Group Elevator Scheduling With Advance Information for Normal and Emergency Modes

Peter B. Luh, *Fellow, IEEE*, Bo Xiong, and Shi-Chung Chang, *Member, IEEE*

***Abstract*—Group elevator scheduling has long been recognized as an important problem for building transportation efficiency, since unsatisfactory elevator service is one of the major complaints of building tenants. It now has a new significance driven by homeland security concerns. The problem, however, is difficult because of complicated elevator dynamics, uncertain traffic in various patterns, and the combinatorial nature of discrete optimization. With the advent of technologies, one important trend is to use advance information collected from devices such as destination entry, radio frequency identification, and sensor networks to reduce uncertainties and improve efficiency. How to effectively utilize such information remains an open and challenging issue. This paper presents the optimized scheduling of a group of elevators with destination entry and future traffic information for normal operations and coordinated emergency evacuation. Key problem characteristics are abstracted to establish a two-level separable formulation. A decomposition and coordination approach is then developed, where subproblems are solved by ordinal optimization-based local search, and top ranked nodes are selectively optimized by using dynamic programming. The approach is then extended to handle up-peak with little or no future traffic information, elevator parking for low intensity traffic, and coordinated emergency evacuation. Numerical testing results demonstrate near-optimal solution quality, computational efficiency, the value of future traffic information, and the potential of using elevators for emergency evacuation.**

***Note to Practitioners*—This paper studies group elevator scheduling with destination entry and future traffic information for normal operations, as well as for coordinated emergency evacuation. By exploiting the separable problem structure, a two-level formulation is established capable of modeling advance information. An approach is then developed by incorporating several innovative ideas into a decomposition and coordination framework, aiming to achieve near-optimal performance. The approach has also been extended for cases with little or no future traffic information and coordinated emergency evacuation. Numerical testing results are encouraging and further improvement is needed to reduce CPU time for online implementation.**

***Index Terms*—Coordinated emergency evacuation, destination entry, dynamic programming, group elevator scheduling, Lagrangian relaxation, optimization.**

Manuscript received September 1, 2006. This paper was recommended for publication by Associate Editor Y. Narahari and Editor N. Viswanadham upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant DMI-0423607 and in part by a Contract from the United Technologies Research Center.

P. B. Luh and B. Xiong are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-2157 USA (e-mail: Peter.Luh@uconn.edu; bo.xiong@uconn.edu).

S.-C. Chang is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan (e-mail: scchang@cc.ee.ntu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2007.895217

I. Introduction

SCHEDULING of a group of elevators in a building has long been recognized as an important issue to improve transportation efficiency, since elevator service ranks second after heating, ventilation and air conditioning (HVAC) as the main complaints of building tenants [29]. Anything to enhance elevator performance, e.g., shorter wait times during rush hours, will improve passenger satisfaction. The problem, however, is difficult because of complicated elevator dynamics, various traffic patterns with uncertain arrivals and destinations, and the combinatorial nature of discrete optimization.

To overcome the difficulties caused by traffic uncertainties, one important trend is to explore advance information. For example, most of conventional elevators have simple up and down buttons for "hall calls," and destinations are not known until passengers placed "car calls" from inside an elevator. Newer systems have keypads to enter passenger destination floors, and destinations are known in advance [18], [12]. Beyond such keypads, future passenger arrival and destination information might be collected from entrance security systems or radio frequency identification (RFID). An RFID tag on a person may allow the system to sense when the person is approaching an elevator and which floor is likely to be the destination, e.g., the home floor or the cafeteria floor (Mitsubishi's smart RFID-enabled elevators [27]. Sensor networks might also be able to collect future passenger arrival information [7]. Clearly, advance information brings new opportunities for group elevator scheduling. How to effectively utilize such information, however, remains an open and challenging issue.

In addition to good performance during normal operations, group elevator scheduling has a new significance on speedy egress driven by homeland security concerns. In high-rise buildings, stairs alone are inefficient for emergency evacuation because they become congested, people slow down during the long distance from top floors to the ground, and the elderly and disabled might not be able to use stairs at all [14]. The potential of using "safe elevators" for evacuation has been demonstrated for certain situations such as the detection of chemical or biological agents, or fires in one wing of a building [19]. Coordinated emergency evacuation is a key egress method where occupants at each floor are assumed to evacuate in a coordinated and orderly manner [13].

This paper presents the optimized scheduling of a group of elevators with destination entry and future traffic information for normal operations and for coordinated emergency evacuation. Relevant results in the literature are reviewed in Section II. The problem formulation is presented in Section III. To model destination entry, the destination floor for each passenger is specified together with his/her arrival time and arrival floor. To model future traffic information, a look-ahead time window is established, where future information within the window is assumed available, and ignored, otherwise. Using a rolling horizon scheme, the time window then moves forward to construct snapshot problems periodically or as needed. For each snapshot, it is noted that individual elevators are coupled only through serving a common pool of passengers. To capture this "separable" structure, a two-level formulation is established. The high level is for passenger-to-elevator assignment, and the low level is for single elevator dispatching, with elevator capacity constraints and elevator dynamics embedded within individual elevators. Such a formulation is generic and applicable to different traffic patterns, including up-peak (all the passengers arrive at the lobby), down-peak (all the passengers go to the lobby), and interfloor (passengers have multiple arrival floors and destination floors).

A snapshot problem has been shown to be NP-hard [23]. To overcome the complexity, a decomposition and coordination approach is developed by exploiting problem separability based on Lagrangian relaxation. Individual elevator subproblems are first obtained by relaxing the coupling passenger-to-elevator assignment constraints. Since a subproblem is still NP-hard [26], our goal is not to optimally solve all or any subproblems. Rather, from the surrogate optimization concept [33], a subproblem solution "better than" the previous one is "good enough" to set the multiplier updating direction. Subproblems are, thus, individually solved by using ordinal optimization-based local search, where nodes of the search tree are first evaluated and ranked by using the "three-passage heuristics," and top ranked nodes are then selectively optimized by using dynamic programming. Individual elevators are then coordinated through the iterative updating of multipliers for near-optimal solutions. The method is presented in Section IV. This method, although targeted for newer elevators with destination entry systems, could be extended for traditional elevators without destination entry.

In Section V, statistical information is gathered online to improve performance for up-peak traffic with little or no future information. A parking strategy is also developed for low intensity traffic with little or no future information. Both are seamlessly integrated into our method. In Section VI, our method is extended for a simplified model of coordinated emergency evacuation.

Extensive numerical testing has been performed. The results presented in Section VII demonstrate near-optimal solution quality and computational efficiency. Future traffic information is shown to be valuable to improve performance by comparing performance obtained with different window lengths. The potential of using elevators for coordinated emergency evacuation has also been demonstrated.

II. Literature Review

Two classes of methods have been developed for group elevator scheduling without destination entry information: heuristics and optimization methods. Many heuristic rules have been presented in the literature, such as the highest unanswered floor first, the longest waiting passenger first, equal load among elevators, static zoning, and collective control (see a good survey in [30]). For collective control, for example, an elevator sequentially serves the collective hall calls in its moving direction before reversing, and stops at the nearest call first without coordinating with other elevators. This rule is easy to implement. However, one drawback is bunching, i.e., several elevators move close to each other in response to a hall call when only one elevator is actually needed. Statistical analysis shows that good steady-state performance can be achieved for up-peak traffic by releasing elevators at the lobby at a proper time interval [3]. In general, heuristic rules are computationally efficient, but cannot consistently yield good performance for various traffic patterns.

Among optimization methods, a steady-state policy based on queueing theory was developed to minimize the average wait time for up-peak without destination entry information, where the decision variable was the interval between successive elevator releases at the lobby [24], [25]. Passenger arrivals were assumed to follow a Poisson process. To make the analysis tractable, time delays due to elevator stops, passenger loading and unloading, and door open and close were aggregated into elevator round-trip times (RTTs), which were assumed to be independent and exponentially distributed. This up-peak dispatching policy will be compared with our method in Section VII. For up-peak and down-peak, a method based on Lagrangian relaxation was developed to minimize the RTT of elevators, but not for general passenger-based performance [6]. To obtain optimal policies for all traffic patterns, a method based on multi-agent reinforcement learning was developed in [9]. Each agent is responsible for controlling one elevator and is trained by using neural networks. This method requires 60 000 offline hours of simulated elevator operation to converge for one specific down-peak scenario. Such a training effort is clearly a major drawback. A method based on genetic algorithm was presented in [8]. It represents possible solutions by chromosomes, and a fitness function is used to evaluate the performance of solutions in a population. Because of the large search space, this method may not be computationally efficient.

The inherent computational complexity and real-time requirements led to the development of incremental optimization methods, i.e., assigning a passenger to an elevator upon each hall call. For example, the new passenger is assigned to an elevator to minimize the delays of passengers already inside or assigned to that elevator in [22]. Incremental methods are generally not that good since they consider the assignment of one passenger at a time and are not aimed to optimize the performance over a set of passengers. To improve performance for cases with low passenger arrivals, the method has been enhanced by adding the parking feature, i.e., elevators are parked at floors where they are likely to be needed [5].

Incremental optimization methods have also developed for group elevator scheduling with destination entry information [12]. For example, the "estimated time to destination" (ETD) method assigns a new passenger to an elevator to minimize his/her estimated time to destination plus the delays of passengers already inside or assigned to that elevator [28]. No optimization method beyond incremental optimization has been found for elevators with destination entry. To investigate the performance limit, a branch-and-bound-based offline method in [17]. To the best of our knowledge, Inamoto is the first one that exploited the separability of problem structure. Based on our

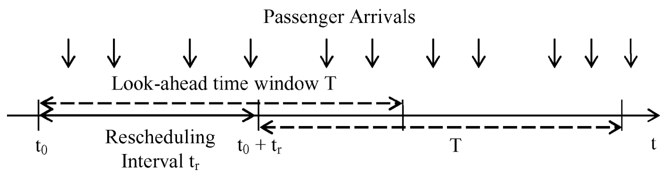

Fig. 1. Rolling horizon scheme with a look-ahead window T.

earlier version of this paper [32], a Lagrangian relaxation-based offline method were developed to obtain a performance limit [31].

Driven by homeland security concerns for speedy egress, the potential of using "safe elevators" for evacuation has been studied and new evacuation methods have been suggested [20], [19]. In [20], for example, elevators are dispatched between the lobby and certain designated "rescue floors." No optimization papers have been found for elevator operations in emergencies. In the following, our optimization formulation for the normal mode is presented first.

III. Problem Formulation for Normal Mode

Consider a building with F floors and a group of N_e elevators with a destination entry system. To model future traffic information, a look-ahead time window parameterized by its length T is used, where future information within the window is assumed available, and ignored otherwise. Cases with different levels of future information can thus be modeled by appropriately adjusting the window length, including zero (i.e., no future information). Using a rolling horizon scheme, the time window moves forward to construct snapshot problems periodically or as needed,[1] as shown in Fig. 1.

At the current rescheduling point t_0, passengers to be considered for the snapshot problem include the ones who are already inside elevators, the ones who are waiting, the ones who will arrive between t_0 and $t_0 + T$. Mathematically, they are grouped into two sets: S_p (N_p passengers who have been *picked up* but not yet delivered to their destination floors) and S_n (N_n passengers who have *not* yet been picked up). Together there are N passengers ($N = N_p + N_n$) under consideration, and the traffic information for passenger i $(0 < i \le N)$ is specified by the arrival time t_i^a, the arrival floor f_i^a, and the destination floor f_i^d for all i. Such information is available for the snapshot problem at t_0 under the previous assumption of perfect information within the time window.

It is noted that the elevators are independent of each other except that they have to serve a common pool of passengers. A two-level formulation is thus established. The high level is for passenger-to-elevator assignment, and the low level is for single elevator dispatching, with elevator capacity constraints and elevator dynamics embedded within individual elevators. This two-level concept follows that of Inamoto *et al.*, [17], although the problems addressed and the approaches developed here are quite different from those of Inamoto *et al.*, [17].

In the following, the coupling assignment constraints are first established in Section III-A, individual elevator constraints in Section III-B, and the objective function and the snapshot problem are then presented in Section III-C.

[1]Note that this scheme can be specialized to the incremental scheduling where a snapshot problem is constructed once a new passenger arrived.

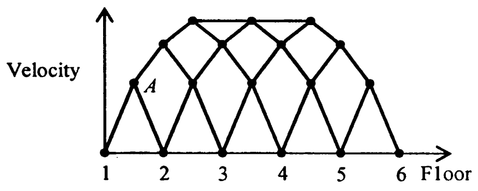

Fig. 2. A model of single elevator dynamics.

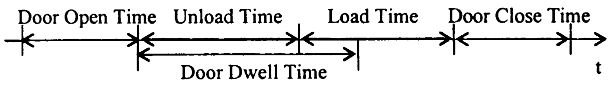

Fig. 3. Elevator parameters.

A. Passenger-to-Elevator Assignment Constraints

Assignment constraints state that each passenger must be assigned to one and only one elevator, i.e.,

$$\sum_{j=1}^{N_e} \delta_{ij} = 1, \quad \forall i \tag{1}$$

where δ_{ij} is a zero-one indicator variable equal to one if passenger i is assigned to elevator j and zero, otherwise. For a snapshot problem, passengers in S_p have already been picked up, and their $\{\delta_{ij}\}$ are fixed. Assignment decisions are thus restricted to passengers who have not yet been picked up, i.e., in S_n.

B. Individual Elevator Constraints

For individual elevators, the dynamics is based on Nikovski and Brand [22], as shown in Fig. 2, for an up-moving elevator. In the figure, the x axis is the floor index and the y axis is the elevator velocity. The "halfway point" A between floor 1 and floor 2 represents the location where the elevator should start decelerating if it is to stop at floor 2, and other halfway points are similarly defined. The trajectory for a down-moving elevator is similarly described. At a rescheduling point, passengers can be picked up by an elevator in its current "one-way trip" only if the elevator has not passed the corresponding halfway point.

Elevator travel times between any two floors are derived from Fig. 2. Other elevator parameters such as door open time, door close time, and load/unload time per passenger are also assumed deterministic and given; and their relationship is shown in Fig. 3. Door open time is the time from the start of door opening until fully opened, and door close time is the time from the start of door closing until fully closed. Load time per passenger and unload time per passenger represent the time for a passenger to get in or get out, respectively. In addition to the above, there is a "door dwell time," representing the minimum amount of time that the door should stay open after fully opened. If there is no passenger unloading or loading after the door dwell time expires, then the door starts to close without additional delay. Based on the above elevator dynamics and parameters, a simulator was developed. For a given dispatching strategy and a given set of passenger assignments, the simulator generates an elevator trajectory while taking into account elevator capacity at passenger loading times.

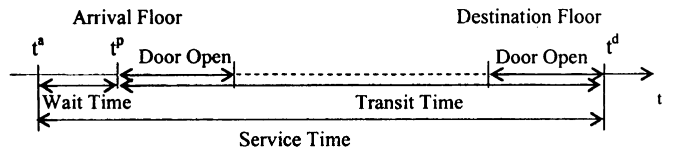

Fig. 4. Definitions of time metrics.

For individual elevators, capacity limits are formulated as linear inequality constraints

$$\sum_{i=1}^{N} \zeta_{ijt} \le C_j, \qquad \forall j, t \tag{2}$$

where ζ_{ijt} is a zero-one indicator variable equal to one if passenger i is in elevator j at time t and zero, otherwise, and C_j is the capacity of elevator j in terms of the number of passengers. Let the pickup time of passenger i by elevator j be denoted as t_i^p and the departure time be denoted as t_i^d, then $\zeta_{ijt} = 1 \; iff \; t_i^p \le t < t_i^d$. Pickup times $\{t_i^p\}$ and departure times $\{t_i^d\}$ can in theory be represented as a function of the dispatching strategy φ_j for elevator j and relevant information of passengers assigned to the elevator

$$\{t_i^p, t_i^d\} = \Gamma\left(\varphi_j, \{t_{i'}^a, f_{i'}^a, f_{i'}^d, \forall i' \in S_j\}\right), \\ \text{where } S_j \equiv \{i' | \delta_{i'j} = 1\} \text{ and } i \in S_j. \tag{3}$$

In view that the number of variables $\{\zeta_{ijt}\}$ is infinite for a continuous-time formulation and the mapping Γ could be too complicated to describe, constraints (2) and (3) are not explicitly represented. Rather, the simulator is used to simulate an elevator's dynamics (3), and to enforce capacity constraints (2). In this sense, constraints (2) and (3) are implicitly embedded within the simulator.

C. The Objective Function and the Snapshot Problem

For passenger i, the wait time is defined as $T_i^W \equiv t_i^p - t_i^a$, the transit time as $T_i^T \equiv t_i^d - t_i^p$, and the service time as $T_i^S \equiv T_i^W + T_i^T$. These definitions are depicted in Fig. 4.

For passengers, long wait times and many stops during a ride are not good. To improve passenger satisfaction, the objective function is a weighted sum of wait times and transit times of all passengers, i.e.,

$$J \equiv \sum_{i=1}^{N} T_i \quad \text{with } T_i = \alpha T_i^W + \beta T_i^T \tag{4}$$

where α and β are weights specified by building designers or management. Note that if $\alpha = \beta = 1$, then $T_i = T_i^S$. For each snapshot, the optimization problem is to minimize the objective function (4) subject to (1)–(3). The decisions include $\{\delta_{ij}\}$ for all $i \in S_n$ and the dispatching strategy $\{\varphi_j\}$ for all j.

As presented earlier, constraints (1) that couple individual elevators are linear inequality constraints, constraints (2) and (3) are embedded within individual elevators, and the objective function (4) is additive in terms of passengers but not elevators. To facilitate the decomposition of the problem into individual

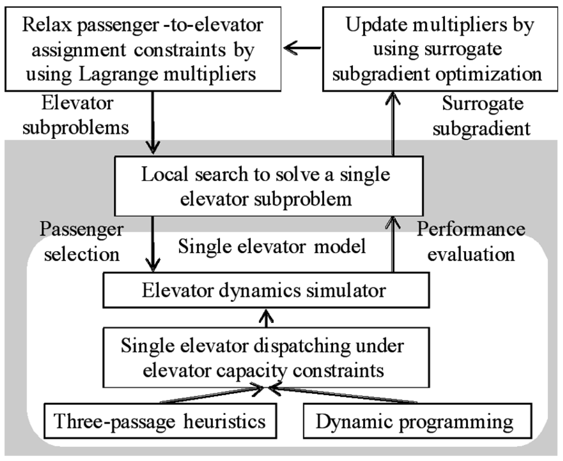
Relax passenger -to-elevator assignment constraints by using Lagrange multipliers
Update multipliers by using surrogate subgradient optimization
Elevator subproblems
Surrogate subgradient
Local search to solve a single elevator subproblem
Passenger selection
Single elevator model
Performance evaluation
Elevator dynamics simulator
Single elevator dispatching under elevator capacity constraints
Three-passage heuristics
Dynamic programming

Fig. 5. The decomposition and coordination approach.

elevator subproblems, (4) is transformed into an additive form in terms of elevators by using (1)

$$J = \sum_{i=1}^{N} (T_i \sum_{j=1}^{N_e} \delta_{ij}) = \sum_{j=1}^{N_e} \sum_{i=1}^{N} (\delta_{ij} T_i). \tag{5}$$

With this transformation, a two-level separable formulation is thus established.

IV. Solution Methodology for Normal Mode

For the snapshot problem, a simplified version without detailed elevator dynamics has been shown to be NP-hard [23]. To overcome the complexity, the separable structure of our formulation is exploited. A decomposition and coordination approach is developed, and the problem is decomposed into individual elevator subproblems through the relaxation of assignment constraints in Section IV-A. In view that each subproblem is still NP-hard [26], our goal is not to optimally solve all or any subproblems. Rather, from the surrogate optimization concept [33], a subproblem solution "better than" the previous one is "good enough" to set the multiplier updating direction. Subproblems are thus individually solved by using the ordinal optimization-based local search in Section IV-B. For the local search, passenger selections are first quickly evaluated and ranked by using the three-passage heuristics, and top ranked selections are then selectively optimized by using dynamic programming. Individual elevators are coordinated through the iterative updating of multipliers for near-optimal solutions in Section IV-C. The schematic of our approach is shown in Fig. 5, and the detailed procedure is presented below.

A. Decomposition Into Individual Elevator Subproblems

Assignment constraints (1) that couple individual elevators are relaxed by using nonnegative Lagrange multipliers $\{\lambda_i\}$

$$\begin{aligned} L(\{\lambda_i\}, \{\delta_{ij}\}) &= \sum_{j=1}^{N_e} \sum_{i=1}^{N} (\delta_{ij} T_i) + \sum_{i=1}^{N} \lambda_i \left(1 - \sum_{j=1}^{N_e} \delta_{ij}\right) \\ &= \sum_{j=1}^{N_e} \sum_{i=1}^{N} (\delta_{ij} T_i - \lambda_i \delta_{ij}) + \sum_{i=1}^{N} \lambda_i. \end{aligned} \tag{6}$$

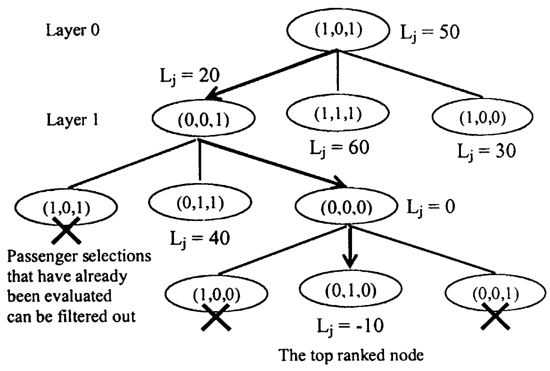

Fig. 6. Illustration of the local search.

By collecting all the terms related to elevator j from (6), the subproblem j is obtained as

$$\min_{\{\delta_{ij},\varphi_j,\forall i\in S_n\}} L_j, \quad \text{with } L_j \equiv \sum_{i=1}^{N}(\delta_{ij}T_i - \lambda_i\delta_{ij}) \qquad (7)$$

subject to (2) and (3). Subproblem (7) is to find the optimal passenger selections and the optimal dispatching of elevator j to serve the selected passengers for a given set of multipliers $\{\lambda_i\}$.

B. Solving Individual Elevator Subproblems by Ordinal Optimization-Based Local Search

Since each subproblem is still NP-hard [26], our goal is not to optimally solve all or any subproblems. Rather, from the surrogate optimization concept [33], a proper multiplier updating direction can be obtained if the surrogate optimization condition, the surrogate initialization condition (to be presented in Section IV-B1, and the stepsize condition (to be presented inSection IV-B3) are satisfied. The surrogate optimization condition states that a subproblem solution should be "better than" the previous one, i.e.,

$$L_j\left(\{\lambda_i^n\},\{\delta_{ij}^n\}\right) < L_j\left(\{\lambda_i^n\},\{\delta_{ij}^{n-1}\}\right) \qquad (8)$$

where $\{\lambda_i^n\}$ is the set of multipliers at the nth iteration. In the following, subproblems are individually solved by using ordinal optimization-based local search in Section IV-B1 to satisfy (8). For the local search, the three-passage heuristics in Section IV-B2 is used to quickly rank nodes in the search tree, and dynamic programming in Section IV-B3 is used to selectively optimize top ranked nodes.

1) Ordinal Optimization-Based Local Search: To find a set of passenger selections $\{\delta_{ij}\}$ that satisfies (8) under a given set of multipliers $\{\lambda_i\}$, a local tree search method is developed. In the search tree, each node is an I_n -tuple consisting of $\{\delta_{ij}|\forall i \in S_n\}$ indicating whether passenger i is selected for elevator j or not. For purpose of illustration, consider the subproblem for elevator j with three passengers, as shown in Fig. 6. The root node, i.e., a three-tuple (1, 0, 1), denotes that the first and the third passengers are selected but not the second one; and is the passenger selections obtained from the previous iteration. This root is "locally" expanded to generate three child nodes by varying one passenger selection at a time, as shown in layer 1 of the figure. For each node, the passenger selections $\{\delta_{ij}\}$ are fixed, while the dispatching strategy $\{\varphi_j\}$ is yet to be optimized

$$\min_{\{\varphi_j\}} L_j, \quad \text{with } L_j \equiv \sum_{i=1}^{N}(\delta_{ij}T_i - \lambda_i\delta_{ij}) \qquad (9)$$

subject to (2) and (3).

This optimization problem (9) is still NP hard [15], and has to be solved for each node so as to select a node that is better than the root node of the search tree. To overcome the costly computational requirements, the ordinal optimization concept that ranking is robust even with rough evaluations is utilized [16]. Our key idea is to use the three-passage heuristics (to be presented in Section IV-B2) to efficiently dispatch the elevator and evaluate L_j $(\{\lambda_i\},\{\delta_{ij}\})$ for each node. After the search tree is traversed and nodes are ranked, exact optimization by dynamic programming (to be presented in Section IV-B3) is then used to find a high ranking node that is better than the root node [(8) above]. The ordinal optimization-based local search is summarized as follows. $r = 0$.

Step 0. [Initialize the root node.] For the first iteration, all subproblems should be optimally solved and this is to satisfy the surrogate initialization condition (20), [33]. A quick way to satisfy this condition is to initialize λ_i^0 at 0 for all i, since in this case the optimal solutions for all subproblems are simply $\delta_{ij}^0 \equiv 0$. For all other iterations, set passenger selections to their latest values $\{\delta_{ij}^{n-1}\}$ obtained from the previous iteration. For both cases, set the layer of search $r = 0$.

Step 1. [Generate child nodes.] Locally perturb the selections by changing the selection of one passenger at a time over N passengers to generate a set of child nodes, and set $r = r + 1$.

Step 2. [Rank.] Use the same set of multipliers $\{\lambda_i^n\}$ for all nodes. Evaluate L_j $(\{\lambda_i^n\},\{\delta_{ij}^n\})$ for each of the child nodes by using the three-passage heuristics, and identify the node that leads to the smallest value of L_j. If this node has a smaller value than that of the best node obtained thus far, rank it higher than that node, store it in a queue, and then go to *Step 1.* Otherwise, go to *Step 3.*

Step 3. [Verify.] Verify if the highest ranked node obtained based on the rough evaluation of *Step 2* indeed represents a solution better than the root node by using dynamic programming for exact optimization of (9). If this node is indeed better than the original node, then it is accepted and the search stops. Otherwise, the second highest ranked node is taken from the queue and then evaluated by using dynamic programming (DP), etc. If no better selections are found, the original selections (the root node) are maintained and the next subproblem is solved.

2) Three-Passage Heuristics: The three-passage heuristics [11] is used to quickly rank nodes in the search tree, as briefly summarized below. For an elevator moving up, the passengers are served in three "passages." The first passage is from the elevator's current position to the first "reversal floor" (the highest floor with loading or unloading if the elevator moves up, and the lowest floor with loading or unloading otherwise) to serve up-traffic on the first-come-first-serve basis subject to elevator capacity constraints. For this passage, passengers can be picked up only if the elevator has not passed the corresponding halfway point. The second passage is from the first reversal floor to the second reversal floor to serve down-traffic, also on the first-

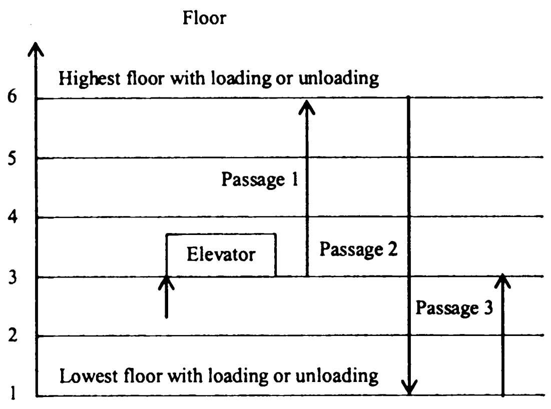
Floor
Highest floor with loading or unloading
Passage 1
Elevator
Passage 2
Passage 3
Lowest floor with loading or unloading
1
2
3
4
5
6

Fig. 7. Illustration of the three-passage heuristics.

come-first-serve basis. The third passage is then from the second reversal floor to serve up-traffic, and the process repeats. An example is shown in Fig. 7, where the elevator moves up from floor 3. The first passage is from floor 3 to floor 6, the second passage is from floor 6 to floor 1, and the third passage is from floor 1 to floor 3. Using this heuristics as the dispatching strategy φ_j, our simulator quickly generates the trajectory for the elevator to obtain pickup times, departure times, and service times for passengers under consideration ($\{i|\delta_{ij} = 1\}$). The value L_j ($\{\lambda_i\}, \{\delta_{ij}\}$) is then evaluated to be used in the search tree.

3) Dynamic Programming (DP) : To verify if a top ranked node is better than the root node, DP is used to optimize the dispatching strategy ϕ_j for elevator j through a novel definition of DP stages, states, decisions, and costs. It is noted that a one-way trip consists of multiple stop floors (i.e., floors to be served by the elevator), including a start floor and a reversal floor. Our key idea is that for the one-way trip, if the set of stop floors is fixed, then the passengers to be delivered on this trip are limited to those traveling between the stop floors. With this, a stage is defined as a one-way trip, as opposed to a discrete time interval or a stop. The mathematical definitions of individual DP components are presented below.

DP Stages: A stage is defined to be a one-way trip of the elevator without changing its direction. For stage $k (k > 0)$, let time τ_k be the start time when the door is fully opened at the start floor; and let time τ_{k+1} be the end time when the door is fully opened at the reversal floor. The end time of stage k is, thus, the start time of stage $k+1$, and the current rescheduling time is the start time of stage 0. Note that the number of DP stages is not a constant, but depends on the selection of passengers of a node on which DP is applied.

DP States: For elevator j, let S_k denote the set of selected passengers that have not been delivered at τ_k. The minimum information required to describe the system includes the start time τ_k, the elevator position f_{jk} and direction d_{jk} at τ_k, and traffic information (arrival times, arrival floors, and destination floors) for passengers in S_k. The DP state x_k is thus represented by

$$x_k = \left(t_k, f_{jk}, d_{jk}, \{t_i^a, f_i^a, f_i^d | \forall i \in S_k\}\right). \quad (10)$$

DP Decisions: Let F_k denote the set of stop floors of the one-way trip in stage k. In view of Fig. 2, a floor is not considered at the rescheduling point if the elevator has passed the

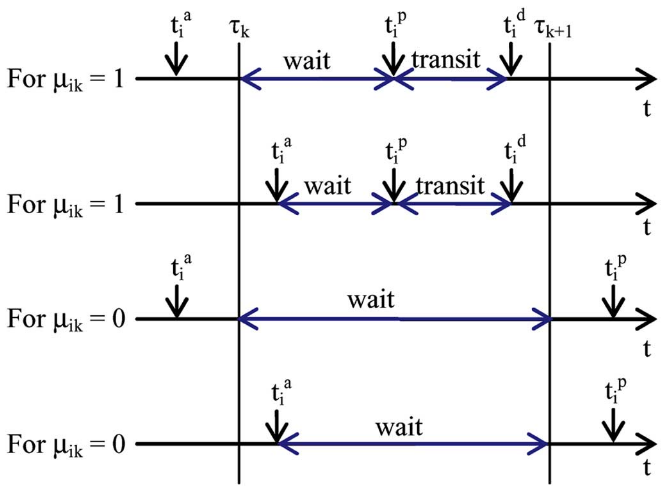

Fig. 8. Stagewise costs in stage k for different cases.

corresponding halfway point. Once the set F_k is determined, passengers to be delivered in the one-way trip are then limited to those traveling between the stop floors. The passengers are served according to the stop sequence subject to elevator capacity constraints. Without loss of generality, those with identical arrival and destination floors are served under the first-come-first-serve assumption. The decision u_k includes which floors to stop and which passengers to serve in stage k. Mathematically, $u_k = \{F_k, \mu_{ik} | \forall i \in S_k, f_i^a \in F_k, f_i^d \in F_k\}$, where μ_{ik} is a zero-one decision variable equal to one if passenger i is to be delivered to the destination floor in stage k and zero, otherwise.

System Dynamics: Given x_k and u_k, the elevator trajectory in stage k is generated by using the simulator. As a result, the pickup times $\{t_i^p\}$ and the departure times $\{t_i^d\}$ of passengers delivered in stage k, the start time τ_{k+1} of stage $k+1$, and the elevator position and direction at τ_{k+1} are obtained. With this, the DP state x_{k+1} evolving from x_k is determined.

DP Stagewise Costs: For DP, the cost function must be represented in a stagewise additive form. The total wait time for a passenger is the sum of the wait times that the passenger spends waiting in individual stages. The total wait time $t_i^p - t_i^a$ is thus stagewise additive, as depicted in Fig. 8.

Mathematically, if the passenger is served in stage $k(\mu_{ik} = 1)$, then the wait time is $t_i^p - \max(\tau_k, t_i^a)$, as illustrated by the first two cases in Fig. 8; otherwise, it is $\tau_{k+1} - \max(\tau_k, t_i^a)$, as illustrated by the last two cases. It can also be seen from the figure that the transit time is within a single stage, under the assumption that each passenger is served in a one-way trip. If the passenger is served in stage k, then the transit time is $t_i^d - t_i^p$; otherwise, it is 0. The objective function (4), i.e., the weighted sum of wait and transit times, can thus be represented in terms of the following stage-wise cost:

$$g_k(x_k, u_k) = \sum_{i \in S_k, \mu_{ik}=1} \left\lfloor \alpha\left(t_i^p - \max(\tau_k, t_i^a)\right) + \beta\left(t_i^d - t_i^p\right)\right\rfloor + \sum_{i \in S_k, \mu_{ik}=0} \alpha\left(\tau_{k+1} - \max(\tau_k, t_i^a)\right). \quad (11)$$

As presented above, for each state x_k in stage k, state transitions can be obtained by considering all possible decisions

$\{u_k\}$. Associated with x_k and u_k, the elevator trajectory in stage k is generated by using the simulator and the cost in stage k can then be computed according to (11). For a given initial state, multiple final states exist. Therefore, forward DP instead of backward DP is applied to obtain an optimal trajectory. The process is standard, and can be found, for example, in [4, p. 490].

C. Update Multipliers Using Surrogate Subgradient

As presented earlier, the subproblems are solved one at a time by using local search such that $\{\delta_{ij}\}^n$ satisfies the surrogate optimization condition (8). The surrogate dual is obtained from (6)

$$\tilde{L}^n \equiv \tilde{L}\left(\{\lambda_i^n\},\{\delta_{ij}^n\}\right) = \sum_{j=1}^{N_e}\sum_{i=1}^{N}\left(\delta_{ij}^n T_i^n - \lambda_i^n \delta_{ij}^n\right) + \sum_{i=1}^{N}\left(\lambda_i^n\right). \tag{12}$$

The Lagrangian multipliers $\{\lambda_i\}^n$ are updated by using the surrogate subgradient method

$$\lambda_i^{n+1} = \lambda_i^n + s^n \tilde{g}_i^n, \tag{13}$$

where component i of the surrogate subgradient $\tilde{g}^n$ is

$$\tilde{g}_i^n = 1 - \sum_{j=1}^{N_e} \delta_{ij}^n \tag{14}$$

in [33, eq. (26)]. The stepsize s^n in (13) is required to satisfy the following surrogate stepsize condition:

$$0 < s^n < (L^* - \tilde{L}^n) \Big/ \sum_{i=1}^{N} (\tilde{g}_i^n)^2 \tag{15}$$

in [33, eq. (27)], where L^* is the optimal dual cost. Since L^* is unknown in general, it needs to be estimated online to calculate s^n. To obtain such an estimate, a feasible primal cost is evaluated to serve as an upper bound on L^* as presented below.

To obtain a feasible primal cost, a feasible solution is needed. The solutions to subproblems, when put together, are generally infeasible, i.e., the relaxed assignment constraints are violated, e.g., a passenger is assigned to no elevator or to two elevators. Based on an infeasible solution, a heuristics has been developed to construct a feasible solution. Specifically, any passenger who was not assigned to an elevator is randomly assigned to an elevator; any passenger who was assigned to an elevator is assigned to the same elevator; and any passenger who was assigned to two or more elevators is randomly assigned to one of those elevators. To reduce computational requirements, a feasible solution is constructed every few iterations as opposed to every iteration to evaluate the corresponding feasible cost. Let P^n be the minimal feasible cost obtained thus far. Based on P^n and the surrogate dual cost $\tilde{L}^n$, the optimal dual cost is then estimated according to

$$\hat{L}^* = (P^n + \tilde{L}^n)/2. \tag{16}$$

With (15), the stepsize is then set as

$$s^n = \rho(\hat{L}^* - \tilde{L}^n) \Bigg/ \sum_{i=1}^{N} (\tilde{g}_i^n)^2 \tag{17}$$

where $0 < \rho < 1$. The convergence of our method is guaranteed assuming that L^* in (15) is known. In view that L^* is generally unknown, the parameter ρ is usually kept small especially for large n to avoid, to some extent, possible violation of (15). Numerical testing in Section VII will demonstrate that this simple stepsizing heuristics produces good results.

The algorithm terminates when one of the following two stopping criteria is met: $(P^n - \tilde{L}^n)/\tilde{L}^n$ is less than a small positive number ε, or the maximum number of iterations has been reached. To quantify the solution quality for a snapshot problem, a dual cost (a lower bound to the optimal cost) can be obtained offline by optimally solving all the subproblems using a global tree search with DP applied to each node. The corresponding duality gap is then calculated as the relative difference between a feasible cost and the dual cost.

In addition to performance of snapshot problems, performance of the overall problem, which is solved as a sequence of snapshot problems, is of interest. The feasible cost for the overall problem can be evaluated based on realized pickup times and departure times of all passengers obtained across multiple snapshot problems. A lower bound to the overall performance can be obtained by solving the entire problem in one snapshot, where the window length equals the entire time horizon. Then, the corresponding dual cost serves as a lower bound.

As mentioned earlier, our method is developed for newer elevators with destination entry systems. Nevertheless, it can be extended to traditional elevators without destination entry. The idea is to estimate origin–destination pairs of passengers based on historical traffic data by exploiting the Kalman filter model of Ashok and Ben-Akiva [2]. With the estimated origin–destination pairs, snapshot problems for individual hall calls can then be formed and solved by using our method. The details are beyond the scope of this paper and are thus omitted here.

V. Cases With Little or No Future Traffic Information

For cases with little or no future traffic information as modeled by having small or zero time windows (still with destination entry), the optimization of the above snapshot problems is "myopic," and the overall performance may not be good. For example, suppose that there are four elevators available at the lobby and four passengers with different destination floors arrived at the lobby at about the same time in up-peak traffic. The "best" decision for this snapshot problem, e.g., to minimize the total service time, would be to dispatch one elevator for each passenger. This, however, would result in "bunching" of elevators, i.e., elevators moving close to each other. Passengers who arrive a little bit later than the previous four passengers then have to wait till one of the elevators returns to the lobby, resulting in a high overall cost. Bunching is less an issue for cases with sufficient future information since in this case optimized snapshot decisions are not myopic.

Another concern is related to low intensity traffic with little or no future information. In such a case, elevators are usually "idle," i.e., without passenger assignments. It has been shown that performance can be improved by "parking" idle elevators at floors where elevators are likely to be needed [5]. In the following, our method developed in Section IV will be extended to address these two concerns with little or no future traffic information.

A. *Extension for Up-Peak With Little or No Future Information*

To overcome the myopic difficulty of snapshot solutions with little or no future traffic information, statistical traffic information beyond what is available within the limited time window needs to be explored. One straightforward way is to form a stochastic optimization problem for each node in the search tree of Fig. 6 by considering random future passenger arrivals. This problem, however, is difficult to solve by using stochastic DP because of inherent complexity and the existence of multiple final states. As an alternative, our formulation and method are slightly modified based on the insight obtained from Barney's model for up-peak.

Barney's model for up-peak assumes that passengers arrive at the lobby at a constant rate with destination floors uniformly distributed [3, p. 151]. Statistical analysis shows that good steady-state performance can be achieved for such up-peak traffic by releasing elevators from the lobby at an equal time interval (the elevator "interdeparture time"), assuming that the capacity of an elevator is sufficient to accommodate new arrivals within the interdeparture time.

Based on the above, the method of Section IV is strengthened for up-peak by adopting the interdeparture time concept. The resulting method is to add elevator "hold" and "release" conditions to the formulation of Section III to space elevator departures from the lobby for up-peak. Specifically, for an even flow of passengers, elevators are "held" at the lobby for at least τ, i.e.,

$$t^m + \tau \le t^{m+1} \tag{18}$$

where t^m and t^{m+1} are successive departure times of elevators from the lobby, and τ is the interdeparture time. To derive τ, first the average RTT of an elevator is obtained from probabilistic analysis as

$$\mathrm{RTT} = 2\left(F - \sum_{f=1}^{F-1}\left(\frac{f}{\mathrm{F}}\right)^{P}\right)t_f + \left(F + 1 - F\left(\frac{F-1}{F}\right)^{P}\right)t_s + Pt_p \tag{19}$$

in [3, eq. (4.11)]. In the above, P is the number of passenger arrivals within the interdeparture time, t_f is the travel time between two adjacent floors, t_s is the time delay due to each stop, and t_p is the time delay due to loading and unloading of each passenger. Then, τ equals RTT divided by the number of elevators:

$$\tau = \mathrm{RTT}/N_e \tag{20}$$

in [3, eq. (4.7)]. For a given arrival rate, RTT in (19) is a function of P which depends on τ, while τ is a function of RTT based on (20). In view of this nested relationship, an iterative procedure has been developed to calculate RTT and τ for a given arrival rate [21]. Note that the "threshold rate" at which the number of new arrivals within the interdeparture time τ equals the elevator capacity C can be obtained as C/τ. Condition (18) is active for up-peak traffic if the window length T is less than τ, representing cases with little or no future traffic information.

For up-peak with a varying arrival rate, the interdeparture time τ needs to be calculated online. As mentioned above, the interdeparture time can be obtained for a given arrival rate. For our problem, the arrival rate is calculated as the number of passengers arrived during the past few (e.g., five) minutes plus the number of arrivals in the time window, then divided by the length of the interval considered [21].

To cover burst arrivals, elevators are "released" from the lobby when a certain percentage of elevator capacity is filled, i.e.,

$$\sum_{t^m \le t_i^p < t^{m+1}} \delta_{ij} > \zeta C_j \tag{21}$$

where ζ is a user specified percentage of elevator capacity.

To solve the problem with "hold" and "release" conditions, the decomposition and coordination approach of Section IV is used. At a rescheduling point, the sequence of elevators returning to the lobby is known and subproblems are solved in this sequence. For each elevator subproblem, conditions (18) and (21) are enforced for the elevator's first departure from the lobby. Subsequent departures will be considered in future snapshot problems.

For up-peak traffic, an elevator has no opportunity to serve other passengers once left the lobby until it returns to the lobby. As a result, elevator capacities are not well utilized and bunching is a serious issue when the window length T is small. For down-peak traffic, passengers arrive at multiple floors, and an elevator has opportunities to serve other passengers on its way down. Therefore, bunching is less an issue for down-peak traffic, and the same is true for interfloor traffic. Consequently, "hold" and "release" conditions are used only for up-peak with little or no future information.

B. *Parking Strategy for Low Intensity Traffic*

To develop a parking strategy for low intensity traffic, our idea is to divide the building into a number of nonoverlapping "zones," each consisting of a set of contiguous floors. Probabilities that the next passenger would arrive at individual zones are estimated, and idle elevators are parked at zones where they are likely to be needed. To avoid excessive move of elevators, floors in the same zone are not differentiated.

Specifically, suppose that an elevator became idle, making the number of idle elevators $N_e', 1 \le N_e' \le N_e$. At the current rescheduling point, the probability P^f of having new passenger arrivals at floor f before the next rescheduling point is estimated by assuming that the floors have independent Poisson arrivals. This is a reasonable assumption and has been validated in [1]. As a result, $P_f = 1 - e^{-\nu_f t_r}$, where ν_f is the arrival rate of floor f at the time of consideration, and t_r is the rescheduling interval. The probability of having new passenger arrivals at zone z before the next rescheduling point is $P_z = \sum_{f \in z} P^f$. The number of desired elevators parked at zone z is then calculated as $\lfloor N_e' \times P_z \rfloor$ (a truncated integer). By comparing the number of desired elevators in each zone with the number of elevators already parked there, the zones needing an idle elevator are identified. The newly idled elevator is then parked at one of these zones which is nearest to it. Our parking strategy covers up-peak, down-peak, and interfloor traffic, and is invoked when an elevator becomes idle. This strategy, in conjunction with the previous "hold" and "release" conditions for up-peak with little or no future information, is embedded within our method developed in Section IV to form a single "standard method."

VI. Scheduling for the Emergency Mode

As mentioned in Section I, fireproof elevators are potentially invaluable to support stairs for building egress in emergencies. In this section, a simplified model of coordinated emergency evacuation is developed, assuming that occupants comply with the instructions of evacuation coordinators (e.g., firefighters) to evacuate to the lobby in an orderly manner. The overall egress time, i.e., the time required to evacuate all the occupants, depends on the percentages of floor populations assigned to elevators versus stairs. To balance the traffic between elevators and stairs, a heuristic procedure has been developed in [21], where the percentage of floor populations assigned to elevators is a truncated linear function of the floor number. Within this context, the elevator egress time $T_e (= \max_i\{t_i^d\})$, i.e., the time required to evacuate occupants assigned to elevators, needs to be minimized. In the following, our focus is to solve this elevator evacuation problem.

For the elevator evacuation problem, occupants who are on the way to elevators are considered as future traffic, and their information can be collected through sensor networks [7]. To model such information, a look-ahead time window same as that for the normal mode is used. Suppose that the traffic information including arrival times and arrival floors is known within the time window and the destination floor is the lobby, then our elevator evacuation problem is to minimize T_e^2 (quadratic form is used for good algorithm convergence)

$$\min_{\{\delta_{ij},\varphi_j,\forall i\in S_n,\forall j\}} J, \quad \text{with } J \equiv T_e^2 \tag{22}$$

subject to passenger-to-elevator assignment constraints (1) and individual elevator constraints (2) and (3).

This problem is not separable since the objective function in (22) is not additive in terms of elevators as in (5). Therefore, the decomposition and coordination approach developed in Sections IV and V cannot be directly applied. Nevertheless, a separable formulation can be obtained by introducing a few extra variables. Specifically, let T_{ej} be the time required for elevator j to evacuate all the passengers assigned to it, i.e., $\max_i\{t_i^d|\delta_{ij}=1\}$. By requiring that T_{ej} be less than or equal to the egress time T_e for all j, N_e linear inequality "egress-time constraints" are formed for elevators

$$T_{ej} \le T_e, \quad \forall j. \tag{23}$$

With (23), the problem is converted to a separable form, and our decomposition and coordination approach can be applied. A Lagrangian function is first obtained by relaxing the assignment constraints (1) with nonnegative multipliers $\{\lambda_i\}$, and the new egress time constraints (23) with nonnegative multipliers $\{\mu_j\}$

$$\begin{aligned} L(\lambda,\delta) &= T_e^2 + \sum_{i=1}^{N}\lambda_i\left(1-\sum_{j=1}^{N_e}\delta_{ij}\right) + \sum_{j=1}^{N_e}\mu_j(T_{ej}-T_e) \\ &= \left(T_e^2 - \sum_{j=1}^{N_e}\mu_j T_e\right) \\ &\quad + \sum_{j=1}^{N_e}\left(\mu_j T_{ej} - \sum_{i=1}^{N}(\lambda_i\delta_{ij})\right) + \sum_{i=1}^{N}\lambda_i. \end{aligned} \tag{24}$$

Elevator subproblems are then constructed and solved, and a new "egress-time subproblem" for T_e is introduced, as presented below.

By collecting all the terms related to elevator j from (24), the subproblem j is obtained as

$$\min_{\{\delta_{ij},\varphi_j,\forall i\in S_n\}} L_j, \quad \text{with } L_j \equiv \mu_j T_{ej} - \sum_{i=1}^{N}(\lambda_i\delta_{ij}) \tag{25}$$

subject to (2) and (3). This subproblem is solved by using the ordinal optimization-based local search as presented in Section IV, where nodes of the search tree are first quickly evaluated and ranked by using the three-passage heuristics. The top ranked nodes are then exactly optimized by using DP, where T_{ej} is represented by the following stagewise cost:

$$g_k(x_k,u_k) = \tau_{k+1} - \tau_k. \tag{26}$$

The additional egress-time subproblem is obtained by collecting all the terms related to T_e from (24)

$$\min_{T_e} L_0, \quad \text{with } L_0 \equiv T_e^2 - \sum_{j=1}^{N_e}\mu_j T_e. \tag{27}$$

In view of its quadratic objection function with a nonpositive linear coefficient, this subproblem solution is $\sum_{j=1}^{N_e}((\mu_j)/2)$. The component of the surrogate subgradient used to update $\{\mu_j\}$ at the nth iteration is

$$\tilde{g}_j^n = T_{ej}^n - T_e^n. \tag{28}$$

The multipliers are then updated by following what was described in Section IV.

VII. Numerical Results and Insights

The above methods have been implemented in C++ and run on a P4 2 GHz Linux PC with 512 MB memory. Extensive numerical testing has been conducted. Five examples are presented below for a building with ten floors. Example 1 examines the performance of our standard method of Section V for small data sets in normal operations. It demonstrates near-optimal quality of solutions to snapshot problems and to the overall problem, and the utilization of destination entry information. Example 2 is based on large data sets with a constant arrival rate and demonstrates values of future traffic information, ordinal optimization, dynamic programming, batch optimization, statistical traffic information, and the parking strategy. Example 3 studies how performance changes over time for semi-realistic traffic with varying arrival rates. Example 4 compares results of our method with those reported in [24] to demonstrate the value of destination entry and future traffic information. Example 5 then examines coordinated emergency evacuation and demonstrates the potential of using elevators for building egress. For all the examples, elevators are uniformly distributed among floors at the beginning. Detailed data and results are available at http://www.engr.uconn.edu/msl/test_data/elevator.html.

Example 1: The first example considers normal operations with small data sets over a 120-s time horizon to demonstrate

TABLE I
ELEVATOR CONFIGURATIONS

Elevator Speed (m/s)	2.5	Elevator Acceleration (m/s^2)	0.7
Elevator Capacity (person)	16	Door Open Time (s)	1.5
Door Close Time (s)	2.5	Door Dwell Time (s)	3.0
Load Time per Passenger (s)	2.0	Unload Time per Passenger (s)	1.5

TABLE II
RESULTS FOR VARIOUS TRAFFIC PATTERNS

		Up-peak	Down-peak	Inter-floor
Avg. Service Time (s)	Mean	29.5	27.7	26.3
	STDEV	0.41	0.35	0.37
Avg. CPU Time per Snapshot (s)		2.3	2.8	2.1
Optimally Solve All Subproblems	Avg. Duality Gap (%)	4.8	3.6	4.2
	Avg. CPU Time (s)	50.1	43.3	48.6
Solve Problem in a Single Snapshot	Avg. Low bound (s)	27.2	26.1	24.4
	Avg. CPU Time (s)	842.6	751.9	810.3

near-optimal quality of solutions to snapshot problems and to the overall problem, and to demonstrate the utilization of destination entry information. There are two elevators whose parameters are taken from [10], as summarized in Table I. Three traffic patterns including up-peak, down-peak, and interfloor are examined, with a constant arrival rate of one passenger per ten seconds. For all the traffics, arrival times are independent and uniformly distributed over the entire horizon. For up-peak traffic, arrival floors are fixed at the first floor, i.e., the lobby; and to magnify the utilization of destination entry information, destination floors are independent and uniformly distributed between the fourth and the ninth floors for this example. For down-peak traffic, the destination floor is fixed at the lobby, and arrival floors are independent and uniformly distributed between the fourth and the ninth floors. For interfloor traffic, arrival floors are independent and uniformly distributed between the first and the sixth floors, and destination floors are independent and uniformly distributed between the fourth and the ninth floors. The objective function is the service time (i.e., $\alpha = \beta = 1$).

Snapshot problems are constructed by using a window length of 10 s with a rescheduling interval of 2.5 s, and then solved by using our standard method. Results from ten Monte Carlo simulation runs are summarized in Table II. For up-peak, the average service time is 29.5 s with a standard deviation of 0.41 s, and the average CPU time per snapshot is 2.3 s. To quantify the solution quality for snapshot problems, duality gaps are obtained offline by optimally solving all subproblems per Section IV-C. With an average CPU time of 50.1 s, the average duality gap is 4.8%, demonstrating that near-optimal solutions to snapshot problems are obtained. To quantify the solution quality for the overall problem, a lower bound is obtained offline by solving the entire problem in a single snapshot per Section IV-C. The average lower bound is 27.2 s obtained with an average CPU time of 842.6 s. This lower bound is close to the mean average service time of 29.5 s, demonstrating that near-optimal overall performance is achieved. For down-peak and interfloor, the above remarks for up-peak also hold.

To demonstrate the utilization of destination entry information, the scheduling details obtained from one of the Monte Carlo runs for up-peak are summarized in Table III. For each passenger, the arrival floor, the arrival time, the destination floor, the elevator serving the passenger, and the pickup time are shown. It can be seen that passengers 1, 3, and 4 going to the ninth floor are picked up by elevator 2 at 31.4 s; and passengers 2, 5, and 6 going to the fourth floor are picked up by elevator 1 at 49.5 s. This indicates that passengers with the same destination floor tend to be served by the same elevator, as opposed to being served according to their arrival times without considering their destinations.

TABLE III
RESULTS FOR A SAMPLE OF UP-PEAK TRAFFIC

Passenger ID	Arrival Floor	Arrival Time (s)	Destination Floor	Elevator ID	Pickup Time (s)
1	1	2.2	9	2	31.4
2	1	11.9	4	1	49.5
3	1	21.6	9	2	31.4
4	1	31.4	9	2	31.4
5	1	40.8	4	1	49.5
6	1	49.5	4	1	49.5
7	1	60.1	9	2	68.4
8	1	71.3	4	1	80.6
9	1	80.6	4	1	80.6
10	1	91.4	9	2	103.8
11	1	102.2	9	2	103.8
12	1	115.7	4	1	115.7

Example 2: This example considers normal operations with large data sets over a 1-h time horizon to demonstrate values of future traffic information, ordinal optimization, dynamic programming, batch optimization, statistical information, and the parking strategy. There are four elevators whose parameters are taken from Table I. Three traffic patterns including up-peak, down-peak, and interfloor are examined with a constant arrival rate of one passenger per three seconds. For all the traffics, arrival times are independent and uniformly distributed over the entire horizon. For up-peak traffic, destination floors are independent and uniformly distributed among all the floors except the lobby. For down-peak traffic, arrival floors are independent and uniformly distributed among all the floors except the lobby. For interfloor traffic, arrival floors are independent and uniformly distributed among all the floors, and for a given arrival floor, destination floors are independent and uniformly distributed among other floors. To examine the value of future traffic information, three window lengths of 0, 30, and 60 s are tested for each traffic pattern, with rescheduling intervals of 7.5, 7.5, and 15 s, respectively. The objective function is the service time.

Results obtained from ten Monte Carlo simulation runs are summarized in Table IV, where means and standard deviations of average service times and average CPU times per snapshot are shown. For up-peak, average service times for the three window lengths of 0, 30, and 60 s are, respectively, 47.3, 34.5, and 28.1 s, with standard deviations of 0.55, 0.45, and 0.54 s. This demonstrates that performance is improved with an increase in future traffic information, accompanied by a concomitant increase in the CPU time since more passengers are considered for a snapshot problem with a larger window. For down-peak and interfloor, the above remarks for up-peak also hold.

TABLE IV
RESULTS FOR VARIOUS TRAFFIC PATTERNS WITH DIFFERENT WINDOW LENGTHS

Unit (s)	Time Window	Rescheduling Interval	Avg. Service Time		Avg. CPU Time per Snapshot
			Mean	STDEV	
Up-Peak	0	7.5	47.3	0.55	5.7
	30	7.5	34.5	0.48	6.1
	60	15	28.1	0.54	9.0
Down-peak	0	7.5	38.9	0.58	6.5
	30	7.5	32.6	0.47	6.9
	60	15	27.1	0.52	9.1
Inter-Floor	0	7.5	34.3	0.59	6.6
	30	7.5	28.8	0.43	7.8
	60	15	26.2	0.51	9.6

TABLE V
RESULTS FROM DIFFERENT METHODS FOR SUBPROBLEM SOLVING

Unit (s)		Standard Method	Dynamic Programming Alone	Three-passage Heuristics Alone
Avg. Service Time	Mean	47.3	45.9	55.8
	STDEV	0.55	0.76	1.82
Avg. CPU Time per Snapshot		5.7	146.3	0.8

Furthermore, for each time window, interfloor has the smallest average service time among the three traffic patterns, followed by down-peak. This is reasonable since for interfloor traffic, an elevator can serve passengers on its way up and down, and for down-peak traffic on its way down, but for up-peak traffic only at the lobby.

To demonstrate the values of ordinal optimization, dynamic programming, batch optimization, statistical information, and the parking strategy, variations of our standard method are tested. The following results are obtained for up-peak traffic with a window length of zero and a rescheduling interval of 7.5 s unless stated otherwise, and are compared with those in Table IV as detailed below.

Ordinal Optimization and Dynamic Programming: Without ordinal optimization, subproblems are solved by using DP for each node in the search tree. In this case, the average service time is 45.9 s with a standard deviation of 0.76 s, as shown in Table V. Both metrics are close to those obtained by using our standard method, while the average CPU time of 146.3 s is significantly higher than 5.7 s obtained before. This demonstrates that ordinal optimization improves computational efficiency without a significant loss of solution quality. As another extreme, subproblems are solved by using the three-passage heuristics alone without dynamic programming in the local search. In this case, the average service time is 55.8 s with a standard deviation of 1.82 s. Both metrics are higher than those obtained by using our standard method, although the average CPU time of 0.8 s is less than 5.7 s obtained before. This indicates that our standard method selectively uses DP to achieve good performance without a significant requirement on CPU time.

Incremental Versus Batch Optimization: For our standard method, batch optimization is employed in the sense that passenger-to-elevator assignment decisions are optimized at

TABLE VI
RESULTS FROM BATCH OPTIMIZATION AND INCREMENTAL OPTIMIZATION

Optimization Schemes		Batch Optimization	Incremental
Avg. Service Time (s)	Mean	47.3	61.8
	STDEV	0.55	0.83
Avg. CPU Time per Snapshot (s)		5.7	2.2

TABLE VII
RESULTS FOR UP-PEAK WITH DIFFERENT WINDOW LENGTHS

Time Window (s)	Rescheduling Interval (s)	Avg. Service Time (s)		
		Standard Method	Not Apply	Always Apply
0	7.5	47.3	48.4	47.3
10	2.5	39.1	39.7	39.1
30	7.5	34.5	34.5	34.8
60	15	28.1	28.1	29.4

rescheduling points over a set of passengers. For incremental optimization, however, the assignment decision is optimized upon the arrival of each passenger assuming that no future traffic information is available. To compare these two optimization schemes, a window length of zero is used with a rescheduling interval of 7.5 s, and results are summarized in Table VI. With incremental optimization, the average service time is 61.8 s with a standard deviation of 0.83. Both metrics are higher than those obtained by using batch optimization, although the average CPU time of 2.2 s is less than 5.7 s obtained before. This indicates that incremental optimization cannot take the advantage of optimizing over a set of passengers.

Statistical Information: To demonstrate that statistical traffic information is valuable for up-peak with little or no future traffic information, two variations of our standard method are examined: "hold" and "release" conditions are not applied for any T, and two conditions are always applied. Four window lengths of 0, 10, 30, and 60 s are tested for each traffic pattern, with rescheduling intervals of 7.5, 2.5, 7.5, and 15 s, respectively. Average service times obtained by using our standard method and the two variations are summarized in Table VII. For cases with window lengths of 0 and 10 s, not applying the two conditions yields the worst performance among the three. For cases with window lengths of 30 and 60 s, however, always applying the conditions yields the worst performance among the three. This is consistent with our intuition that statistical traffic information is valuable for cases with little or no future information, but not for cases with sufficient future information.

Parking Strategy: To demonstrate the value of our parking strategy, low intensity interfloor traffic over a 1-h time horizon is examined, with a constant arrival rate of one passenger per 30 s. Arrival times, arrival floors, and destination floors are generated in the same way as those for interfloor traffic of Table IV. Results obtained with and without the parking strategy are summarized in Table VIII. With the parking strategy, the average service time of 18.4 s is significantly less than 25.7 s obtained without the strategy, although the average CPU time of 0.5 s is slightly higher than 0.4 s obtained without the strategy. This demonstrates that the strategy improves performance for low intensity traffic by parking elevators at floors where they are likely to be needed. The average standard deviation of 0.38 s is slightly

TABLE VIII
RESULTS FOR LOW INTENSITY INTERFLOOR TRAFFIC

Schemes		With Parking	Without Parking
Avg. Service Time (s)	Mean	18.4	25.7
	STDEV	0.38	0.26
Avg. CPU Time per Snapshot (s)		0.5	0.4

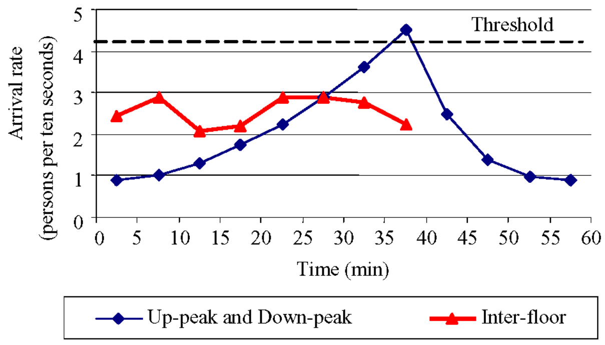

Fig. 9. Passenger arrival rates for each interval.

TABLE IX
RESULTS FOR VARIOUS TRAFFIC PATTERNS WITH VARYING ARRIVAL RATES

Unit (s)	Time Window	Rescheduling Interval	Avg. Service Time	Avg. Wait Time	Avg. Transit Time	CPU Time per Snapshot
Up-Peak	0	7.5	53.47	25.02	28.45	5.1
	30	7.5	44.73	16.82	27.91	8.5
	60	15	37.05	10.59	26.46	13.7
Down-Peak	0	7.5	49.87	26.35	23.52	5.0
	30	7.5	42.09	19.73	22.36	8.4
	60	15	35.02	12.87	22.15	13.6
Inter-Floor	0	7.5	46.37	23.11	23.26	5.1
	30	7.5	36.27	13.16	23.11	8.0
	60	15	33.55	11.52	22.03	12.7

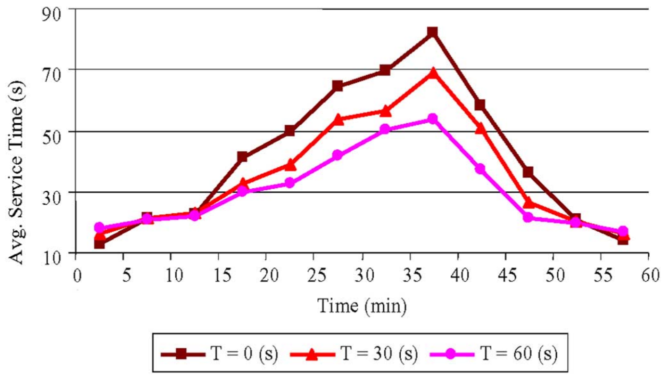

Fig. 10. Average service times for up-peak with different window lengths.

higher than 0.26 s obtained without the strategy. This might be caused by the hedging nature of the parking strategy.

Example 3: This example considers normal operations with semi-realistic traffic of varying arrival rates. There are four elevators whose parameters are taken from Table I. Three traffic data sets from [10] are examined, including up-peak, down-peak, and interfloor. To graphically represent time-varying arrival rates for each traffic pattern, the time horizon is divided into intervals of 5 min, and passenger arrival rates for each interval are plotted against time in Fig. 9. The "threshold rate" at which the number of new arrivals within its interdeparture time equals the elevator capacity can be obtained per Section V-A and is represented by the horizontal dotted line. For the up-peak traffic, arrival rates are relatively low before 20 min, high between 20 and 45 min, and then low after 45 min. For the interval between 35 and 40 min, the arrival rate is higher than the threshold rate. The down-peak traffic is generated by reversing the arrival and destination floors of the up-peak traffic. The interfloor traffic fluctuates within a relatively small range, as compared with the up-peak traffic. Detailed data for the three traffics are available at http://www.engr.uconn.edu/msl/test_data/elevator.html. For each traffic, three window lengths, 0, 30, and 60 s are tested, with rescheduling intervals of 7.5, 7.5, and 15 s, respectively. The objective function is the service time.

Average service times obtained by using our standard method are summarized in Table IX. For each traffic pattern, the average service time decreases as the window length increases. This demonstrates that future traffic information is valuable to improve performance, accompanied by a concomitant increase in the CPU time. To further analyze the results, average wait times and average transit times are also presented in Table IX. For up-peak, the average wait time decreases as the window length increases, but the average transit time stays almost the same. The reason for the latter might be that passengers with the same destination floor tend to be served by the same elevator, and the number of stops for each elevator on its way up stays almost the same (i.e., $2 \approx 9$ floors/4 elevators) regardless of the window length. For down-peak and interfloor, the above remarks for up-peak also hold.

For up-peak traffic, as mentioned earlier, the arrival rate for the interval between 35 and 40 min exceeds the threshold rate. To examine how performance changes over time under this traffic, average service times for each interval are plotted against time in Fig. 10. For each time window, the average service time increases from the beginning, reaches its maximum between 35 and 40 min, and then decreases. For an interval before 20 min and after 45 min during which the arrival rate is relatively low, the three time windows have similar average service times. For an interval between 20 and 45 min during which the arrival rate is relatively high, the 60-s window yields much better performance than the 30-s window, which in turn yields much better performance than the 0-s window. This indicates that future traffic information is valuable for heavy traffic.

For down-peak and interfloor, curves of average service times are omitted here since they are similar to those in Fig. 10. Furthermore, as can be seen from Table IX, for each time window, interfloor has the smallest average service time among the three traffics, followed by down-peak. This is consistent with the results of Table IV in Example 2.

Example 4: This example compares results obtained by using our method with those reported in Example B of [24] to demonstrate the value of destination entry and future traffic information in our dynamic optimization. For that paper, as reviewed in Section II, passenger arrivals were assumed to follow a Poisson

TABLE X
RESULTS FROM THE UP-PEAK DISPATCHING POLICY AND OUR METHOD

		Results Reported in Example B of Pepyne and Cassandras (1997)	Time window (s)	
			0	30
Avg. Wait Time (s)	Mean	23.6	21.5	18.4
	STDEV	/	0.38	0.31
Avg. CPU Time per Snapshot (s)		/	2.6	3.3

process. To make the analysis tractable, time delays due to elevator stops, passenger loading and unloading, and door open and close were aggregated into elevator round trip times, which were assumed to be independent and exponentially distributed. A steady-state policy was then developed based on queueing theory to minimize the average wait time for up-peak without destination entry information. For Example B, two elevators were considered, each with a capacity of ten passengers. The arrival rate was one passenger per 10 s and the average elevator RTT was 60 s. The resulting average wait time was 23.6 s, as shown in Table X.

For comparison purpose, the data used for our testing are set as follows. Elevator capacity is set to ten passengers. Elevator speed is set to 1.1 m/s and other elevator parameters of Table I are set to zero such that the average RTT obtained from (19) is 60 s. Passenger arrivals are then generated in the same way as that for Example B, and the objective function is the same average wait time. To demonstrate the value of destination entry in our dynamic optimization, a window length of 0 s is tested. In addition, a window length of 30 s is tested to demonstrate the value of future traffic information.

Results obtained by using our standard method are summarized in Table X. With the 0-s window, the average wait time is 21.5 s, less than 23.6 s reported in [24]. This might be caused by different assumptions, our utilization of destination entry information, and our dynamic optimization as opposed to their steady-state analysis. With the 30-s window, the average wait time is 18.4 s with a standard deviation of 0.31 s. Both metrics are less than those obtained with the 0-s window, although the average CPU time of 3.3 s is higher than 2.6 s obtained with the 0-s window. This demonstrates that future traffic information is valuable to improve performance.

Example 5: This example considers coordinated emergency evacuation to demonstrate the potential of using elevators for building egress. There are four elevators whose parameters are taken from Table I. The overall down-peak traffic consists of 600 arrivals uniformly distributed in time and among floors in a 5-min interval. To minimize the overall egress time, this traffic is balanced between two stairs and the elevators by using a procedure such that the egress time via stairs is about the same as that via elevators [21]. The egress time via stairs is obtained by using probabilistic analysis with a given average passenger walking speed and stair width [21]. The window length is set to 0, with a rescheduling interval of 7.5 s. The elevator egress time is then obtained by using our method presented in Section VI, and it equals the overall egress time under the balanced traffic assumption.

Egress results obtained from ten Monte Carlo runs by using stairs alone, and by using elevators and stairs are summarized in Table XI. Without elevators, all the traffic is evacuated by using two stairs. In this case, the egress time of 872.3 s is considerably larger than 425.1 s for the case with elevators, demonstrating the significant potential of using elevators for emergency evacuation.

TABLE XI
RESULTS FOR EVACUATION WITH ELEVATORS AND WITHOUT ELEVATORS

Evacuation Policies		Without Elevators	With Elevators
Overall Egress Time (s)	Mean	872.3	425.1
	STDEV	0.99	2.08
Avg. CPU Time per Snapshot (s)		N/A	14.6

VIII. CONCLUSION

Advance information brings new opportunities to reduce uncertainty and improve efficiency. How to effectively model and utilize such information for group elevator scheduling remains an open and challenging issue. In this paper, key problem characteristics are abstracted to establish a two-level separable formulation. To handle this NP-hard problem, an approach has been developed by incorporating several innovative ideas into a decomposition and coordination framework. The approach has also been extended for cases with little or no future traffic information and coordinated emergency evacuation. Numerical testing results demonstrate near-optimal solution quality, computational efficiency, the value of future traffic information, and the potential of using elevators for emergency egress. Further improvement is needed to reduce CPU time for online implementation. Our study in optimized group elevator scheduling could have major impacts on configuring elevators for buildings. With optimized operation, a building may require fewer elevators to meet given performance specifications, and may lead to less floor space, equipment, and costs.

ACKNOWLEDGMENT

The authors would like to thank Dr. M. J. Atalla of UTRC for providing the problem context and invaluable suggestions, and Prof. C. G. Cassandras of Boston University and Mr. J. Sun of Tsinghua University for their constructive and insightful comments.

REFERENCES

[1] N. A. Alexandris, "Statistical models in lift systems," Ph.D. dissertation, Univ. Manchester, Inst. Sci. Technol. (UMIST), Manchester, U.K., 1977.

[2] K. Ashok and M. E. Ben-Akiva, "Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping to path flows and link flows," *Transp. Sci.*, vol. 36, no. 2, pp. 184–198, May 2002.

[3] G. C. Barney, *Elevator Traffic Handbook – Theory and Practice*. New York: Spon Press, 2003.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1995.

[5] M. Brand and D. Nikovski, "Optimal parking in group elevator control," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2004, vol. 1, pp. 1002–1008.

[6] W. L. Chan, A. T. P. So, and K. C. Lam, "Dynamic zoning in elevator traffic control," *Elevator World*, vol. 3, pp. 136–140, 1997.

[7] C. K. Christakos, "Sensor networks applied to the problem of building evacuation: An evaluation in simulation," in *Proc. 15th IST Mobile and Wireless Summit*, Jun. 2006, pp. 134–138.

[8] P. Cortes, J. Larraneta, and L. Onieva, "A genetic algorithm for controlling elevator group systems," in *Artificial Neural Net, Problem Solving Methods*. Berlín, Germany: Springer-Verlag, 2003, pp. 313–320.

[9] R. Crites and A. Barto, "Improving elevator performance using reinforcement learning," in *Advances in Neural Information Processing Systems 8*. Cambridge, MA: MIT Press, 1996, pp. 1017–1023.

[10] Elevate, Peters Research Ltd., 2002. [Online]. Available: www.elevate.peters-research.com

[11] Z. Gagov, Y. C. Cho, and W. H. Kwon, "Improved concept for derivation of velocity profiles for elevator systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 2419–2423.

[12] J. Gale, "Destination control and tower top access in Belgium," *Elevator World*, vol. 10, pp. 45–49, 2002.

[13] N. E. Groner and B. M. Levin, Human factors considerations in the potential for using elevators in building emergency evacuation NIST, Washington, DC, NIST-GCR-92-615, Jul. 1992.

[14] H. Hakonen, "Simulation of building traffic and evacuation by elevators," Licentiate thesis, Dept. Eng. Phys. Math., Helsinki Univ. Technol., Helsinki, Finland, 2003.

[15] D. Hauptmeier, S. O. Krumke, J. Rambau, and H. C. Wirth, "Euler is standing in line dial-a-ride problems with FIFO precedence constraints," *Discrete Appl. Math.*, vol. 113, pp. 87–107, 2001.

[16] Y. C. Ho, C. G. Cassandras, C. H. Chen, and L. Dai, "Ordinal optimization and simulation," *J. Oper. Res. Soc.*, vol. 51, no. 4, pp. 490–500, 2000.

[17] T. Inamoto, H. Tamaki, H. Murao, and S. Kitamura, "An application of branch-and-bound method to deterministic optimization model of elevator operation problems," in *Proc. 41st SICE Annu. Conf.*, 2002, pp. 987–992.

[18] J. Koehler and D. Ottiger, "An AI-based approach to destination control in elevators," *AI Mag.*, vol. 23, no. 3, pp. 59–78, Fall 2002.

[19] J. Koshak, "Elevator evacuation in emergency situations," in *Proc. Workshop on Use of Elevators in Fires and Other Emergencies*, Atlanta, GA, Mar. 2004, pp. 2–4.

[20] E. Kuligowski, "Elevators for occupant evacuation and fire department access," in *Proc. Int. Conf. Tall Buildings*, Malaysia, May 2003, pp. 8–10.

[21] P. B. Luh, L. Michel, E. Santos, Jr., D. Yu, A. See, B. Xiong, G. Johnson, and S. C. Chang, "Coherent configuration and operation of building transportation systems," in *Proc. IEEE Conf. Autom. Sci. Eng.*, 2005, pp. 178–184.

[22] D. Nikovski and M. Brand, "Decision-theoretic group elevator scheduling," in *Proc. 13th Int. Conf. Automated Planning and Scheduling ICAPS03*, Trento, Italy, 2003, pp. 133–142.

[23] A. C. Oghlan, S. O. Krumke, and T. Nierhoff, *"Average Case Analysis of a Hard Dial-a-Ride Problem,"*. Berlin, Germany: Konrad-Zuse-Zentrum für Informationstechnik, 2002.

[24] D. L. Pepyne and C. G. Cassandras, "Optimal dispatching control for elevator systems during up-peak traffic," *IEEE Trans. Control Syst. Technol.*, vol. 5, no. 6, pp. 629–643, Nov. 1997.

[25] D. L. Pepyne and C. G. Cassandras, "Design and implementation of an adaptive dispatching controller for elevator systems in up-peak traffic," *IEEE Trans. Control Syst. Technol.*, vol. 6, no. 5, pp. 635–649, Sep. 1998.

[26] B. Seckinger, "Synthesis of elevator controls based on constraint-based search," M.S. thesis, Albert Ludwig Univ., Freiburg, German, 1999.

[27] N. S. Shimbun, "Elevators Get Smart," RFID in Japan, Dec. 2004.

[28] R. Smith and R. Peters, "ETD algorithm with destination dispatch and booster options," *Elevator World*, vol. 7, pp. 136–142, 2002.

[29] A. T. P. So and S. K. Liu, "An overall review of advanced elevator technologies," *Elevator World*, vol. 6, 1996.

[30] G. R. Strakosch, *Vertical Transportation: Elevators and Escalators*. New York: Wiley., 1998.

[31] J. Sun, Q. Zhao, P. B. Luh, and M. Atala, "Estimation of optimal elevator scheduling performance," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1078–1083.

[32] B. Xiong, P. B. Luh, and S. C. Chang, "Group elevator scheduling with advanced traffic information for normal operations and coordinated emergency evacuation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 1419–1424.

[33] X. Zhao, P. B. Luh, and J. Wang, "The surrogate gradient algorithm for Lagrangian relaxation method," *J. Optim. Theory Appl.*, vol. 100, no. 3, pp. 699–712, Mar. 1999.

Peter B. Luh (S'77–M'80–SM'91–F'95) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1973, the M.S. degree in aeronautics and astronautics engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, in 1980.

Since 1980, he has been with the University of Connecticut, and currently is the SNET Professor of Communications and Information Technologies in the Department of Electrical and Computer Engineering, and a Visiting Professor at the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing, China. He is interested in planning, scheduling, and coordination of design, manufacturing, and supply chain activities, and schedule, auction, and portfolio optimization and load/price forecasting for power systems.

Dr. Luh is the founding Editor-in-Chief of the newly created IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, an Associate Editor of the IIE *Transactions on Design and Manufacturing*, an Associate Editor of *Discrete Event Dynamic Systems*, and was the Editor-in-Chief of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION (1999–2003).

Bo Xiong received the B.S. and M.S. degrees in automation engineering from Shanghai Jiaotong University, China, in 2000 and 2003, respectively. He is currently working towards the Ph.D. degree in electrical and computer engineering at the University of Connecticut, Storrs.

His main research interests are in discrete-event simulation and optimization theory, with applications in supply chain management, inventory control, and elevator dispatching.

Shi-Chung Chang (S'83–M'87) received the B.S.E.E. degree from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1979, and the M.S. and Ph.D. degrees in electrical and systems engineering from the University of Connecticut, Storrs, in 1983 and 1986, respectively.

Since 1994, he has been with the Department of Electrical Engineering, National Taiwan University. His research interests include optimization theory and algorithms, production scheduling and control, network management, Internet economics, and distributed decision making.