THE ELEVATOR DISPATCHING PROBLEM: HYBRID SYSTEM MODELING AND RECEDING HORIZON CONTROL

K. S. Wesselowski, C. G. Cassandras [1]

Center for Information and Systems Engineering
Boston University

Abstract: Elevator dispatching is a combinatorially hard problem with many control constraints, time-varying traffic patterns, partial state information, and random effects. We develop a hybrid system model of a building with multiple elevator cars and apply a receding horizon control approach, bypassing some of the problem's complexities. Thus, we obtain a "universal" dispatcher which is robust with respect to changing traffic patterns and avoids the problem of having to switch among different controllers when these patterns change over the course of a day (as currently done). Moreover, simulation results show that the performance of this approach improves upon that of state-of-the-art dispatchers.
Copyright © 2006 IFAC
Keywords: Hybrid Systems, Receding Horizon Control, Elevator Dispatching

1. INTRODUCTION

In an elevator system, the "dispatcher" is a centralized, upper-level controller which determines where each car is to stop, either to load passengers (in response to "hall calls" at various floors) or to unload them (in response to "car calls"). Effective elevator dispatching must meet certain quality-of-service criteria and gives rise to a problem of great practical importance which remains open because of several difficulties.

A multi-car elevator system is a hybrid system that combines the time-driven motion dynamics of the cars with the event-driven dynamics imposed by the dispatching controller. The state space of such a system is enormous; as an example, a building with 18 floors and 6 cars has a state space size of the order of 10^{44}, which is approximately the same size as that of the game of chess. The dispatching problem is further complicated by numerous elevator operating constraints and by partial state information, e.g., the controller knows about the existence of a hall call at some floor, but not the number of passengers there.

Because of these difficulties, the problem is often decomposed into a set of archetypal passenger arrival patterns, with control algorithms customized for a specific pattern. For example, the *uppeak traffic pattern* occurs during the morning rush hour at a typical office building, when hall calls occur at the lobby only and result in car calls to all floors. Given a cost function based on specified quality-of-service criteria, the dispatching problem can, in principle, be solved through dynamic programming. In practice, however, this is only possible for very limited cases under specific modeling assumptions, such as the uppeak scenario (Pepyne and Cassandras, 1997) and the parking problem (Brand and Nikovski, 2004). Other dynamic programming approaches, e.g. (Nikovski and Brand, 2003), may not handle all traffic scenarios well and/or impose a high computational burden. To avoid the intractability of dynamic

[1] The authors' work is supported in part by NSF under grant DGE-0221680 and by AFOSR under grant FA9550-04-1-0208.

programming approaches, many heuristic algorithms have been developed, most based on the idea of dynamically assigning a car to serve a certain sector of the building, e.g., the Dynamic Load Balancing (DLB) algorithm in (Cassandras *et al.*, 1990) and other sectoring schemes, e.g., (Chan *et al.*, 1997). Efforts to use fuzzy logic to control switching between algorithms depending on the traffic patterns (Powell and Sirag, 1993), or attempts to implement various reinforcement learning techniques using neural networks (Siikonen, 1997), (Crites and Barto, 1996), have yielded limited success.

The goal of this work is to explore a new approach to elevator dispatching based on a formal hybrid system model in conjunction with a receding horizon controller. Our motivation is twofold. First, there is a continuing need to demonstrate that substantially better performance is achievable by new controllers relative to state-of-the-art heuristics. Second, with existing algorithms customized to specific traffic patterns, dispatchers are forced to switch among algorithms by detecting when it is optimal to do so. It is, therefore, desirable to have a "universal" dispatching controller which is equally effective in all situations and involves few, if any, tunable parameters. The approach we propose is characterized by this feature, while also showing (based on simulation experiments) substantial performance improvements compared to state-of-the-art dispatching algorithms. By using a hybrid model of the elevator system, our approach allows us to conveniently calculate the travel times of cars to various hall or car calls. Moreover, this model facilitates the use of the cooperative receding horizon control developed in (Li and Cassandras, 2006).

2. A HYBRID SYSTEM MODEL

Consider a multi-car elevator system with C cars, each with capacity P passengers, operating in an N-floor building, where floor 1 is the ground floor (or lobby) and floor N is the top floor. The system can be represented by a graph we will refer to as the *spine model* shown in Figure 1, where each floor is associated with three nodes: two spike nodes and one spine node. Letting $M = 2N$, the spike nodes are indexed by $i = 1, \ldots, M$ in a clockwise direction, where nodes $i = 1, \ldots, N$ on the left-hand side are associated with upward-bound passengers, and nodes $i = N+1, \ldots, M$ on the right-hand side are associated with downward-bound passengers. Nodes $i = 1, \ldots, N-1$ can be origins for upward-bound passengers, and nodes $i = 2, \ldots, N$ can be destinations for upward-bound passengers. Similarly, nodes $i = N + 1, \ldots, M-1$ and $i = N+2, \ldots, M$ can be origins and destinations, respectively, of downward-bound passengers. The spine nodes, indexed by $i = M+1, \ldots, M+N$ (in the center of the model in Figure 1) represent *decision points* for stopping at floors $1, \ldots, N$ respectively. That is, a car reaching node $M+i$ at top speed may choose to decelerate and stop at node i (if moving upwards) or node $M - i + 1$ (if moving downward); otherwise it continues moving at top speed and skips floor i.

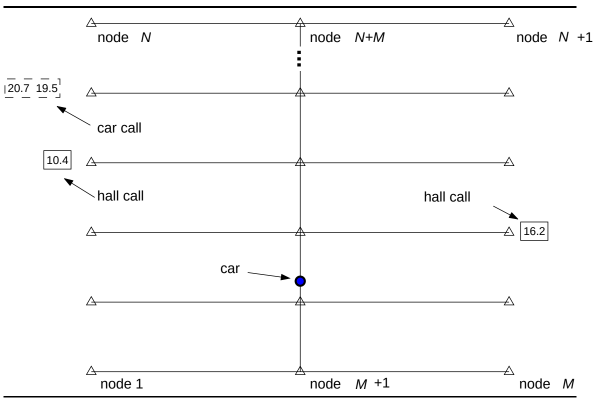

Fig. 1. The "spine model" of an elevator system.

Cars move on the arcs connecting the nodes: on vertical arcs, cars move between floors at top speed with a corresponding arc travel time D; on horizontal arcs (or "spikes"), cars decelerate as they approach a floor or accelerate away from some floor with a corresponding arc travel time d. A car located at a spike node is stopped at the corresponding floor either serving upward-bound passengers (for $i = 1, \ldots, N$) or downward-bound passengers (for $i = N + 1, \ldots, M$). We represent hall calls and car calls by numbers in boxes next to the appropriate spike node. A hall call is generated by the first passenger at a node who presses an up or down button. Its box is bordered by a solid line and lists all passenger waiting times (note that additional passengers may arrive at the same node after the initial hall call is initiated). A car call is generated when a car answers a hall call and a passenger entering this car selects a destination floor. A car call is represented by a box bordered in a dashed line shown next to the destination node and lists all times since corresponding passengers in that car issued the call. For this model, we assume that all acceleration/deceleration times (required for a car to go between a complete stop and full speed) are fixed and equal and all top speeds are fixed and equal to a value v_{max}. For simplicity, we also assume that the vertical distance travelled by a car during acceleration is less than half the vertical distance between floors.

The state of this hybrid system consists of a continuous (real-valued) component for the C cars and a discrete (non-negative integer-valued) component for all hall and car calls present. The state of car $j = 1, \ldots, C$ is a vector $[e_j, v_j]$ where e_j is the elevation of car j and $v_j = \dot{e}_j$ is its velocity. The state of hall calls in the system is described by the hall call queue lengths $h(i)$, $i = 1, \ldots, M$ and by $a^h(i, l)$, $l = 1, \ldots, h(i)$, the arrival time of each passenger waiting at the hall call at node i. The state of car calls is described by the car call queue lengths $\psi_j(i)$, $i = 1, \ldots, M$, $j = 1, \ldots, C$, and by $a^c_j(i, l)$, $l = 1, \ldots, \psi_j(i)$, the arrival times of each passenger in car j with destination node i.

The dynamics of the system depend on an *event set* E and a *set of control actions* U imposed by the dispatcher. Events in E correspond to the arrival of cars at various decision points on the spine model, along with an exogenous event for passenger arrivals at (hall-call) nodes. The type of event that occurs dictates the feasible transitions, or decisions, for each car. There are $(6C+4)(N-1)$ events in E.

Next, we define a small set of control actions, $U = \{STOP, GOUP, GODN, LOAD\}$, which may be taken upon the occurrence of certain events. For example, a control action must be taken when some upward-moving car j arrives at a spine node $i \in \{M+2, \ldots, M+N\}$ defining a decision point. The dispatcher must select: (i) $STOP$ if the car is to be routed to the corresponding spike node $i \in \{2, \ldots, M\}$, or (ii) $GOUP$ if the car is to be routed to spine node $i+1$.

A different set of commands may be feasible when a car is at a spike node. For example, when a car finishes a loading operation at a spike node i, the dispatcher must select: (i) $GOUP$ (or $GODN$) if the car is to immediately depart node $i \in \{1, \ldots, N-1\}$ (or $i \in \{N+1, \ldots, M-1\}$), or ($ii$) $STOP$ if the car is to remain at node i with the doors open to possibly accept more passengers. This construction allows for the car to idle and a "timeout" to be implemented to prevent a loaded passenger from waiting more than a set amount of time before departing.

The dispatcher may issue a command to a parked (idling) car as a result of some event that is exogenous to that car. This event may be a passenger arrival event or an event associated with some car other than the parked car. In either case, the dispatcher may issue a command to the parked car to either (i) $LOAD$ if the event occurs at the floor where the car is located, or (ii) $GOUP$ (or $GODN$) if the car is to immediately depart.

Clearly, the choice of control action at a decision point depends on the full state of the system, denoted by $\mathbf{x}$, at that time. For example, if a car arrives at a spike node i with $\psi_j(i) > 0$, then j is obligated to serve the associated car calls with destination i, therefore $STOP$ must be selected from the feasible control action set $U = \{STOP, GOUP, GODN, LOAD\}$. An $UNLOAD$ command is not necessary, because passengers must be allowed to depart at their destination.

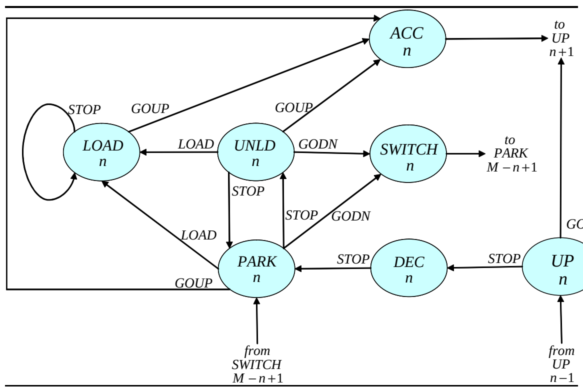

Fig. 2. State transition diagram for a car at or approaching floor $n = 2, \ldots, N-1$.

The complete state transition diagram for the hybrid automaton model of this system is extremely complex and depends on the specific dispatching control scheme used. However, we can obtain a state transition diagram for each individual car based on the observation that dispatching decisions depend only on certain discrete states defining a discrete event system abstracted from the hybrid automaton. For example, the state transition diagram shown in Figure 2 corresponds to a car moving upward at some floor $n < N$. For simplicity, the control actions are shown, but not events that may precipitate transitions.

Note that all hall call and car call states are also abstracted out of this subsystem, thus allowing for all possible control actions as defined above. A state transition diagram for the entire system would include the union of C state transition diagrams, each with N subsystems as shown in Figure 2, along with all car call and hall call states. Which of these transitions and states are feasible depends on the dispatching controller defined (see next section). As an example, consider Figure 2 for some upward-moving car j with the added information that $\psi_j(n) > 0$ and $\sum_{i=n+1}^{N} \psi_j(n) > 0$, i.e., there is a car call with destination n and there are additional car calls for higher floors. In this case, for example, transitions to the $SWITCH$ state are not feasible.

3. THE RECEDING HORIZON DISPATCHING CONTROLLER

The basic idea of the Receding Horizon Controller (RHC) used for dispatching cars in an elevator system is similar in spirit to that developed in (Li and Cassandras, 2006) for the control of coordinated teams of fully-actuated autonomous vehicles moving freely in a 2-dimensional space.

Receding horizon schemes are also associated with model-predictive control (e.g., (Mayne and Michalska, 1990)). In the case of dispatching, the optimal control problem is combinatorially intractable and further complicated by random events due to uncertain passenger arrivals and stochastic loading/unloading times. Typically, dispatching algorithms assign cars to floors as new hall calls and car calls are generated. In contrast, the RHC we propose issues commands from the set $U = \{STOP, GOUP, GODN, LOAD\}$ that are applied to a car whenever it comes to a decision point as a result of the events defined in the previous section. There is no attempt to commit a car to a floor except at the last possible point possible (i.e., when the car arrives at the spine node for that floor). This allows for taking into account unexpected events that may alter a prior car-to-floor assignment. Moreover, the control action set U is much smaller than the set of all possible car-to-floor assignments for C cars and N floors.

The RHC is invoked whenever any new event occurs in the system at some time t, at which point it solves an optimization problem using all available current state information $\mathbf{x}_t$ and a given objective function defined over the time interval $[t, t + H(\mathbf{x}_t)]$, where $H(\mathbf{x}_t)$ is called the *planning horizon.* However, the control actions determined by the RHC at t are executed only until the next system event is observed (typically much sooner than $t + H(\mathbf{x}_t)$), at which point a new optimization problem is solved with all updated state information. For vehicles moving freely in a simple two-dimensional space, as in (Li and Cassandras, 2006), the notion of Euclidean distance between two points is well-defined and simple to use. In contrast, each car of an elevator system is constrained to move in the graph topology of the spine model of Figure 1. Thus, we need to adapt the RHC from a Euclidean space to a graph.

The first step is to develop an appropriate (non-Euclidean) distance metric for a graph. Let $d(r, s)$ be the *distance* between points r, s both belonging to the same arc on the graph as the time required for a car to travel from r to s. Depending on the arc in the spine model, this is simply the time to travel from r to s at speed $v_{\max}$ (spine arcs) or subject to the acceleration/deceleration parameters given (spike arcs). If $r = s$ is a node of the graph, then $d(r, r)$ is the time spent by a car at a node, e.g., while loading/unloading. This time generally depends on both the car and the node states. This motivates a car-dependent definition of the metric for nodes, $d_j(r, r, \boldsymbol{x}_t)$, which also incorporates dependence on the system state $\mathbf{x}_t$.

The *neighborhood* of a point r on the graph, $\mathcal{N}(r)$, is the set of all nodes with arcs connecting them to r. For any two points r, s, a *path* $\mathbf{p}(r, s)$ from r to s is a sequence of nodes $\mathbf{p}(r, s) = \{n_1, \ldots, n_L\}$ such that $n_1 \in \mathcal{N}(r)$, $n_L \in \mathcal{N}(s)$, and $n_{l+1} \in \mathcal{N}(n_l)$, $l = 1, \ldots, L-1$. We can then define the distance (or length) of a path as

$$d_j(r, s, \mathbf{x}) = d_j(r, n_1, \mathbf{x}) + \sum_{l=1}^{L-1} [d_j(n_l, n_l, \mathbf{x}) + d(n_l, n_{l+1})] + d_j(n_L, s, \mathbf{x}) \quad (1)$$

Of particular interest are paths from a car j's location at time t, denoted by c_j^t, to nodes where there are hall calls or car calls to serve. We shall refer to these as "target nodes" for car j and denote the corresponding set at time t by R_j^t. For any $r \in R_j^t$, a car-to-target node path is

$$\mathbf{P}_j^t(r) \equiv \mathbf{p}(c_j^t, r), \quad r \in R_j^t \quad (2)$$

Given the path length in (1), we define the *target time* $\tau_j^t(c_j^t, r)$ as the time when car j reaches r along that path, i.e., $\tau_j(c_j^t, r) = t + d_j(c_j^t, r, \mathbf{x})$.

Finally, each path has a corresponding *path input sequence* $\boldsymbol{\mathcal{U}}_j^t(r)$, where the ith element, $u_j(r, i)$, is the ith feasible control action involved along the path $\mathbf{P}_j^t(r)$, as a result of events that would occur as car j travels on the path to target r.

Car trajectory construction under RHC. In the simple 2-dimensional space used in (Li and Cassandras, 2006), the RHC's function is to solve an optimization problem whose solution gives optimal vehicle headings at time t. Given a vehicle speed v and a planning horizon H, the set of points on a circle of radius H/v around the vehicle's current position defines its feasible *horizon points.* The RHC determines the optimal heading that leads to some corresponding optimal horizon point, $x_H^*(t)$. The optimal RHC trajectory is a straight line to $x_H^*(t)$, and the RHC trajectory's target time for a target at point y is simply $t + H + \|y - x_H^*(t)\| / v$, where $\|\cdot\|$ is the Euclidean metric.

In our spine model setting, the set of feasible horizon points has to be defined accordingly. Given a path $\mathbf{P}_j^t(r)$ as in (2), a horizon point is a point that corresponds to the car's future position on this path at time $t + H(\mathbf{x}_t)$. Formally, we define a horizon point $\omega_j(r)$ for some target node $r \in R_j^t$ as a point uniquely given by

$$d_j(c_j^t, \omega_j(r), \mathbf{x}_t) = H(\mathbf{x}_t)$$
$$d_j(c_j^t, n_k, \mathbf{x}_t) \le H(\mathbf{x}_t) \le d_j(c_j^t, n_{k+1}, \mathbf{x}_t)$$

for some $n_k \in \mathbf{P}_j^t(r)$. The horizon point $\omega_j(r)$ belongs to some arc (n_k, n_{k+1}) on the path to target node r such that the distance from c_j^t to $\omega_j(r)$ is the horizon $H(\mathbf{x}_t)$. Note that it is possible for $\omega_j(r)$ to coincide with a node in $\mathbf{P}_j^t(r)$ since cars can spend a positive amount of time at a node in the spine model. The set of all horizon points for a car j is denoted by $\Omega_j^t = \cup_{r \in R_j^t} \{\omega_j(r)\}$. We can then define a path $\mathbf{p}(\omega, r)$ from any horizon point

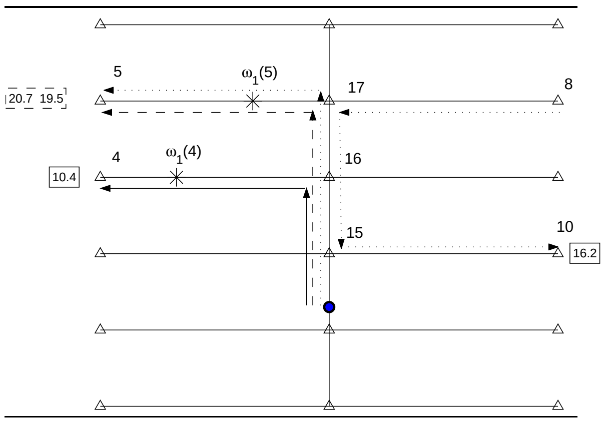

Fig. 3. Paths from a car to three target nodes, and resulting horizon points.

$\omega \in \Omega_j^t$ to any target $r \in R_j^t$ and an associated target time

$$\tau_j(\omega, r) = t + H(\mathbf{x}_t) + d_j(\omega, r, \mathbf{x}_{t+H(\mathbf{x}_t)})$$

As an example of paths from cars to target nodes, consider Figure 3, where a car is carrying passengers upward in a building with $N = 6$ floors. Since the car cannot switch directions until the car call at node 5 is served, the path to the downward-bound car call at $r = 10$ is $\mathbf{P}_1^t(10) = \{15, 16, 17, 5, 8, 17, 16, 15, 10\}$, as shown by the dotted line. Figure 3 also shows how a given planning horizon $H(\mathbf{x}_t)$ locates the horizon points $\omega_1(4)$ and $\omega_1(5) = \omega_1(10)$ in the set Ω_1^t.

The final issue we address before discussing the optimization problem that the RHC must solve is that of selecting an appropriate planning horizon $H(\mathbf{x}_t)$. Motivated by the analysis in (Li and Cassandras, 2006), we shall also choose $H(\mathbf{x}_t)$ to be the shortest distance (in time units) from any car to any target node, i.e.,

$$H(\mathbf{x}_t) = \min_{r \in R_j^t, j=1,\ldots,C} d_j(c_j^t, r, \mathbf{x}_t)$$

Intuitively, $H(\mathbf{x}_t)$ provides the earliest time at which the system workload could be reduced by any car.

RHC objective function. To construct the objective function for the RHC, we will concentrate here on minimizing the waiting times of passengers. Recall that Ω_j^t is the set of all feasible horizon points for car j at time t, and $\tau_j(\omega, r)$ is the earliest possible time that a passenger in the hall call queue of some node r can be served if car j chooses horizon point $\omega \in \Omega_j^t$. Then, the *hall-call waiting time function*, denoted by $W_j(\omega, r)$, is

$$W_j(\omega, r) = \sum_{l=1}^{h(r)} \left[\tau_j(\omega, r) - a^h(r, l)\right]$$

where r is a spike node with $h(r) > 0$ and $a^h(r, l) \le t$ is the arrival time of the lth passenger in the hall call r queue. Similarly, the *car-call system time function*, denoted by $Y_j(\omega, r)$, is

$$Y_j(\omega, r) = \sum_{l=1}^{\psi_j(r)} \left[\tau_j(\omega, r) - a_j^c(r, l)\right]$$

where r is a spike node with $\psi_j(r) > 0$ and $a^c(r, l) \le t$ is the arrival time of the lth passenger in the car call queue for destination r originating at some node previously visited by car j.

The *relative proximity function* $q_j(\boldsymbol{w}, r) \in [0, 1]$, where r is a hall call target, and $\boldsymbol{w} = [\omega_1, \ldots, \omega_C]$, $\omega_j \in \Omega_j^t$, $j = 1, \ldots, C$ is intended to engender cooperation between the cars. It was introduced by (Li and Cassandras, 2006), but here we must adapt it to our particular topology. The function $q_j(\boldsymbol{w}, r)$ is monotonically non-increasing in the *relative distance function*, denoted by $\delta_j(\boldsymbol{w}, r)$. It depends on how close ω_j is to target r, compared to all other horizon points. We order the elements of $\boldsymbol{w}$ so that $d_{j_1}(\omega_{j_1}, r, \boldsymbol{x}_{t+H(\boldsymbol{x}_t)}) \le \ldots \le d_{j_C}(\omega_{j_C}, r, \boldsymbol{x}_{t+H(\boldsymbol{x}_t)})$ and using the cars with shortest distances, j_1 and j_2, we define

$$\delta_{j_i}(\boldsymbol{w}, r) = \frac{d_{j_i}(\omega_{j_i}, r, \boldsymbol{x}_{t+H(\boldsymbol{x}_t)})}{\sum_{k=j_1,j_2} d_k(\omega_k, r, \boldsymbol{x}_{t+H(\boldsymbol{x}_t)})}$$

and $\delta_{j_i}(\boldsymbol{w}, r) = 0$ for $i = 3, \ldots, C$ if applicable. A simple and convenient choice of relative proximity function is

$$q_j(\boldsymbol{w}, r) = \begin{cases} 1 & \text{if } \delta_j \le \Delta \\ \dfrac{(1-\Delta) - \delta_j}{1 - 2\Delta} & \text{if } \Delta < \delta_j < 1 - \Delta \\ 0 & \text{if } \delta_j \ge 1 - \Delta \end{cases}$$

where for brevity we let $\delta_j \equiv \delta_j(\boldsymbol{w}, r)$, and $\Delta \in [0, 0.5)$ is a constant adjustable to quantify the extent of car cooperation.

The RHC's function is to select $\boldsymbol{w}$, i.e., a horizon point $\omega_j \in \Omega_j^t$ for each car j, so as to minimize

$$J(\boldsymbol{w}) = \sum_j \sum_{r \in R_j^t} [W_j(\omega_j, r)]^2 q_j(\boldsymbol{w}, r) + Y_j(\omega_j, r)$$

The minimizing vector of horizon points is $\boldsymbol{w}^* = \arg\min J(\boldsymbol{w})$. Each element ω_j^* of $\boldsymbol{w}^*$ is associated with a path input sequence $\boldsymbol{\mathcal{U}}_j^t(\omega_j^*)$, and the control input applied to car $j = 1, \ldots, C$ at time t is the first element $u_j(\omega_j^*, 1)$ of $\boldsymbol{\mathcal{U}}_j^t(\omega_j^*)$. In the example of Figure 3, the optimal first control action is the one associated with leading the car to either $\omega_1(4)$ or $\omega_1(5)$ from its current position, whichever minimizes $J(\boldsymbol{w})$; here it happens that $u_1(\omega_1(4), 1) = u_1(\omega_1(5), 1) = GOUP$. A simple calculation shows that the size of the decision space for the RHC optimization problem is $O\big((2 + 2\lceil d/D \rceil)^C\big)$ and independent of N.

Implementation. In the actual system, the complete state is unknown. The RHC is required to estimate the number of passengers associated with each call based on the known elapsed time since the call was issued and a known (or estimated) passenger arrival rate at each node (determining these rates may itself be a difficult problem.) In addition, if $h(i) = 0$ for some i, an anticipatory element can be added to the RHC by estimating the length of a currently empty hall call queue in the future (specifically, at time $\tau_j(\omega, i)$ for some $\omega \in \Omega_j^t$). Another complication is that cars whose passenger capacity P is reached must be prevented from responding to hall calls.

4. NUMERICAL RESULTS

The RHC and several competing controllers were implemented in a simulation environment developed using MATLAB. Here, we limit the results shown to two different traffic patterns: a two-hour lunchtime traffic scenario (about 1,500 passengers delivered), with a combination of uppeak, downpeak, and interfloor traffic, and a two-hour pure uppeak case. For both tests, each car has a capacity of $P = 16$ passengers, top speed $v_{max} = 1$ m/sec, and acceleration $a = 0.4$ m/sec^2.

For the lunchtime scenario, we compare the performance of the RHC to that of the DLB algorithm of (Cassandras *et al.*, 1990) in a simulated building with $N = 10$ and $C = 4$. Table 1 shows averages over 10 simulation runs where the RHC compares favorably to the DLB algorithm. The passenger arrivals at each floor were simulated according to a time-varying Poisson process.

Table 1. Lunchtime scenario.

Controller	Avg. Waiting Time, sec	Avg. Sys. Time, sec	Avg. Sq. Sys. Time, sec^2
RHC	17.47	44.96	2,670
DLB	20.11	52.52	3,686

A comparison of the performance of the RHC to that of an Uppeak Threshold Controller (UTC) is shown in Table 2 for a simulated building with $N = 20$ and $C = 4$, using a constant Poisson arrival rate of 5 passengers per minute at the lobby. It was shown in (Pepyne and Cassandras, 1997) that the optimal dispatching policy in the pure uppeak case (with no interfloor or downpeak component) is a threshold-based policy, assuming Markovian arrival and service processes. The UTC based on this idea uses a vector $\boldsymbol{T} = [T_1, \ldots, T_C]$, where each element T_i is the threshold (number of passengers) for dispatching a car loading at the lobby when there are i cars available at the lobby. The RHC has the advantage that it uses the location information for other cars, and is able to dispatch a car from the lobby when a returning car gets close enough to place a horizon point at node 1. When applying the RHC, an expected arrival rate $\hat{\lambda}_1$ at node 1 is used to estimate future arrivals and effectively achieve a thresholding behavior. As seen in Table 2, the UTC under $T = [1, 1, 1, 1]$ and the RHC under $\hat{\lambda}_1 = 0$ psgr/min have the same behavior since they both ignore future arrivals (of course, determining a $\hat{\lambda}_1$ that achieves the best possible performance, in this case $\hat{\lambda}_1 = 0.37$ psgr/min, remains a side estimation issue). What is important is that the RHC achieves the same uppeak performance as the UTC *without switching to a specialized operating mode*, substantiating the claim that the RHC is a robust "universal" controller.

Table 2. Uppeak scenario.

Controller	Avg. Sys. Time, sec	Avg. Sq. Sys. Time, sec^2
RHC, $\hat{\lambda}_1 = 0.00$ psgr/min	62.57	5,079
RHC, $\hat{\lambda}_1 = 0.37$	58.49	4,205
UTC, $T = [1, 1, 1, 1]$	62.57	5,079
UTC, $T = [2, 1, 1, 1]$	61.26	4,707

REFERENCES

Brand, M. and D. Nikovski (2004). Optimal parking in group elevator control. In: *Intl. Conf. on Robotics and Automation.* Vol. 1. pp. 1002–1008.

Cassandras, C. G., T. E. Djaferis, J. Lewis and D. P. Looze (1990). Dispatching through dynamic load balancing (dlb): the "noontime" scenario. Tech. Report, Dept. of ECE, U. of Massachusetts.

Chan, W. L., A. T. P. So and K. C. Lam (1997). Dynamic zoning in elevator traffic control. *Elevator World* pp. 136–139.

Crites, R. H. and A. G. Barto (1996). Improving elevator performance using reinforcement learning. In: *Advances in Neural Information Processing Systems 8* (D. S. Touretzky, M. C. Mozer and M. E. Hasselmo, Eds.). MIT Press. pp. 1017–1023.

Li, W. and C. G. Cassandras (2006). A cooperative receding horizon controller for multivehicle uncertain environments. *IEEE Trans. on Auto. Control* **51**(2), 242–257.

Mayne, D. Q. and L. Michalska (1990). Receding Horizon Control of Nonlinear Systems. *IEEE Trans. on Auto. Control* **AC-35**(7), 814–824.

Nikovski, D. and M. Brand (2003). Marginalizing out future passengers in group elevator control. In: *Proc. of the 19th Conf. on Uncertainty in Artificial Intelligence.* pp. 443–450.

Pepyne, D. and C. Cassandras (1997). Optimal dispatching control for elevator systems during uppeak traffic. In: *IEEE. Trans. on Control Systems Technology.* Vol. 5. pp. 629–642.

Powell, B. A. and D. J. Sirag (1993). A new way of thinking about the complexities of dispatching elevators. *Elevator World* pp. 78–84.

Siikonen, M.-L. (1997). Elevator group control with artificial intelligence. Report A67, Systems Analysis Lab., Helsinki U. of Tech.