1\\ \beta y & u=0\end{cases} \tag{5c}$$

where the actions $u=1,2$ reduce the lobby queue length in accordance with (1). This cost structure is motivated by the fact that the minimization of the average queue length is equivalent to the minimization of the average passenger waiting time in the sense that at steady state

$$E[\text{queue length } y]=\lambda E[\text{passenger waiting time}]$$

by Little's Law [6, p. 345].

The following lemma establishes the fact that *any* policy π yields a finite cost.

Lemma 2.1: $V_{\pi}^{\alpha}(i)<\infty$ for $i\in X$ and all policies π, provided $\alpha\in(0,1)$ and $\beta<\infty$.

Proof: Since customers arrive one at a time, for any initial queue length $y_0<\infty$ and any policy π, the lobby queue length at time step k satisfies $y_k\leq y_0+k$. Hence

$$\begin{aligned}V_{\pi}^{\alpha}(i)&=E_{\pi}\left[\sum_{k=0}^{\infty}\alpha^k C(y_k,z_k,u_k)|x_0=i\right]\\ &\leq E_{\pi}\left[\sum_{k=0}^{\infty}\alpha^k\beta y_k|x_0=i\right]\leq\sum_{k=0}^{\infty}\alpha^k\beta(y_0+k)\\ &=\beta\left[\frac{y_0}{1-\alpha}+\frac{\alpha}{(1-\alpha)^2}\right]<\infty.\end{aligned} \tag{6}$$

♦

III. The Dynamic Programming Equations

Given that all policies yield a finite cost by Lemma 2.1, we wish to find the one that gives the least cost in (3). In this section we develop the dynamic programming equations satisfied by such a policy. Let $V_n^{\alpha}(i)$ denote the optimal cost-to-go over n time steps starting with state i. Then, since the one-step costs defined in (5) are nonnegative and the action set $U=\{0,1,2\}$ is finite, it is well known (see, for example, [5] and [6]) that, for $\alpha\in(0,1)$, the dynamic programming algorithm

$$V_{n+1}^{\alpha}(i)=\min_{u\in U(i)}\left[C(i,u)+\alpha\sum_j P_{ij}(u)V_n^{\alpha}(j)\right] \tag{7}$$

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

with $V_0^\alpha(i) = 0$ converges to the optimal value function, i.e.,

$$V_{\pi^*}^\alpha(i) = \lim_{n\to\infty} V_n^\alpha(i) \tag{8}$$

and that $V_{\pi^*}^\alpha(i)$ satisfies the optimality equation

$$V_{\pi^*}^\alpha(i) = \min_{u\in U(i)} \left[C(i,u) + \alpha \sum_j P_{ij}(u) V_{\pi^*}^\alpha(j) \right]. \tag{9}$$

Moreover, under the same conditions, there is an optimal stationary policy obtained through

$$u^*(i) = \arg\min_{u\in U(i)} \left[C(i,u) + \alpha \sum_j P_{ij}(i) V_{\pi^*}^\alpha(j) \right]. \tag{10}$$

Using the state transition probabilities (2a)–(2f) and the state transition costs (5a)–(5c), we can obtain dynamic programming equations of the form (7) as follows. First, for states of the form $(y,0)$, no cars are available so the only admissible action is $u = 0$ (do nothing), i.e.,

$$V_{n+1}^\alpha(y,0) = \beta y + \alpha\lambda V_n^\alpha(y+1,0) + 2\alpha\mu V_n^\alpha(y,1). \tag{11a}$$

In (11a), as will be the case in all the dynamic programming equations to follow, the first term is the one-step cost, the second term corresponds to a pa event, and all remaining terms correspond to ca events.

Similarly, for states of the form $(y,1)$, there is one car available and the control action is to either hold it waiting for more passengers to arrive $(u=0)$ or dispatch it $(u=1)$, i.e.,

$$\begin{aligned} &V_{n+1}^\alpha(y,1) \\ &= \min\{\beta y + \alpha\lambda V_n^\alpha(y+1,1) + \alpha\mu V_n^\alpha(y,2) \\ &\quad + \alpha\mu V_n^\alpha(y,1) \\ &\quad \beta[y-C]^+ + \alpha\lambda V_n^\alpha([y-C]^+ + 1,0) + 2\alpha\mu V_n^\alpha([y-C]^+,1)\}. \end{aligned} \tag{11b}$$

The first term in the bracket in (11b) corresponds to holding the car, and the second corresponds to dispatching the car. Also, note that in the first term, one of the ca events is fictitious and causes no state change, since only one car is busy. In the second term, however, both ca events are real: one car was already busy at the beginning of the time step, and the other was made busy when it was dispatched at the beginning of the time step.

Finally, for states of the form $(y,2)$, all actions are admissible, i.e., hold both cars $(u=0)$, load and dispatch one of them $(u=1)$, or load and dispatch both of them $(u=2)$, and we get $(u=2)$

$$\begin{aligned} &V_{n+1}^\alpha(y,2) \\ &= \min\{\beta y + \alpha\lambda V_n^\alpha(y+1,2) + 2\alpha\mu V_n^\alpha(y,2) \\ &\quad \cdot \beta[y-C]^+ + \alpha\lambda V_n^\alpha([y-C]^+ + 1,1) \\ &\quad + \alpha\mu V_n^\alpha([y-C]^+,2) \\ &\quad + \alpha\mu V_n^\alpha([y-C]^+,1) \\ &\quad \beta[y-2C]^+ + \alpha\lambda V_n^\alpha([y-2C]^+ + 2\alpha\mu V_n^\alpha([y-2C]^+,1)\}. \end{aligned} \tag{11c}$$

The first term in the bracket above corresponds to holding both cars (hence, both ca events are fictitious), the second to dispatching one of the cars (in which case only the ca event corresponding to the elevator that was dispatched at the beginning of the time step is a real one), and the third to dispatching both cars (hence, both ca events are real).

Using the dynamic programming equations (11a)–(11c) above, the optimal dispatching policy can be obtained from the dynamic programming algorithm (7) by letting $n \to \infty$. Finding the optimal dispatching policy this way, however, is prohibitive in terms of the computational and memory requirements, especially since the dynamic programming algorithm must be solved, and the optimal policy stored, for each set of system parameters $\{\beta, \lambda, \mu, C, \alpha\}$ of interest. In what follows, we show that the optimal policy for each such set can be parameterized in terms a few thresholds.

IV. Structure of the Optimal Policy

In this section we use the dynamic programming equations (11a)–(11c) to show that the structure of the optimal policy minimizing the total discounted cost in (3) is a threshold policy. In the next section we extend the result to the average cost case. We begin (Section IV-A) by presenting some lemmas and corollaries concerning properties of the optimal value function $V_\pi^\alpha(x)$. The optimality of a threshold policy follows directly from the corollaries (Section IV-B). To save space and aid readability, proofs for the lemmas and corollaries are contained in the Appendixes.

In the discussion to follow, we will use a simplified notation for the dynamic programming equations (11a)–(11c). To do so, set

$$A_{0,n}(y) = \beta y + \alpha\lambda V_n^\alpha(y+1,0) + 2\alpha\mu V_n^\alpha(y,1)$$

so that (11a) is rewritten as

$$V_{n+1}^\alpha(y,0) = A_{0,n}(y). \tag{12a}$$

Similarly, in (11b) let $A_{1,n}(y)$ and $B_{1,n}(y)$ denote the two terms in the min bracket, respectively, and in (11c) let $A_{2,n}(y), B_{2,n}(y)$, and $C_{2,n}(y)$ denote the three terms in the min bracket, respectively, so that we can write

$$V_{n+1}^\alpha(y,1) = \min\{A_{1,n}(y), B_{1,n}(y)\} \tag{12b}$$

and

$$V_{n+1}^\alpha(y,2) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}. \tag{12c}$$

In addition, we will find it useful to define

$$\begin{aligned} A_0(y) &= \lim_{n\to\infty} A_{0,n}(y) \\ A_1(y) &= \lim_{n\to\infty} A_{1,n}(y), \quad B_1(y) = \lim_{n\to\infty} B_{1,n}(y) \\ A_2(y) &= \lim_{n\to\infty} A_{2,n}(y), \quad B_2(y) = \lim_{n\to\infty} B_{2,n}(y) \\ C_2(y) &= \lim_{n\to\infty} C_{2,n}(y) \end{aligned} \tag{13}$$

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

so that, in view of (8), the optimality (9) for states of the form $(y,0),(y,1)$, and $(y,2)$ becomes

$$V^{\alpha}_{\pi^*}(y,0) = A_0(y) \tag{14a}$$
$$V^{\alpha}_{\pi^*}(y,1) = \min\{A_1(y), B_1(y)\} \tag{14b}$$
$$V^{\alpha}_{\pi^*}(y,2) = \min\{A_2(y), B_2(y), C_2(y)\} \tag{14c}$$

We use the notation above with the understanding that each term on the right in (12) and (14) is also dependent on β, λ, C, μ, and α. For notational compactness, however, we will omit these dependencies.

A. Properties of the Value Function

We begin by presenting five lemmas concerning the properties of the optimal value function in (9). Proofs for the lemmas are contained in Appendix A.

Lemma 4.1: $V^{\alpha}_{\pi^*}(y+1,z) \geq V^{\alpha}_{\pi^*}(y,z)$ for all y,z.

This lemma states that the value function $V^{\alpha}_{\pi^*}(x)$ is nondecreasing in the queue length y. Intuitively, the greater the number of waiting passengers, the greater the rate of cost accumulation, and, therefore, the greater the cost-to-go becomes.

Lemma 4.2: $V^{\alpha}_{\pi^*}(y,0) \geq V^{\alpha}_{\pi^*}(y,1) \geq V^{\alpha}_{\pi^*}(y,2)$ for all y.

This lemma indicates that there is an ordering among the value functions, imposed by z. For a fixed queue length, the greater the number of available cars z, the smaller the cost-to-go is. This is a consequence of the fact that the queue length, and thus the rate of cost accumulation, can be reduced as more cars become available: by dispatching, the queue length and hence the rate of cost accumulation is reduced.

Lemma 4.3: $V^{\alpha}_{\pi^*}(y+C,1) \geq V^{\alpha}_{\pi^*}(y,0)$ for all $y \geq C$.

Lemma 4.4: $V^{\alpha}_{\pi^*}(y+C,2) \geq V^{\alpha}_{\pi^*}(y,1)$ for all $y \geq C$.

Lemma 4.5: $V^{\alpha}_{\pi^*}(y+2C,2) \geq V^{\alpha}_{\pi^*}(y,0)$ for all $y \geq 2C$.

These three lemmas show a relationship between the number of available cars z and the car capacity C. Consider Lemma 4.3 for example: taking the action $u=1$ from the state $(y+C,1)$ immediately puts the system in the state $(y,0)$ from which the only admissible action is to hold the car. Thus, the cost-to-go for the state $(y+C,1)$ can never be less than the cost-to-go for the state $(y,0)$, and if $u=0$ from the state $(y+C,1)$ the cost can only be greater. The other two lemmas have a similar interpretation.

A set of corollaries can be derived from the above lemmas. These corollaries serve to reveal the structure of the optimal dispatching policy. Proofs for the corollaries are contained in Appendix B.

Corollary 4.1: $A_1(y) - B_1(y)$ is strictly increasing for all $0 \leq y < C$.

Corollary 4.2: $A_1(y) > B_1(y)$ for all $y \geq C$.

These two corollaries give the structure of the optimal policy for states of the form $(y,1)$ as illustrated in Fig. 2. When the queue length is less than the car capacity, Corollary 4.1 asserts that $A_1(y) - B_1(y)$ is strictly increasing in the queue length y. From (14b), recall that the optimal control action is $u^*(y) = 0$ if $A_1(y) \leq B_1(y)$ and $u^*(y) = 1$ otherwise. These facts imply the existence of a threshold $\theta^*_{1,1}$ such that if $y \leq \theta^*_{1,1}$ then $A_1(y) - B_1(y)$ is negative and hence $u^*(y) = 0$ (Region I), and if $y > \theta^*_{1,1}$ then $A_1(y) - B_1(y)$ is positive and hence $u^*(y) = 1$ (Region II). For queue lengths greater than the car capacity, i.e., $y \geq C$, Corollary 4.2 simply asserts that $u^*(y) = 1$ (Region III), i.e., a car should always be dispatched in this case.

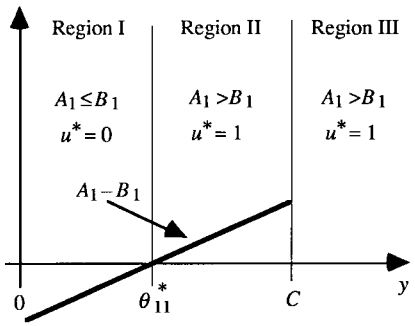

Fig. 2. Summarizing Corollaries 4.1 and 4.2.

The next five corollaries give the structure of the optimal policy for states of the form $(y,2)$.

Corollary 4.3: $C_2(y) - B_2(y)$ is a nonnegative constant independent of y for all $y \leq C$.

Corollary 4.4: $A_2(y) - B_2(y)$ is strictly increasing for all $y < C$.

Corollary 4.5: $A_2(y) > B_2(y)$ for all $y \geq C$.

Corollary 4.6: $B_2(y) - C_2(y)$ is strictly increasing for all $C < y < 2C$.

Corollary 4.7: $A_2(y) > C_2(y)$ and $B_2(y) > C_2(y)$ for all $y \geq 2C$.

The corollaries above can be interpreted with the help of Fig. 3. As was the case for states of the form $(y,1)$, Fig. 3 illustrates how states of the form $(y,2)$ also admit a threshold policy. These states, however, have two thresholds, one for dispatching one car, and another for dispatching both cars. When the queue length is less than the car capacity C, Corollary 4.3 asserts that $B_2(y) \leq C_2(y)$, i.e., $C_2(y)$ is never the smallest term in (14c). By Corollary 4.4, we also have that $A_2(y) - B_2(y)$ is increasing in y. This implies the existence of a threshold $\theta^*_{2,1}$ below which $A_2(y)$ is the smallest term in (14c) and $u^*(y) = 0$ (Region I), and above which, $B_2(y)$ is the smallest and $u^*(y) = 1$ (Region II). For queue lengths in the range $C \leq y < 2C$, we have by Corollary 4.5 that $B_2(y)$ is always smaller than $A_2(y)$, and by Corollary 4.6 that $B_2(y) - C_2(y)$ is increasing in y. This gives a threshold $\theta^*_{2,2}$ below which $B_2(y)$ is the smallest term in (14c) and $u^*(y) = 1$ (Region III), and above which $C_2(y)$ is the smallest and $u^*(y) = 2$ (Region IV). Finally, by Corollary 4.7, when the queue length exceeds twice the elevator capacity, we have $A_2(y) \geq C_2(y)$ and $B_2(y) \geq C_2(y)$, which implies that $C_2(y)$ is the smallest term, i.e., $u^*(y) = 2$ (Region V).

B. Optimality of a Threshold Policy

Based on the lemmas and their corollaries above, we are now in a position to formally state the optimality of a threshold policy for the discounted cost criterion (3).

Theorem: For $\alpha \in (0,1)$ and a bounded positive holding cost β, the optimal dispatching policy yielding the minimal

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

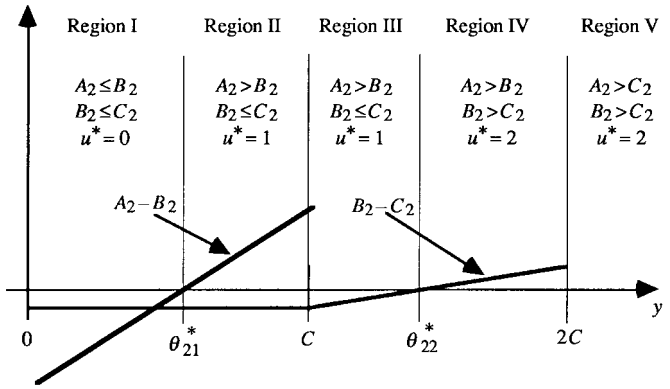

Fig. 3. Summarizing Corollaries 4.3–4.7.

total discounted cost in (3) is a threshold policy. That is, for $z = 1, \cdots, N$ and $i = 1, \cdots, z$ there exist thresholds $\theta^*_{z,i}(\beta, \lambda, \mu, \alpha)$ such that $(i-1)C < \theta^*_{z,i}(\beta, \lambda, \mu, \alpha) \le iC$ and

$$u^*(y,z) = \begin{cases} z & y \ge \theta^*_{z,z} \\ z-1 & \theta^*_{z,z-1} \le y < \theta^*_{z,z} \\ \vdots & \\ 0 & y < \theta^*_{z,1}. \end{cases} \tag{15}$$

Moreover, only N thresholds are required, since the following holds:

$$\theta^*_{z,i} = \theta^*_{z-1,i-1} + C, \qquad z = 2, \cdots, N \quad i = 2, \cdots, z. \tag{16}$$

Proof (2-car case): For the two-car case, we have $0 < \theta^*_{1,1} \le C, 0 < \theta^*_{2,1} \le C, C < \theta^*_{2,2} \le 2C$ and

$$u^*(y,1) = \begin{cases} 1 & y \ge \theta^*_{1,1} \\ 0 & y < \theta^*_{1,1} \end{cases}$$

$$u^*(y,2) = \begin{cases} 2 & y \ge \theta^*_{2,2} \\ 1 & \theta^*_{2,1} \le y < \theta^*_{2,2} \\ 0 & y < \theta^*_{2,1}. \end{cases} \tag{17}$$

Moreover,

$$\theta^*_{2,2} = \theta^*_{1,1} + C. \tag{18}$$

The proof of (17) follows directly from Corollaries 4.1–4.7. To prove (18), we note that the threshold $\theta^*_{1,1}$ is the value of y corresponding to the zero crossing of $A_1(y) - B_1(y)$ (Fig. 2), and $\theta^*_{2,2}$ is the value of y corresponding to the zero crossing of $B_2(y) - C_2(y)$ (Fig. 3). Then, comparing (11c) with (11b) we observe that $B_{2,n}(y + C) = A_{1,n}(y)$ and $C_{2,n}(y + C) = B_{1,n}(y)$, and the result follows. ♦

Although the proof above is only for the case of two cars, it should be clear at this point how to extend the result to the $N > 2$ car case. Each additional car adds an extra admissible control action and corresponding dynamic programming equation. In each dynamic programming equation there will be an additional ca term for each car added. The method of proof is exactly the same. The only complication in extending to the $N > 2$ car case is the inevitable notational burden.

The optimality of a threshold policy is an intuitively appealing result. For instance, when only one car is available and the queue length exceeds the car's capacity, it makes sense to dispatch this car immediately. It also makes sense to dispatch only one car when the number of waiting passengers is less than the elevator capacity: dispatching with both cars only partially full, when one elevator could have served the entire queue, would leave those passengers who arrive before the cars return waiting longer and accumulating costs. For such situations, the optimal policy is obvious. The intuition behind the general threshold policy lies in seeking to match the service rate to the pa rate.

Several remarks about the theorem are in order. First, throughout our analysis, we have assumed that a measure of the queue length is available, and that only one car is loaded at a time. To implement the threshold policy in practice, however, one does not need to measure the queue length. All that is needed is the number of passengers in each car, and as mentioned in Section I, many elevator systems already have on-board scales or light sensors to count the number of passengers in each car. In addition, as explained in Section II-B, the requirement that one car is loaded at a time is easily enforced using the so called "next car" feature. Next, (16) is a very attractive property of the optimal dispatching policy. In general, the total number of thresholds is given by $(1 + 2 + \cdots + N)$. As indicated by (16), however, only N of these thresholds actually need to be determined to implement the policy. When using the "next car'" feature, the most convienient set of thresholds to use is $\{\theta^*_{1,1}, \theta^*_{2,1} \cdots, \theta^*_{N,1}\}$, because these are the thresholds that tell us when to dispatch the designated "next car."

V. Extension to the Average Cost Case

In this section our objective is to find the optimal stationary policy ϕ^* minimizing the average cost criterion defined in (4). Under certain conditions, this can be done by showing that the properties of the optimal policy derived for the discounted cost criterion in (3) are retained as the discount factor $\alpha \to 1$. In particular, it is known [23, p. 289] that if the one-step costs are nonnegative, and there is an initial state $i_o \in X$ and some finite constant K such that

$$|V^\alpha_{\pi^*}(i) - V^\alpha_{\pi^*}(i_0)| \le K \tag{19}$$

for all $\alpha \in (0,1)$ and all states $i \in X$ then, 1) there is a optimal stationary policy ϕ^*; 2) the average cost for the optimal policy is given by

$$c = \overline{V}(\phi^*) = \lim_{\alpha \to 1} (1 - \alpha) V^\alpha_{\pi^*}(i_0) \tag{20}$$

and 3) there is a bounded function $h(i)$ that satisfies the following optimality equation:

$$c + h(i) = \min_{u \in U(i)} \left[C(i,u) + \sum_j P_{ij}(u) h(j) \right] \tag{21}$$

where

$$h(i) = \lim_{\alpha \to 1} [V^\alpha_{\pi^*}(i) - V^\alpha_{\pi^*}(i_0)]. \tag{22}$$

For the uppeak dispatching problem, defined in the previous sections, the one step costs (5a)–(5c) are nonnegative, and

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

Lemma 2.1 guarantees that condition (19) is satisfied. Hence, we can use the results above to obtain the optimal stationary policy for the average cost case. Equation (21) implies that $h(i)$ has the same properties as the value function $V_{\pi^*}^{\alpha}(i)$ for the discounted case. Specifically, $h(i)$ satisfies Lemmas 4.1–4.5. Moreover, the optimality equation (21), is nothing more than the dynamic programming (9) for the discounted cost case with a discount factor $\alpha = 1$ and a constant c added to the left-hand side. Since each of the corollaries, which give the structure of the optimal policy, involve taking differences, it is not hard to see that $h(i)$ also satisfies Corollaries 4.1–4.7. We conclude, therefore, that the structure of the optimal policy for the average cost case is also a threshold policy. In general, however, the thresholds for the average cost case are different than the thresholds for the discounted cost case.

VI. Examples

In this section we present several examples to illustrate our previous analysis. For the first example (Section VI-A) we numerically solve the dynamic programming algorithm for a two-car discounted cost case and generate plots analogous to Figs. 2 and 3. For the second example (Section VI-B) we simulate the two-car case and perform an exhaustive search for the optimal values of the thresholds minimizing the average passenger waiting time for two different pa rates. This example shows that the thresholds change with different pa rates, and that there can be a significant performance penalty when the thresholds are chosen arbitrarily. As a final example (Section VI-C), we solve the dynamic programming equations for a four-car discounted cost case to show how the results extend to the $N > 2$ car situation.

A. Two-Car Discounted Cost Example

In this example we numerically solve the dynamic programming algorithm (7) for the case of two identical cars each with capacity $C = 10$ passengers and service rates of $\mu = 5$ trips each 5 min, a pa rate of $\lambda = 30$ passengers each 5 min, a total event rate of $\gamma = \lambda + 2\mu = 40$, and a holding cost $\beta = 1$. To numerically solve the algorithm, we had to assume a finite lobby queue limited to 100 waiting passengers. Passengers arriving when the lobby queue is full, are turned away. Because the pa rate is relatively low, limiting the queue length to 100 passengers does not seriously affect the infinite capacity assumption.

With a discount factor of $\alpha = 0.99$, the dynamic programming algorithm is well converged after 200 iterations. Fig. 4(a) shows a plot of the "switching function" $A_1(y) - B_2(y)$ (c.f., Fig. 2) from which it can be seen that the optimal threshold for dispatching an elevator when $z = 1$ for this example is $\theta_{1,1}^* = 4$. Fig. 4(b) shows plots of $A_2(y) - B_2(y)$ and $B_2(y) - C_2(y)$ (c.f., Fig. 3) from which it can be seen that the threshold for dispatching one elevator when $z = 2$ is $\theta_{2,1}^* = 3$ and the threshold for dispatching both elevators is $\theta_{2,2}^* = 14 = \theta_{1,1}^* + C$ as expected from (16).

A point to notice is that the thresholds for dispatching one car are usually different depending on the number of available cars. In this example, as is often the case, we have $\theta_{1,1}^* > \theta_{2,1}^*$.

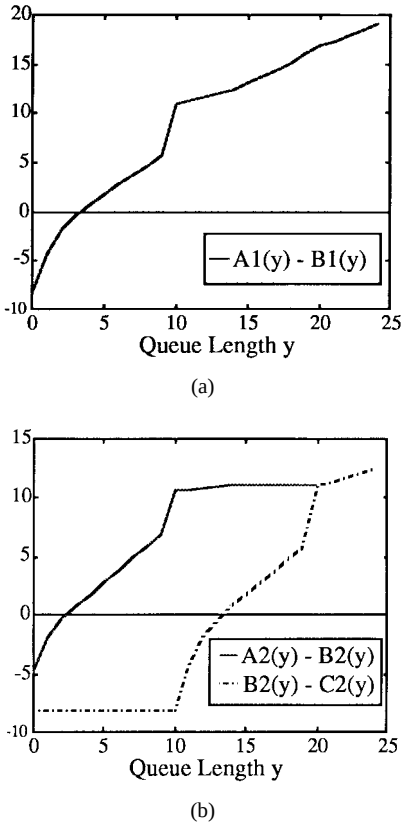

Fig. 4. Plots giving the optimal thresholds for a two car example obtained by numerically solving the dynamic programming algorithm (compare with Figs. 2 and 3, respectively).

B. Two-Car Average Cost Example

For this two-car example, brute-force simulation is used to find the optimal thresholds minimizing the average passenger waiting time. To perform the example, we developed a discrete-event simulator of a two-car elevator system. In the simulator, passengers arrive one at a time according to a Poisson process at rate λ to a finite lobby queue capable of holding 100 passengers. The time of arrival of each passenger is recorded. Each elevator has a capacity of $C = 10$ passengers. A dispatching policy with two thresholds θ_1, θ_2 is used. The dispatching policy works as follows: When one car is available $(z = 1)$ and the queue length $y \geq \theta_1$, then up to $C = 10$ passengers are immediately loaded and one elevator is dispatched; when $z = 2$ and $y \geq \theta_2$, then if $y \leq C$ one car is immediately loaded and dispatched, otherwise up to $2C = 20$ passengers are immediately loaded and both cars are dispatched. For each car, the service time is a random variable drawn from an exponential distribution with a rate of $\mu = 5$ trips each 5-min interval. Passengers load first in–first out (FIFO) into the cars. The passenger waiting time is the time interval from the passenger's time of arrival to the lobby queue up until the time the car serving the passenger is dispatched. At the beginning of each run, $y = 0$ and $z = 2$.

Response surfaces showing the average passenger waiting time for arrival rates of $\lambda = 30$ passengers each 5 min and $\lambda = 45$ passengers each 5 min are shown in Figs. 5 and 6, respectively. The response surfaces $J(\theta_1, \theta_2)$ are plots of the

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

average passenger waiting time on the z axis, the threshold θ_1 on the x axis, and the threshold θ_2 on the y axis. Each point on the response surface is obtained by averaging the passenger waiting times over ten simulation runs, with each simulation run serving 10 000 passengers (i.e., each point represents the waiting times averaged over 100 000 passengers).

From Fig. 5, the optimal thresholds minimizing the average passenger waiting time occur at $\theta_1^* = \theta_2^* = 4$. This differs from the discounted cost case in Example 1 where the thresholds for the same arrival rate were found to be $\theta_1^* = 4, \theta_2^*, = 3$. This is to be expected, because although the value functions in the average and discounted cost case have the same properties, in general, they do not have the same numerical values and the corresponding thresholds are different.

At the optimal thresholds, the average passenger waiting time is 23.61 s. The worst waiting time at this arrival rate, is 45.46 s when $\theta_1 = \theta_2 = 10$, 93% longer than the optimal wait. In Section I we mentioned two simple uppeak dispatching policies, one that dispatches as soon as one passenger enters an elevator, and another that waits for an elevator to fill to half its capacity or a 20-s timer to expire. Dispatching when the first passenger enters is equivalent to a policy with $\theta_1 = \theta_2 = 1$, which gives a waiting time of 29.15 s, 23% longer than the optimal wait. For the half-capacity plus 20-s time-out policy, we cannot exactly estimate the waiting time from Fig. 5 because it is not possible to asses the effect the 20-s time-out would have. Choosing equal thresholds $\theta_1 = \theta_2 = 5 = C/2$, however, gives a waiting time of 25.72 s, 9% longer than the optimal wait.

Fig. 6 shows that the optimal thresholds for a higher pa rate of $\lambda = 45$ passengers each 5 min is $\theta_1^* = 7, \theta_2^* = 4$. At this arrival rate, we have $\theta_1^* > \theta_2^*$ which is often the case.

At the optimal thresholds, the average waiting time for the higher arrival rate is 26.77 s. The worst waiting time is 36.58 s when $\theta_1 = 1, \theta_2 = 2$, 37% longer than the optimal wait. Comparing with the simple policy of dispatching when the first passenger enters an elevator (i.e., $\theta_1 = \theta_2 = 1$) gives a waiting time of 35.99 s, 34% longer than the optimal wait. The half capacity plus 20-s time-out (neglecting the effect of the time-out, i.e., $\theta_1 = \theta_2 = 5 = C/2$) gives a waiting time of 28.33 s, 6% longer than the optimal wait. To summarize, this example demonstrates that 1) the optimal thresholds change with the pa rate and 2) the "bowl" shape of the response curve suggests a potentially substantial penalty for choosing the wrong thresholds. This points to the need for a dispatching controller with the ability to *adapt* the thresholds to changes in the system operating conditions (i.e., the pa rate). In a follow-up paper, we will demonstrate such an adaptive dispatching controller based on the theory and the ideas of sample path constructability [6].

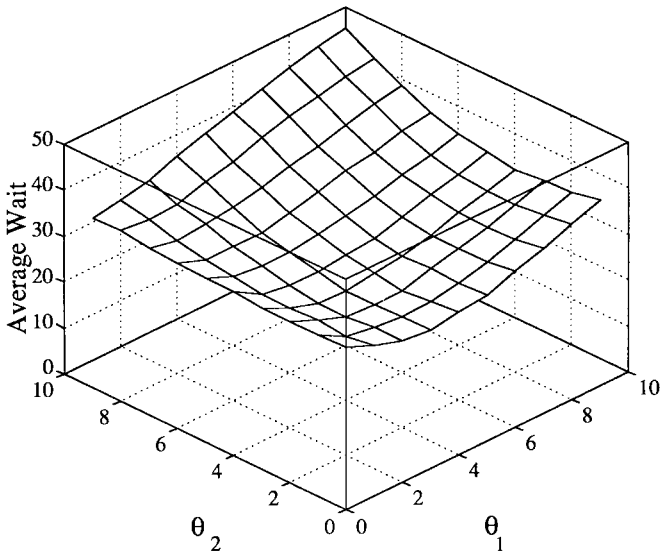

Fig. 5. Response surface for a pa rate of $\lambda = 30$ passengers each 5 min. Here, the minimum waiting time of 23.61 s is obtained with $\theta_1^* = \theta_2^* = 4$.

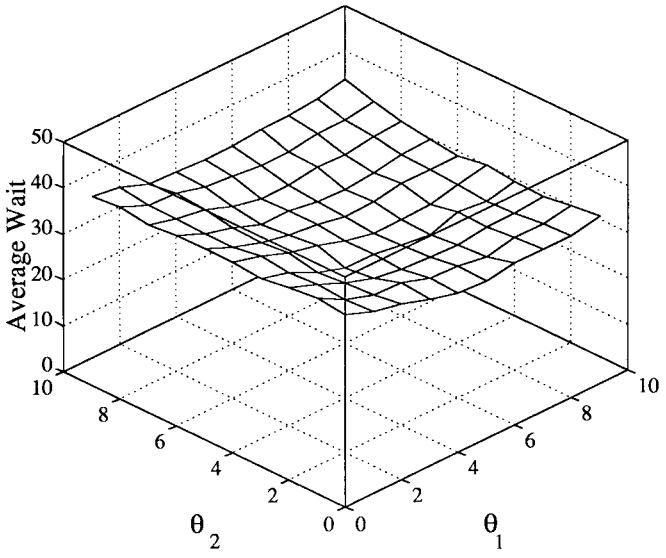

Fig. 6. Response surface for a pa rate of $\lambda = 45$ passengers each 5 min. Here, the minimum waiting time of 26.77 s is obtained with $\theta_1^* = 7, \theta_2^* = 4$.

C. Four-Car Discounted Cost Example

For this final example, we numerically solve the dynamic programming algorithm for a four-car system to show how the results developed in this paper extend to the $N > 2$ car case. In this example, each car has a capacity of $C = 10$ passengers and a service rate of $\mu = 5$ trips each 5 min, the pa rate is $\lambda = 30$ passengers each 5 min, the total event rate is $\gamma = \lambda + 4\mu = 50$, the holding cost is $\beta = 1$, and the discount factor is $\alpha = 0.99$. The reader can verify that the optimality equations for the four-car case are given by

$$\begin{aligned} V_{\pi^*}^{\alpha}(y,0) &= \beta y + \frac{\alpha\lambda}{\gamma} V_{\pi^*}^{\alpha}(y+1,0) + 4\frac{\alpha\mu}{\gamma} V_{\pi^*}^{\alpha}(y,1) \\ &= A_0(y) \\ V_{\pi^*}^{\alpha}(y,1) &= \min\left\{ \beta y + \frac{\alpha\lambda}{\gamma} V_{\pi^*}^{\alpha}(y+1,1) \right. \\ &\quad + \frac{\alpha\mu}{\gamma} V_{\pi^*}^{\alpha}(y,1) \\ &\quad \left. + 3\frac{\alpha\mu}{\gamma} V_{\pi^*}^{\alpha}(y,2), \right. \end{aligned}$$

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

$$\beta[y-C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-C]^+ + 1,) + 4\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,1)\Big\} = \min\{A_1(y), B_1(y)\}$$

$$V^\alpha_{\pi^*}(y,2) = \min\Big\{\beta y + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}(y+1,2) + 2\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}(y,2) + 2\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}(y,3),$$
$$\beta[y-C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-C]^+ + 1,1) + \frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,1) + 3\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,2),$$
$$\beta[y-2C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-2C]^+ + 1,0) + 4\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-2C]^+,1)\Big\} = \min\{A_2(y), B_2(y), C_2(y)\}$$

$$V^\alpha_{\pi^*}(y,3) = \min\Big\{\beta y + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}(y+1,3) + 3\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}(y,3) + \frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}(y,4),$$
$$\beta[y-C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-C]^+ + 1,2) + 2\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,2) + 2\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,3),$$
$$\beta[y-2C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-2C]^+ + 1,1) + \frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-2C]^+,1) + 3\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-2C]^+,2),$$
$$\beta[y-3C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-3C]^+ + 1,0) + 4\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-3C]^+,1)\Big\} = \min\{A_3(y), B_3(y), C_3(y), D_3(y)\}$$

$$V^\alpha_{\pi^*}(y,4) = \min\Big\{\beta y + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}(y+1,4) + 4\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}(y,4),$$
$$\beta[y-C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-C]^+ + 1,3) + 3\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,3) + \frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-C]^+,4),$$
$$\beta[y-2C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-2C]^+ + 1,2) + 2\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-2C]^+,2) + 2\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-2C]^+,3),$$
$$\beta[y-3C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-3C]^+ + 1,1) + \frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-3C]^+,1) + 3\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-3C]^+,2),$$
$$\beta[y-4C]^+ + \frac{\alpha\lambda}{\gamma}V^\alpha_{\pi^*}([y-4C]^+ + 1,0) + 4\frac{\alpha\mu}{\gamma}V^\alpha_{\pi^*}([y-4C]^+,1)\Big\} = \min\{A_4(y), B_4(y), C_4(y), D_4(y), E_4(y)\}$$

Fig. 7(a)–(d) show the plots of all the switching functions involved to determine the thresholds for the four-car case. Fig. 7(a) gives the threshold $\theta^*_{1,1} = 2$, Fig. 7(b) gives the thresholds $\theta^*_{2,1} = 2, \theta^*_{2,2} = 12$, Fig. 7(c) gives the thresholds $\theta^*_{3,1} = 1, \theta^*_{3,2} = 12, \theta^*_{3,3} = 22$, and Fig. 7(d) gives the thresholds $\theta^*_{4,1} = 1, \theta^*_{4,2} = 11, \theta^*_{4,3} = 22, \theta^*_{4,4} = 32$. Note that (16) is satisfied in all cases.

VII. Conclusions

In this paper we represented the uppeak elevator dispatching problem as a batch service queueing system and used dynamic programming to show that the optimal policy minimizing the discounted or average passenger waiting time is a threshold-based policy. An attractive property of the policy is that the number of thresholds that must be determined is linear in the number of cars. Also attractive is that implementation of the policy does not require knowledge of the lobby queue length. All that is required is knowledge of the number of available cars at the first floor and the number of passengers inside one car. Since most elevator systems have sensors giving the locations of the cars, and since many also have weight sensors or light sensors to estimate the number of passengers inside each car, the threshold policy can be easily implemented. Since our results do not give numerical values for the thresholds, the remaining challenge is to determine them. One way to obtain the thresholds is by solving the dynamic programming equations. Since the thresholds change as a function of several parameters including the pa rate, this would require solving many dynamic programming problems for each anticipated pa rate. Another approach is to use one of several reinforcement learning algorithms such as TD(λ) [21] or Q-learning [24]. A potential problem with these methods, however, is their slow convergence. The method we suggest for determining the thresholds is a scheme based on perturbation analysis and sample path constructability [6], [9], [11]. In a forthcoming paper, we will present such a scheme and show how it can be used to rapidly determine the optimal thresholds on-line in a model-free manner that uses only observable state information. Finally, we note that since many transportation systems can be modeled as multiple bulk server queueing systems (for example, airport shuttle busses, or the shipment of parcels or military supplies) our results can also be applied to those systems.

Appendix A
Proofs of Lemmas 4.1–4.5

This appendix contains the proofs for Lemmas 4.1–4.5. We know from Section III that the dynamic programming

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

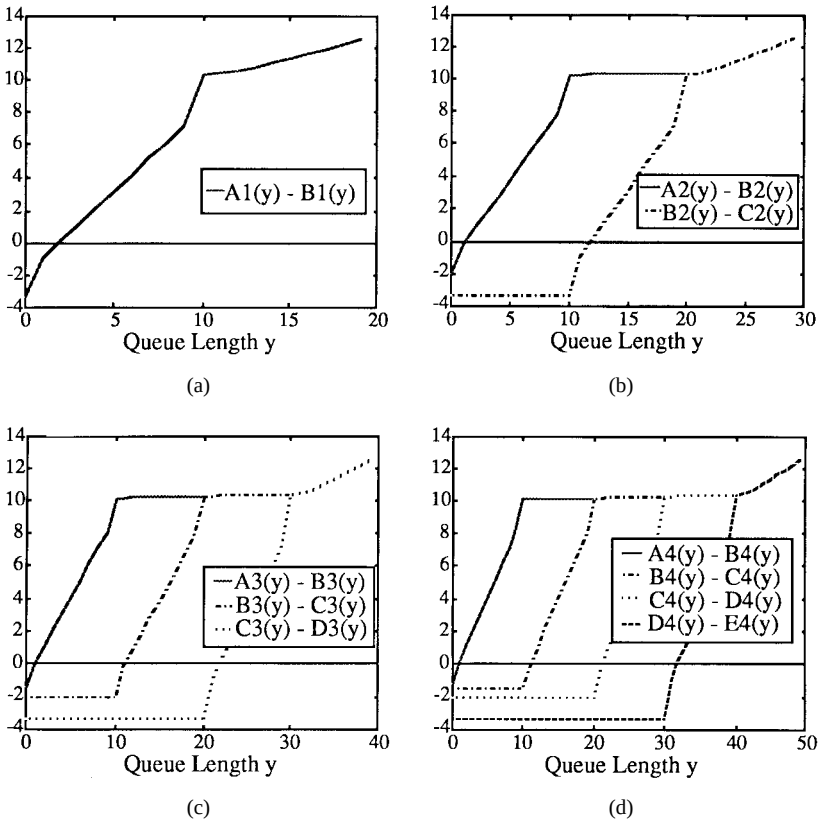

Fig. 7. Switching curves giving the optimal thresholds for a four-car example.

algorithm (7) converges to the optimal value function as $n \to \infty$. To prove the properties of the optimal value function $V^{\alpha}_{\pi^*}(x)$, therefore, it suffices to establish that $V^{\alpha}_n(x)$ satisfies the same properties for all n. This is accomplished through mathematical induction on the update step n of the dynamic programming algorithm.

Proof of Lemma 4.1: Since $V^{\alpha}_{\pi^*}(x) = \lim_{n\to\infty} V^{\alpha}_n(x)$, it suffices to show that $V^{\alpha}_n(y+1,z) \geq V^{\alpha}_n(y,z)$ for all $n = 1, 2, \cdots$ This is accomplished by induction on n.

First, for $n = 1$, since $V^{\alpha}_0(x) = 0$, we immediately have from (11a)

$$V^{\alpha}_1(y+1,0) = \beta(y+1) > \beta y = (y,0).$$

Observing that $y \geq [y-C]^+ \geq [y-2C]^+$ and that $[y+1-C]^+ \geq [y-C]^+, [y+1-2C]^+ \geq [y-2C]^+$ it follows from (11b) and (11c) that

$$V^{\alpha}_1(y+1,1) = \beta[y+1-C]^+ \leq \beta[y-C]^+ = V^{\alpha}_1(y,1)$$
$$V^{\alpha}_1(y+1,2) = \beta[y+1-2C]^+ \geq \beta[y-2C]^+ = V^{\alpha}_1(y,2).$$

Next, for some $n > 1$, the induction hypothesis is $V^{\alpha}_n(y+1,z) \leq V^{\alpha}_n(y,z)$ for all y, z and it remains to show that $V^{\alpha}_{n+1}(y+1,z) \geq V^{\alpha}_{n+1}(y,z)$ for all y, z.

1) For $z = 0$, (11a) gives

$$\begin{aligned} V^{\alpha}_{n+1}(y+1,0) &- V^{\alpha}_{n+1}(y,0) \\ &= \beta + \alpha\lambda[V^{\alpha}_n(y+2,0) - V^{\alpha}_n(y+1,0)] \\ &\quad + 2\alpha\mu[V^{\alpha}_n(y+1,1) - V^{\alpha}_n(y,1)] > 0 \end{aligned}$$

from which it immediately follows that $V^{\alpha}_{\pi^*}(y+1,0) \geq V^{\alpha}_{\pi^*}(y,0)$.

2) For $z = 1$, (11b) with the notation of (12b) gives

$$\begin{aligned} V^{\alpha}_{n+1}(y+1,1) &- V^{\alpha}_{n+1}(y,1) \\ &= \min\{A_1(y+1), B_{1,n}(y+1)\} \\ &\quad - \min\{A_{1,n}(y), B_{1,n}(y)\}. \end{aligned} \tag{A.1}$$

Using the induction hypothesis we have

$$\begin{aligned} A_{1,n}(y+1) &- A_{1,n}(y) \\ &= \beta + \alpha\lambda[V^{\alpha}_n(y+2,1) - V^{\alpha}_n(y+1,1)] \\ &\quad + \alpha\mu[V^{\alpha}_n(y+1,2) - V^{\alpha}_n(y,2)] \\ &\quad + \alpha\mu[V^{\alpha}_n(y+1,1) - V^{\alpha}_n(y,1)] > 0. \end{aligned} \tag{A.2}$$

Recalling that $[y+1-C]^+ \geq [y-C]^+$ we also get

$$\begin{aligned} B_{1,n}(y+1) &- B_{1,n}(y) \\ &= \beta[y+1-C]^+ - \beta[y-C]^+ \\ &\quad + \alpha\lambda[V^{\alpha}_n([y+1-C]^+ + 1, 0) \\ &\quad - V^{\alpha}_n([y-C]^+ + 1, 0)] \\ &\quad + 2\alpha\mu[V^{\alpha}_n([y+1-C]^+, 1) \\ &\quad - V^{\alpha}_n([y-C]^+, 1)] \geq 0. \end{aligned} \tag{A.3}$$

We will make use of (A.2)–(A.3) to show that (A.1) is nonnegative (i.e., $V^{\alpha}_{n+1}(y+1,1) \geq V^{\alpha}_{n+1}(y,1)$). There are four possible cases.

1) $A_{1,n}(y+1) \leq B_{1,n}(y+1)$ and $A_{1,n}(y) \leq B_{1,n}(y)$. From (A.2) it follows immediately that (A.1) is nonnegative, i.e., $V^{\alpha}_{n+1}(y+1,1) - V^{\alpha}_{n+1}(y,1) = A_{1,n}(y+1) - A_{1,n}(y) \geq 0$.

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

2) $A_{1,n}(y+1) > B_{1,n}(y+1)$ and $A_{1,n}(y) > B_{1,n}(y)$. From (A.3) it follows that (A.1) is nonnegative, i.e., $V^{\alpha}_{n+1}(y+1,1) - V^{\alpha}_{n+1}(y,1) = B_{1,n}(y+1) - B_{1,n}(y) \geq 0$.

3) $A_{1,n}(y+1) \leq B_{1,n}(y+1)$ and $A_{1,n}(y) > B_{1,n}(y)$. In this case

$$V^{\alpha}_{n+1}(y+1,1) - V^{\alpha}_{n+1}(y,1) = A_{1,n}(y+1) - B_{1,n}(y) \geq A_{1,n}(y) - B_{1,n}(y) \geq 0$$

where the first inequality follows from (A.2) and the second holds by assumption.

4) $A_{1,n}(y+1) > B_{1,n}(y+1)$ and $A_{1,n}(y) \leq B_{1,n}(y)$. In this case

$$V^{\alpha}_{n+1}(y+1,1) - V^{\alpha}_{n+1}(y,1) = B_{1,n}(y+1) - A_{1,n}(y) \geq B_{1,n}(y) - A_{1,n}(y) \geq 0$$

where the first inequality follows from (A.3) and the second holds by assumption.

Given 1)–4) above, it follows, as in part 1), that, $V^{\alpha}_{\pi^*}(y+1,1) \geq V^{\alpha}_{\pi^*}(y,1)$ for all y.

5) Finally, for $z=2$, (11c) with the notation of (12c) gives

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\} - \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}. \quad \text{(A.4)}$$

Using the induction hypothesis we have

$$A_{2,n}(y+1) - A_{2,n}(y) = \beta + \alpha\lambda[V^{\alpha}_n(y+2,2) - V^{\alpha}_n(y+1,2)] + 2\alpha\mu[V^{\alpha}_n(y+1,2) - V^{\alpha}_n(y,2)] > 0. \quad \text{(A.5)}$$

Recalling that $[y+1-C]^+ \geq [y-C]^+$ and $[y+1-2C]^+ \geq [y-2C]^+$ we also get

$$\begin{aligned} B_{2,n}(y+1) - B_{2,n}(y) &= \beta[y+1-C]^+ - \beta[y-C]^+ \\ &\quad + \alpha\lambda[V^{\alpha}_n([y+1-C]^+ + 1, 1) \\ &\quad - V^{\alpha}_n([y-C]^+ + 1, 1)] \\ &\quad + \alpha\mu[V^{\alpha}_n([y+1, C]^+, 2) - V^{\alpha}_n([y-C]^+, 2)] \\ &\quad + \alpha\mu[(V^{\alpha}_n([y+1-C]^+, 1) - V^{\alpha}_n([y-C]^+, 1)] \\ &\geq 0 \end{aligned} \quad \text{(A.6)}$$

and

$$\begin{aligned} C_{2,n}(y+1) - C_{2,n}(y) &= \beta[y+1-2C]^+ - \beta[y-2C]^+ \\ &\quad + \alpha\lambda[V^{\alpha}_n([y+1-2C]^+ + 1 \\ &\quad - V^{\alpha}_n([y-2C]^+ + 1, 0)] + 2\alpha\mu[V^{\alpha}_n([y+1-2C]^+, 1) \\ &\quad - V^{\alpha}_n([y-2C]^+, 1)] \geq 0. \end{aligned} \quad \text{(A.7)}$$

Given (A.5)–(A.7), we proceed as in part 2) to show that the expression (A.4) is nonnegative for all nine possible cases.

1) $A_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $A_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. It follows immediately from (A.5) that (A.4) is nonnegative.

2) $B_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $B_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. It follows immediately from (A.6) that (A.4) is nonnegative.

3) $C_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $C_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. It follows immediately from (A.7) that (A.4) is nonnegative.

4) $A_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $B_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. In this case, using (A.5) we get

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = A_{2,n}(y+1) - B_{2,n}(y) \geq A_{2,n}(y) - B_{2,n}(y) \geq 0.$$

5) $A_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $C_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. Proceeding exactly as in 4) we get

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = A_{2,n}(y+1) - C_{2,n}(y) \geq A_{2,n}(y) - C_{2,n}(y) \geq 0.$$

6) $B_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $A_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. In this case, using (A.6) we get

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = B_{2,n}(y+1) - A_{2,n}(y) \geq B_{2,n}(y) - A_{2,n}(y) \geq 0.$$

7) $B_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $C_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. Proceeding exactly as in 6) we get

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = B_{2,n}(y+1) - C_{2,n}(y) \geq B_{2,n}(y) - C_{2,n}(y) \geq 0.$$

8) $C_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $A_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}, C_{2,n}(y)\}$. In this case, using (A.7) we get

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = C_{2,n}(y+1) - A_{2,n}(y) \geq C_{2,n}(y) - A_{2,n}(y) \geq 0.$$

9) $C_{2,n}(y+1) = \min\{A_{2,n}(y+1), B_{2,n}(y+1), C_{2,n}(y+1)\}$ and $B_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}$. Proceeding exactly as in 8) we get

$$V^{\alpha}_{n+1}(y+1,2) - V^{\alpha}_{n+1}(y,2) = C_{2,n}(y,2) - B_{2,n}(y) \geq C_{2,n}(y) - B_{2,n}(y) \geq 0.$$

Given 1)–9) above it follows that $V^{\alpha}_{\pi^*}(y+1,2) \geq V^{\alpha}_{\pi^*}(y,2)$ for all y. Combining parts 1)–3) completes the proof of the lemma. ♦

Proof of Lemma 4.2: As in Lemma 4.1, it suffices to show that $V^{\alpha}_n(y,0) \geq V^{\alpha}_n(y,1) \geq V^{\alpha}_n(y,2)$ for all $n = 1, 2, \cdots$. This is accomplished by induction on n.

First, for $n=1$, since $V^{\alpha}_0(x) = 0$, and $y \geq [y-C]^+ \geq [y-2C]^+$, combining (11a)–(11c) we immediately get $V^{\alpha}_1(y,0) = \beta y \geq \beta[y-C]^+ = V^{\alpha}_1(y,1) \geq \beta[y-2C]^+ = V^{\alpha}_1(y,2)$.

For $n>1$, the induction hypothesis is $V^{\alpha}_n(y,0) \geq V^{\alpha}_n(y,1) \geq V^{\alpha}_n(y,2)$ for all y, and it remains to show that $V^{\alpha}_{n+1}(y,0) \geq V^{\alpha}_{n+1}(y,1) \geq V^{\alpha}_{n+1}(y,2)$ for all y.

1) We begin by comparing the value functions for the states $(y,0)$ and $(y,1)$ using the notation of (12a) and (12b)

$$V^{\alpha}_{n+1}(y,0) - V^{\alpha}_{n+1}(y,1) = A_{0,n}(y) - \min\{A_{1,n}(y), B_{1,n}(y)\}. \quad \text{(A.8)}$$

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

From (11a), (11b), and the induction hypothesis we get

$$A_{0,n}(y) - A_{1,n}(y) = \alpha\lambda[V_n^\alpha(y+1,0) - V_n^\alpha(y+1,1)] + \alpha\mu[V_n^\alpha(y,1) - V_n^\alpha(y,2)] \geq 0$$

from which it follows that:

$$A_{0,n}(y) - \min\{A_{1,n}(y), B_{1,n}(y)\} \geq A_{0,n}(y) - A_{1,n}(y) \geq 0$$

which establishes the fact that (A.8) is nonnegative.

2) Next we compare the value functions for the states $(y,1)$ and $(y,2)$

$$V_{n+1}^\alpha(y-1) - V_{n+1}^\alpha(y,2) = \min\{A_{1,n}(y), B_{1,n}(y)\} - \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}. \quad \text{(A.9)}$$

As in part 1), it is straightforward to show using (11b) and (11c) that

$$A_{1,n}(y) - A_{2,n}(y) \geq 0, \quad B_{1,n}(y) - B_{2,n}(y) \geq 0.$$

Then, we can show that (A.9) is nonnegative for all six possible cases that can arise. This is done exactly as in Lemma 4.1 except for the two cases where

$$C_{2,n}(y) = \min\{A_{2,n}(y), B_{2,n}(y), C_{2,n}(y)\}.$$

For the case where $A_{1,n}(y) = \min\{A_{1,n}(y), B_{1,n}(y)\}$ we have

$$V_{n+1}^\alpha(y,1) - V_{n+1}^\alpha(y,2) = A_{1,n}(y,2) - C_{2,n}(y) \geq A_{2,n}(y) - C_{2,n}(y) \geq 0$$

and for the case where $B_{1,n}(y) = \min\{A_{1,n}(y), B_{1,n}(y)\}$ we have

$$V_{n+1}^\alpha(y,1) - V_{n+1}^\alpha(y,2) = B_{1,n}(y) - C_{2,n}(y) \geq B_{2,n}(y) - C_{2,n}(y) \geq 0.$$

Combining parts 1) and 2) completes the proof. ♦

Proof of Lemma 4.3: As in the previous lemmas, we will use induction on n to establish that $V_n^\alpha(y+C,1) \geq V_n^\alpha(y,0)$ for all $y \geq C$ and all $n = 1,2,\cdots$.

For $n = 1$, since $V_0^\alpha(x) = 0$, from (11a) and (11b) we immediately get $V_1^\alpha(y+C,1) = \beta y = V_1^\alpha(y,0)$.

For $n > 1$, the induction hypothesis is $V_n^\alpha(y+C,1) \geq V_n^\alpha(y,0)$ for all $y \geq C$, and it remains to show that $V_{n+1}^\alpha(y+C,1) - V_{n+1}^\alpha(y,0) \geq 0$ for all $y \geq C$.

In the notation of (12a)–(12b), we have

$$V_{n+1}^\alpha(y+C,1) - V_{n+1}^\alpha(y,0) = \min\{A_{1,n}(y+C), B_{1,n}(y+C)\} - A_{0,n}(y). \quad \text{(A.10)}$$

From (11a) and (11b), we get

$$A_{1,n}(y+C) - A_{0,n}(y) = \beta C + \alpha\lambda[V_n^\alpha(y+C+1,1) - V_n^\alpha(y+1,0)] + \alpha\mu[V_n^\alpha(y+C,2) - V_n^\alpha(y,1)] + \alpha\mu[V_n^\alpha(y+C,1) - V_n^\alpha(y,1)] > 0$$

where we have used the induction hypothesis and the monotonicity in y of $V_n^\alpha(y,z)$ established in Lemma 4.1. Similarly

$$B_{1,n}(y+C) - A_{0,n}(y) = \alpha\lambda[V_n^\alpha(y+1,0) - V_n^\alpha(y+1,0)] + 2\alpha\mu[V_n^\alpha(y,1) - V_n^\alpha(y,1)] = 0.$$

The last two relationships immediately yield the fact that (A.10) is nonnegative and the proof is complete. ♦

Proof of Lemma 4.4: The proof is similar to that of Lemma 4.3, i.e., we proceed by induction to show that $V_n^\alpha(y+C,2) \geq V_n^\alpha(y,1)$ for all $y \geq C$ for all $n = 1,2,\cdots$.

First, for $n = 1$, since $V_0^\alpha(x) = 0$, from (9b)–(9c) we get $V_1^\alpha(y+C,2) = \beta[y-C]^+ = V_1^\alpha(y,1)$.

For $n > 1$, the induction hypothesis is $V_n^\alpha(y+C,2) \geq V_n^\alpha(y,1)$ for all $y \geq C$, and it remains to show that

$$V_{n+1}^\alpha(y+C,2) \geq V_{n+1}^\alpha(y,1) \quad \forall y \geq C.$$

Using the notation of (12b)–(12c), we have

$$V_n^\alpha(y+C,2) - V_n^\alpha(y,1) = \min\{A_{2,n}(y+C), B_{2,n}(y+C), C_{2,n}(y+C)\} - \min\{A_{1,n}(y), B_{1,n}(y)\}. \quad \text{(A.11)}$$

Using (11b) and (11c) it is straightforward to show using arguments similar to those in the previous lemmas that

$$A_{2,n}(y+C) - A_{1,n}(y) \geq 0, \quad B_{2,n}(y+C) - B_{1,n}(y) \geq 0.$$

It now remains to check that (A.11) is nonnegative for all six possible cases that can arise. All cases are straightforward to check, except for the one where

$$C_{2,n}(y+C) = \min\{A_{2,n}(y+C), B_{2,n}(y+C), C_{2,n}(y+C)\}. \quad \text{(A.12)}$$

Comparing (11c) with (11b) we observe that $C_{2,n}(y+C) = B_{1,n}(y)$ and $B_{2,n}(y+C) = A_{1,n}(y)$. When (A.12) holds, we must have $C_{2,n}(y+C) \leq B_{2,n}(y+C)$ which implies that $B_{1,n}(y) \leq A_{1,n}(y)$ and (A.11) becomes

$$V_n^\alpha(y+C,2) - V_n^\alpha(y,1) = C_{2,n}(y+C) - B_{1,n}(y) = 0$$

and the proof is complete. ♦

Proof of Lemma 4.5: The result follows directly from the previous two lemmas: For $y \geq 2C$ we have

$$V_{\pi^*}^\alpha(y+2C,2) \geq V_{\pi^*}^\alpha(y+C,1) \geq V_{\pi^*}^\alpha(y,0). \quad ♦$$

APPENDIX B
PROOFS FOR COROLLARIES 4.1–4.7

This appendix contains the proofs for Corollaries 4.1–4.7. The corollaries use the notation in (13) and follow from the dynamic programming equations (11) and the lemmas proved in Appendix A.

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

Proof of Corollary 4.1: For $0 \leq y < C$ we have $[y - C]^+ = 0$ and $[y + 1 - C]^+ = 0$. Let

$$\Delta_1(y) = [A_1(y+1) - B_1(y+1)] - [A_1(y) - B_1(y)].$$

It then suffices to show that $\Delta_1(y) \geq 0$ for $0 \leq y < C$. From the definitions of $A_1(y), B_1(y)$ [see (11)–(13)] we get

$$\begin{aligned}\Delta_1(y) = &\beta + \alpha\lambda[V^\alpha_{\pi^*}(y+2,1) - V^\alpha_{\pi^*}(y+1,1) + V^\alpha_{\pi^*}(1,0) \\ &- V^\alpha_{\pi^*}(1,0)] + \alpha\mu[V^\alpha_{\pi^*}(y+1,2) \\ &- V^\alpha_{\pi^*}(y,2) + V^\alpha_{\pi^*}(0,1) - V^\alpha_{\pi^*}(0,1)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(y+1,1) - V^\alpha_{\pi^*}(y,1) + V^\alpha_{\pi^*}(0,1) \\ &- V^\alpha_{\pi^*}(0,1)] > 0\end{aligned}$$

where the inequality is established by using Lemma 4.1. ♦

Proof of Corollary 4.2: Since $[y - C]^+ = y - C$ for $y \geq C$, the difference $A_1(y) - B_1(y)$ becomes

$$\begin{aligned}&A_1(y) - B_1(y) \\ &\quad = \beta C + \alpha\lambda[V^\alpha_{\pi^*}(y+1,1) - V^\alpha_{\pi^*}(y - C + 1, 0)] \\ &\qquad + \alpha\mu[V^\alpha_{\pi^*}(y,2) - V^\alpha_{\pi^*}(y - C, 1)] \\ &\qquad + \alpha\mu[V^\alpha_{\pi^*}(y,1) - V^\alpha_{\pi^*}(y - C, 1)] > 0\end{aligned}$$

where the inequality is established by applying Lemmas 4.1, 4.3, and 4.4. ♦

Proof of Corollary 4.3: Recalling that $[y - C]^+ = 0$ and $[y - 2C]^+ = 0$ when $y \leq C$, taking the difference, and using Lemma 4.2 gives

$$\begin{aligned}C_2(y) - B_2(y) = &\alpha\lambda[V^\alpha_{\pi^*}(1,0) - V^\alpha_{\pi^*}(1,1)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(0,1) - V^\alpha_{\pi^*}(0,2)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(0,1) - V^\alpha_{\pi^*}(0,1)] \geq 0\end{aligned}$$

which is clearly a nonnegative constant. ♦

Proof of Corollary 4.4: Again $[y - C]^+ = 0$ when $y < C$. If we define $\Delta_2(y) = [A_2(y+1) - B_2(y+1)] - [A_2(y) - B_2(y)]$ then

$$\begin{aligned}\Delta_2(y) = &\beta + \alpha\lambda[V^\alpha_{\pi^*}(y+2,2) - V^\alpha_{\pi^*}(y+1,2)] \\ &+ 2\alpha\mu[V^\alpha_{\pi^*}(y+1,2) - V^\alpha_{\pi^*}(y,2)] \\ &+ \alpha\lambda[V^\alpha_{\pi^*}(1,1) - V^\alpha_{\pi^*}(1,1)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(0,2) - V^\alpha_{\pi^*}(0,2)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(0,1) - V^\alpha_{\pi^*}(0,1)] > 0\end{aligned}$$

where the inequality is established by applying Lemma 4.1. ♦

Proof of Corollary 4.5: Since $[y + 1 - C]^+ = y + 1 - C$ and $[y - C]^+ = y - C$ for $y \geq C$ we can take the difference and use Lemmas 4.1 and 4.4 to establish the inequality

$$\begin{aligned}&A_2(y) - B_2(y) \\ &\quad = \beta C + \alpha\lambda[V^\alpha_{\pi^*}(y+1,2) - V^\alpha_{\pi^*}(y - C + 1, 1)] \\ &\qquad + \alpha\mu[V^\alpha_{\pi^*}(y,2) - V^\alpha_{\pi^*}(y - C, 2)] \\ &\qquad + \alpha\mu[V^\alpha_{\pi^*}(y,2) - V^\alpha_{\pi^*}(y - C, 1)] > 0.\end{aligned}$$

♦

Proof of Corollary 4.6: Recall $[y - C]^+ = y - C, [y + 1 - C]^+ = y + 1 - C$ and $[y - 2C]^+ = 0, [y + 1 - 2C]^+ = 0$ for $C < y < 2C$. If we define $\Delta_2(y) = [\beta_2(y+1) - C_2(y+1)] - [B_2(y) - C_2(y)]$ then, using Lemma 4.1 we get

$$\begin{aligned}\Delta_2(y) = &\beta + \alpha\lambda[V^\alpha_{\pi^*}(y+2-C,1) - V^\alpha_{\pi^*}(y+1-C,1)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(y+1-C,2) - V^\alpha_{\pi^*}(y-C,2)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(y+1-C,1) - V^\alpha_{\pi^*}(y-C,1)] \\ &+ \alpha\mu[V^\alpha_{\pi^*}(1,0) - V^\alpha_{\pi^*}(1,0)] \\ &+ 2\alpha\mu[V^\alpha_{\pi^*}(0,1) - V^\alpha_{\pi^*}(0,1)] > 0.\end{aligned}$$

♦

Proof of Corollary 4.7: Since $[y - C]^+ = y - C$ and $[y - 2C]^+ = y - 2C$ for $y \geq 2C$ we get

$$\begin{aligned}&A_2(y) - C_2(y) \\ &\quad = 2\beta C + \alpha\lambda[V^\alpha_{\pi^*}(y+1,2) - V^\alpha_{\pi^*}(y - 2C + 1, 0)] \\ &\qquad + 2\alpha\mu[V^\alpha_{\pi^*}(y,2) - V^\alpha_{\pi^*}(y - 2C, 1)] > 0\end{aligned}$$

where the inequality is established by applying Lemmas 4.4, and 4.5. Similarly, using Lemmas 4.1, 4.3, and 4.4 gives

$$\begin{aligned}&B_2(y) - C_2(y) \\ &\quad = \beta C + \alpha\lambda[V^\alpha_{\pi^*}(y - C + 1, 1) - V^\alpha_{\pi^*}(y - 2C + 1, 0)] \\ &\qquad + \alpha\mu(V^\alpha_{\pi^*}(y - C, 2) - V^\alpha_{\pi^*}(y - 2C, 1)] \\ &\qquad + \alpha\mu[V^\alpha_{\pi^*}(y - C, 1) - V^\alpha_{\pi^*}(y - 2C, 1)] > 0.\end{aligned}$$

♦

REFERENCES

[1] N. A. Alexandris, G. C. Barney, and C. J. Harris, "Multicar lift system analysis and design," *Appl. Math. Modeling*, vol. 3, pp. 269–274, Aug. 1979.

[2] H. Aoki and K. Sasaki, "Group supervisory control system assisted by artificial intelligence," *Elevator World*, pp. 70–80, Feb. 1990.

[3] G. Bao, C. G. Cassandras, T. E. Djaferis, A. D. Gandhi, and D. P. Looze, "Elevator dispatchers for down-peak traffic," Elec. and Comp. Eng. Dept., Univ. Mass., Amherst, Tech. Rep., 1994.

[4] G. C. Barney and S. M. dos Santos, *Elevator Traffic Analysis Design and Control*, 2nd ed. London: Peter Peregrinus, 1985.

[5] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[6] C. G. Cassandras, *Discrete-Event Systems: Modeling and Performance Analysis*. Boston, MA: R. D. Irwin, Inc., and Aksen Asso., Inc., 1993.

[7] R. H. Crites and A. G. Barto, "Improving elevator performance using reinforcement learning," *Neural Information Processing Society* (NIPS-8), 1995, submitted.

[8] R. K. Deb and R. F. Serfozo, "Optimal control of batch service queues," *Adv. Appl. Prob.*, vol. 5, pp. 340–361, 1973.

[9] *Discrete-Event Dynamic Systems: Theory and Applications*, Special issue on parallel simulation and optimization of discrete-event systems, vol. 5, no. 2/3, Apr./June 1995.

[10] A. D. Gandhi and C. G. Cassandras, "Optimal control of polling models for transportation applications," *J. Math. Computer Modeling*, vol. 23, pp. 1–23, 1996.

[11] Y. C. Ho and X. R. Cao, *Perturbation Analysis of Discrete-Event Dynamic Systems*. Boston, MA: Kluwer, 1991.

[12] G. T. Hummet, T. D. Moser, and B. A. Powell, "Real-time simulation of elevators," in *Winter Simulation Conf.*, Miami Beach, FL, Dec. 4–6, 1978, pp. 393–402.

[13] N. Kameli and K. Thangavelu, "Intelligent elevator dispatching systems," *AI Expert*, Sept. 1989, pp. 32–37.

[14] C. B. Kim, K. A. Seong, H. Lee-Kwang, J. O. Kim, and Y. B. Lim, "A fuzzy approach to elevator group control system," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 985–990, 1995.

[15] D. Levy, M. Yadin, and A. Alexandrovitz, "Optimal control of elevators," *Int. J. Syst. Sci.*, vol. 8, no. 3, pp. 301–320, 1977.

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.

[16] J. Lewis, "A dynamic load balancing approach to the control of multi-server polling systems with applications to elevator system dispatching," Ph.D. dissertation, Dept. Elec. Comp. Eng., Univ. Mass., Amherst, 1991.
[17] D. L. Pepyne, D. P. Looze, C. G. Cassandras, and T. E. Djaferis, "Application of Q-learning to elevator dispatching," in *Proc. IFAC'96 World Congr.*, 1996.
[18] S. Ross, *Introduction to Stochastic Dynamic Programming*. New York: Academic, 1983.
[19] M. L. Siikonen, "Elevator traffic simulation," *Simulation*, vol. 61, no. 4, pp. 257–267, 1993.
[20] G. R. Strakosch, *Vertical Transportation: Elevators and Escalators*. New York: Wiley, 1983.
[21] R. Sutton, "Learning to predict by the method of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.
[22] S. Tsuji, M. Amano, and S. Hikita, "Application of the expert system to elevator group supervisory control," in *IEEE Proc. 5th Conf. Artificial Intell. Applicat.*, 1989, pp. 287–294.
[23] J. Walrand, *An Introduction to Queueing Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[24] C. J. C. H. Watkins, "Learning from delayed rewards," Doctoral dissertation, Psych. Dept., Cambridge Univ., U.K., 1989.

David L. Pepyne received the B.S.E.E. degree from the University of Hartford, CT, in 1986. In 1991, he entered the University of Massachusetts, Amherst, where he received the M.S.E.C.E. degree in 1995, and is currently completing the Ph.D. degree.

From 1986 to 1990, he was a Flight Test Engineer with the U.S. Air Force at Edwards A.F.B., CA. From 1995 to 1996, he was a Staff Engineer with Alphatech, Inc., Burlington, MA. His research interests include the control of discrete-event systems, optimization methods, optimal control, intelligent control, and learning control.

Christos G. Cassandras (S'82–M'82–SM'91–F'96) received the B.S. degree from Yale University, New Haven, CT, in 1977, the M.S.E.E. degree from Stanford University, CA, in 1978, and the S.M. and Ph.D. degrees in 1979 and 1982, respectively, from Harvard University, Cambridge, MA.

In 1982–1984, he was with ITP Boston, Inc., where he worked on control systems for computer-integrated manufacturing. He is currently Professor of Manufacturing Engineering and Professor of Electrical and Computer Engineering at Boston University, MA. He is the author of more than 100 technical publications in these areas, including a textbook. His research interests include discrete-event systems, stochastic optimization, computer simulation, and performance evaluation and control of computer networks and manufacturing systems.

Dr. Cassandras is on the Board of Governors of the Control Systems Society, and Editor, Technical Notes and Correspondence, of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL. He serves on several other editorial boards and has guest edited for various journals. He was awarded a Lilly Fellowship in 1991. He is a member of Phi Beta Kappa and Tau Beta Pi.

Authorized licensed use limited to: BEIHANG UNIVERSITY. Downloaded on March 28,2022 at 15:20:20 UTC from IEEE Xplore. Restrictions apply.