Optimization of Group Elevator Scheduling With Advance Information

Jin Sun, *Student Member, IEEE*, Qian-Chuan Zhao, *Senior Member, IEEE*, and Peter B. Luh, *Fellow, IEEE*

***Abstract*—Group elevator scheduling has received considerable attention due to its importance to transportation efficiency for mid-rise and high-rise buildings. One important trend to improve elevator systems is to collect advance traffic information. Nevertheless, it remains a challenge to develop new scheduling methods which can effectively utilize such information. This paper is to solve the group elevator scheduling problem with advance traffic information. This problem is difficult due to various traffic patterns, complicated car dynamics, and combinatorial explosion of the search space. A two-level formulation is developed with passenger-to-car assignment at the high-level and single car dispatching that is innovatively formulated as passenger-to-trip assignment at the low-level. Detailed car dynamics are embedded in simulation models for performance evaluation. Taking advantage of advance information, a new door action control method is suggested to increase the flexibility of elevators. In view of the hierarchical problem structure, a two-level optimization framework is established. Key problem characteristics are exploited to develop an effective trip-based heuristic for single car dispatching, and a hybrid nested partitions and genetic algorithm method for passenger-to-car assignment which can be extended to solve a generic class of sequential decision problems. Numerical results demonstrate solution quality, computational efficiency, benefit of advance information and the new door action control method, and values of new features in our hybrid method.**

***Note to Practitioners*—This paper is motivated by the needs to develop new elevator scheduling methods that can make effective use of advance traffic information. A novel two-level formulation is developed, with detailed car dynamics embedded in simulation models for performance evaluation. Taking advantage of advance information, a new door action control method is suggested for car dynamics to increase the flexibility of elevators. Key problem characteristics are exploited to develop an effective two-level optimization framework, where the high-level solution method can be extended to solve a generic class of sequential decision problems. Numerical results demonstrate values of advance information and the new door action control method, and effectiveness of our solution method. Further improvement is needed to reduce CPU time for online implementation.**

***Index Terms*—Advance traffic information, destination entry, genetic algorithm, group elevator scheduling, nested partitions.**

I. Introduction

GROUP elevator scheduling is important to transportation efficiency for mid-rise and high-rise buildings, and how to improve the service quality of elevators has received considerable attention. In conventional elevator systems, only up and down buttons are available for hall calls, and passengers cannot specify their destinations until they enter the elevators. The systems need to make decisions in the presence of uncertainties on passenger arrival times and destinations [1]. Such decisions are sometimes unsatisfactory. For instance, passengers often have a long wait for the next elevator because they missed an elevator that left a few seconds previously; passengers often have to wait for the door to close even if no one is going to board. The reason why these two phenomena arise is that due to lack of traffic information, the elevator systems have to rely on door dwell time, the minimum time interval to keep the door open, to decide when to close the door. To cope with traffic uncertainties, advanced technologies have been introduced to collect and predict traffic information. In a Destination Entry system, passengers can enter their destinations through keyboards before they get into the cars [2]. For these systems, passenger arrival times, origins, and destinations are known before the systems make decisions. The latest advancements in sensor technology and information technology further open up the possibility to collect future traffic information within a certain time window [3]–[6]. Beyond such information, statistical data and/or statistical forecasts have been exploited to create virtual future passenger traffic [7]–[9]. More information enables the development of elevator systems with better performance. Nevertheless, traditional elevator scheduling methods do not have mechanisms to utilize advance information collected, and it remains an open and challenging issue to develop new ones which can effectively utilize such information. This problem is difficult because of various traffic patterns, complicated car dynamics, and combinatorial explosion of the search space.

After a review of related literature in Section II, the problem formulation is presented in Section III. Advance information has been modeled by assuming that traffic information (i.e., passenger arrival times, their origins and destinations) within a look-ahead time window is fully available. A rolling horizon

Manuscript received January 04, 2008. First published August 18, 2009; current version published April 07, 2010. This paper was recommended for publication by Associate Editor D. Ghose and Editor Y. Narahari upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 60274011, Grant 60574067, Grant 60721003, and Grant 60736027, in part by the Program for New Century Excellent Talents in University of China under Grant NCET-04-0094, and in part by the National 111 International Collaboration Project of China, and by the China Scholarship Council.

J. Sun was with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs Mansfield, CT 06269-2157 USA. He is now with the Center for Intelligent and Networked Systems, Department of Automation, TNList, Tsinghua University, Beijing 100084, China (e-mail: sunjin00@mails.tsinghua.edu.cn).

Q.-C. Zhao is with the Center for Intelligent and Networked Systems, Department of Automation, TNList, Tsinghua University, Beijing 100084, China (e-mail: zhaoqc@tsinghua.edu.cn).

P. B. Luh is with the Center for Intelligent and Networked Systems, Department of Automation, TNList, Tsinghua University, Beijing 100084, China and also with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs Mansfield, CT 06269-2157 USA (e-mail: peter.luh@uconn.edu) .

Digital Object Identifier 10.1109/TASE.2009.2024242

scheme is then used to move the time window forward to construct snapshot problems periodically or as needed [6]. By exploiting the two-level concept in [10] and [6] for each snapshot, a two-level integer formulation is established with passenger-to-car assignment at the high-level and single car dispatching at the low-level. Based on a key concept "trip" which denotes the car movement in a single direction, a single car dispatching strategy is innovatively represented as a passenger-to-trip assignment. This representation abstracts key ingredients of detailed passenger service process, separates them from car dynamics, and reduces modeling complexity. The detailed car dynamics are embedded in individual car simulation models only for performance evaluation. Taking advantage of advance information, a new door action control method without relying on door dwell time is suggested for car dynamics to provide more flexibility for elevators, making them more intelligent and efficient. Since the car trajectory can be recovered through simulation models, the objective function is flexible within a large range of car wise or passenger-wise (which can be turned into car-wise) additive measures.

A two-level optimization framework is developed for each snapshot problem in Section IV based on the hierarchical problem structure and the car-wise additive property of the objective function. Given a high-level passenger-to-car assignment, the low-level is to optimize single car dispatching for individual cars and return the optimized results for the evaluation of the given high-level assignment. The high-level is then to optimize passenger-to-car assignment based on the evaluation from the low-level. Key problem characteristics are innovatively exploited to develop efficient algorithms for both levels. At the low-level, an effective trip-based heuristic is developed on the basis of an observation on the optimal dispatching strategy to sequentially decide passengers to be served in each trip for individual cars. At the high-level, an innovative method is developed by exploiting the sequential decision nature of elevator scheduling and innovatively embedding genetic algorithm into the "*local optimization*" and "*global verification and correction*" framework of nested partitions. This hybrid nested partitions and genetic algorithm method (HNPGA) is also suitable to solve a class of sequential decision problems with certain characteristics.

Extensive testing has been conducted in Section V. Near-optimal solutions are obtained in a timely fashion for various traffic patterns (i.e., up-peak, down-peak, and interfloor) and different traffic densities. Advance information and the new door action control method can significantly improve solution quality. The new features of the hybrid method are also shown to be valuable in improving the solution quality and reducing the computational time.

II. Literature Review

For conventional elevator systems without destination entry systems, extensive research has been conducted. Many simple heuristic approaches have been developed, such as collective control in which a car stops to serve the nearest call in its current movement direction, the longest queue first, and the highest unanswered floor first [11]–[13]. Although these methods are generally computationally efficient, they can only generate good performance for specific traffic patterns. To deal with various traffic patterns, zoning approaches were developed, where each car is assigned a number of floors grouped together as a zone [12], [14]. While the zoning approaches are robust in heavy traffic, they lose a significant amount of flexibility [15]. Search-based approaches are developed to dynamically explore possible car assignments to optimize certain criteria such as the average passenger waiting time. These approaches includes greedy search strategies which perform immediate call assignment [16], [17], and non-greedy search strategies which postpone passengers' assignments for batch optimization or consider reassignment of passengers by using updated traffic information [18]–[20]. Greedy strategies require less computation time, whereas non-greedy ones enjoy more flexibility of elevator control. Advanced intelligent technologies such as expert system [21], fuzz logic [22], artificial neural network [23], and reinforcement learning [15] have been used to develop intelligent elevator scheduling methods. These methods generally require rich experiences from professionals or huge offline training efforts to achieve good results. The structure of the optimal control policy to minimize the discounted or average passenger waiting time for up-peak traffic has been studied in [1]. The elevator system is modeled as a queuing system by using stochastic processes to describe the passenger arrival process and service process, and the scheduling problem is modeled as a Markov decision problem. Dynamic programming is used to derive the optimal policy.

Only a few papers have studied destination entry elevator systems. The "Estimated Time to Destination" (ETD) method performs immediate call assignment to optimize the total passenger service time by considering both the time for each elevator to serve the new call and the impact of the new allocation on all other passengers in the system [24]. Although this method is computationally efficient, the performance is not that good due to its greedy nature. Inamoto *et al.* [10] and Luh *et al.* [6] study the case with future traffic information available in a certain time window. Individual cars are coupled through serving a common pool of passengers. Once the passenger-to-car assignment is fixed, different cars are no longer coupled. Exploiting this "separable" structure, a two-level formulation can be established, with passenger-to-car assignment at the high-level and single car dispatching at the low-level. In [10], the operation of carrying a passenger from origin to destination is divided into two jobs: on-job and off-job. Then, a job-based two-level formation is established, with assigning passengers to cars at the high-level and determining the processing orders of jobs for each car at the low-level. Nevertheless, the formulation is complicated due to numerous time-related continuous and integer variables, and several special assumptions are imposed on car dynamics and traffic profiles. The Branch and Bound method is used to generate the optimal performance. This method, however, is very time-consuming for larger problems due to its inherent complexity. In [6], the detailed car dynamics are embedded in individual car simulation models. A decomposition and coordination approach is developed to exploit the separable nature of this problem, where subproblems are solved by ordinal optimization-based local search, and top ranked nodes are selectively optimized by using dynamic programming. Special

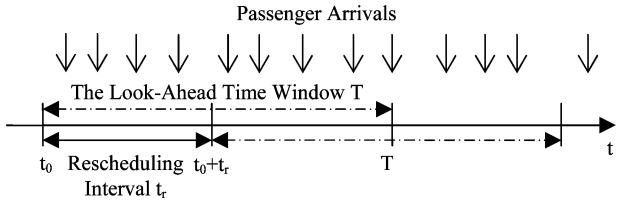

Fig. 1. Rolling horizon scheme.

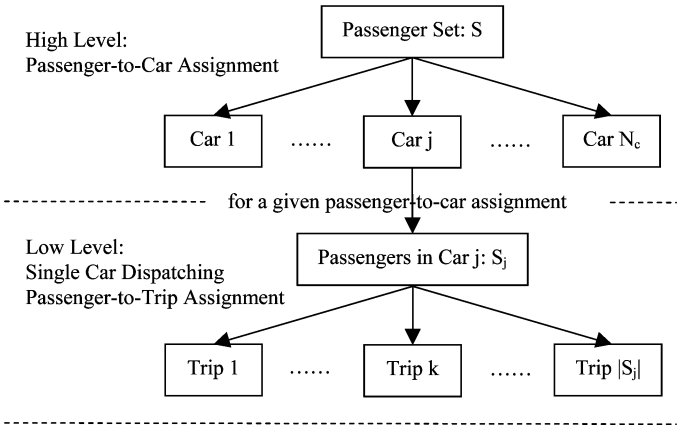

Fig. 2. Two-level problem formulation.

assumptions including no-wait and first-come-first-served are made to reduce the search space of dynamic programming in the expense of the solution quality.

III. Problem Formulation

Consider a building with F floors and a group of N_c cars. The future traffic information is modeled by using a look-ahead time window of length T, where traffic information within the window is assumed available, and ignored otherwise. A rolling horizon scheme is used to shift the time window to construct snapshot problems periodically or as needed, as shown in Fig. 1 [6]. At the current scheduling point t_0, passengers to be considered, denoted by S, includes those who are already inside the cars, those who are still waiting, and those who arrive between t_0 and $t_0 + T$. Those passengers are sorted in the ascending order of their arrival times. For passenger $i(1 \leq i \leq |S|)$, the arrival time t_i^a, the origin floor f_i^a, and the destination floor f_i^d are assumed given.

Following the two-level concept in [10] and [6], a two-level integer programming formulation is established for each snapshot problem, as shown in Fig. 2, with the high-level passenger-to-car assignment in Section III-A, and the low-level passenger-to-trip assignment for single car dispatching in Section III-B. The car dynamics are presented in Section III-C, followed by objective function and overall formulation in Section III-D.

A. High-Level: Passenger-to-Car Assignment

The high-level decision variable is the passenger-to-car assignment, defined as an $|S| \times N_c$ matrix of binary variables, where the $(i,j)_{th}$ element δ_{ij} equals 1 if passenger i is assigned to car j, and 0, otherwise. To guarantee the decision variable feasible, the following constraints should be satisfied.

Passenger-to-Car Assignment Constraints: Each passenger must be assigned to one and only one car, i.e.,

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

For passengers who are already inside the cars before t_0, their $\{\delta_{ij}\}$ are fixed. Assignment decisions are only made on those who have not been picked up.

B. Low-Level: Single Car Dispatching

Given the high-level passenger-to-car assignment $\{\delta_{ij}\}$, the low-level is to solve for each car the single car dispatching problem, which is in essence to determine the service sequence (i.e., loading/unloading) of all the passengers assigned to the same car [10]. To facilitate the definition of the low-level decision variables, a concept "trip" is introduced to describe the car movement in a single direction (i.e., up or down). Then, the entire car trajectory can be divided into multiple trips in which passengers are sequentially served. It is assumed that an elevator will not change its directions until all the passengers inside are transported to their destinations, similar to most of the current elevator systems. Then, the loading and unloading of one passenger will be in the same trip. Once the passengers served in one trip are determined, the service sequence of those passengers can be uniquely determined. In an up (down) trip, the car will travel up (down) to sequentially load passengers that assigned to this trip in the increasing (decreasing) order of their origin floors and the increasing order of their arrival times at each origin floor, and unload them at their corresponding destination floors. At each stop floor, all the unloading services come before the loading services. Further, with a given passenger-to-trip assignment, the service sequence of all the passengers can be uniquely determined simply by sequentially connecting the service sequences of individual trips. Therefore, the decision variable of the low-level can be represented as the passenger-to-trip assignment. This representation captures key ingredients of detailed passenger service process, separates them from car dynamics, and reduces modeling complexity. It should be noted that the car movement between two adjacent nonempty trips could be arbitrary, i.e., a single empty trip or multiple empty trips. The most efficient one should be the single empty trip connecting the last stop floor of the former nonempty trip and the first stop floor of the latter nonempty trip.

Let S_j denote the set of passengers assigned to car j: $\{i|\delta_{ij} = 1\}$. Since each nonempty trip will serve at least one passenger, at most $|S_j|$ nonempty trips are required to serve S_j. Then, the passenger-to-trip assignment for S_j is defined as an $|S_j| \times |S_j|$ matrix of binary variables, where the $(i,k)_{th}$ element ζ_{ik} equals 1 if passenger $i \in S_j$ is assigned to trip k, and 0, otherwise. Regardless of the number of empty trips between two nonempty trips, after serving the former nonempty trip, the car will immediately move to the first stop floor of the latter nonempty trip for improving service efficiency. To guarantee these variables feasible, the following constraints should be satisfied.

Passenger-to-Trip Assignment Constraints: Each passenger must be assigned to one and only one trip, i.e.,

$$\sum_{k=1}^{|S_j|} \zeta_{ik} = 1, \quad \forall i \in S_j. \tag{2}$$

Same Direction Constraints: By definition of "trip," the passengers assigned to the same trip must travel in the same direction. Let S_j^d and S_j^u denote the set of passengers who travel down, $\{i \in S_j | f_i^a > f_i^d\}$, and the set of passengers who travel up, $\{i \in S_j | f_i^a < f_i^d\}$, respectively. Then

$$\sum_{i \in S_j^d} \zeta_{ik} \times \sum_{i \in S_j^u} \zeta_{ik} = 0, \quad \forall k. \tag{3}$$

The two terms in the left-hand side denote the number of passengers who travel down and the number of passengers who travel up in trip k. Equation (3) means that one of the two numbers must be zero, implying that all the passengers are in the same direction.

Capacity Constraints: At any time, the number of passengers in the car must be less than or equal to the capacity of the car. Let S_j^f denote the set of passengers in car$j$ when the car travels between floor f and floor $f+1$, including those passengers who board the car under floor $f+1$ and leave the car above floor f, $\{i \in S_j | (f_i^a < f+1, f_i^d > f)\}$, or board the car above floor f and leave the car under floor $f+1$, $\{i \in S_j | (f_i^a > f, f_i^d < f+1)\}$. Then

$$\sum_{i \in S_j^f} \zeta_{ik} \le C_j, \quad \forall f, k. \tag{4}$$

The left-hand side denotes the number of passengers in car j when the car travels between floor f and floor $f+1$ in trip k. The right-hand side C_j denotes the capacity of car j.

C. Car Dynamics

The high-level decision variable $\{\delta_{ij}\}$ determines passengers to be served in individual cars $\{S_j\}$. For passengers in S_j, the low-level decision variable $\{\zeta_{ik}\}_{i \in S_j}$ determines the service sequence of these passengers. Given S_j and $\{\zeta_{ik}\}_{i \in S_j}$, a sequence of stop floors and the passengers to be picked up or dropped off at each stop floor can be obtained, as mentioned in Section III-B. The car movement between two stop floors and the car action at a stop floor are determined by car dynamics $g(\cdot)$.

In the traditional car dynamics, once the car reaches a stop floor, it will first open the door, unload/load passengers, and keep the door open until the dwell time expires, as depicted in Fig. 3. Here, the dwell time represents the minimum amount of time that the door should stay open after fully opened, and functions as a control variable to accommodate uncertain future arrivals. Nevertheless, once the passenger-to-trip assignment is given, the passengers to be loaded and/or unloaded at one stop floor are fixed, and there is no need to rely on door dwell time to control the door action. A new door action control method is then suggested by removing the dwell time to provide more flexibility to the behavior of elevators: the car could leave the stop floor immediately even if the dwell time is not expired, as long as the car finishes all the loading/unloading tasks at this floor; and the car could stay and wait for passengers to be served even if the dwell time is expired. In this way, elevators would be more intelligent and efficient, thus achieving better performance.

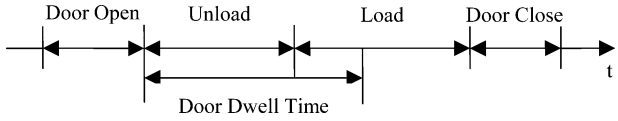

Fig. 3. Traditional car dynamics.

By adopting either kind of car dynamics mentioned above, the car trajectory can be recovered, and then pickup times t_i^p's and departure times t_i^d's can be obtained, i.e.,

$$\{t_i^p, t_i^d\}_{i \in S_j} = g(S_j, \{\zeta_{ik}\}_{i \in S_j}). \tag{5}$$

Since car dynamics (5) are too complicated to be expressed explicitly, they are embedded in individual car simulation models. Based on those simulation models, a large number of passenger-wise and car-wise performance measures (e.g., average passenger waiting time, average passenger service time, etc.) can be derived. Then, the objective function is rather flexible with respect to such choices.

D. Objective Function and Overall Formulation

The usual objective to measure passenger satisfaction is weighted sum of average passenger waiting time and average passenger transit time. For passenger i, the waiting time is the time interval between the arrival time t_i^a and the pickup time t_i^p, and the transit time is the time interval between the pickup time t_i^p and the departure time t_i^d. The objective function is as follows:

$$\begin{aligned} J &= \frac{1}{|S|} \sum_{i=1}^{|S|} \left(\alpha \left(t_i^p - t_i^a \right) + \beta \left(t_n^d - t_n^p \right) \right) \\ &= \frac{1}{|S|} \sum_{j=1}^{N_c} J_j \end{aligned} \tag{6}$$

where

$$J_j = \sum_{i \in S_j} \left(\alpha \left(t_i^p - t_i^a \right) + \beta \left(t_n^d - t_n^p \right) \right). \tag{7}$$

Equations (6) and (7) imply that all the passenger-wise additive objective functions can be rewritten in car-wise additive forms by reorganizing the original form based on the passenger-to-car assignment. When the coefficients α and β both equal 1, the performance will be average passenger service time.

The overall problem is to minimize J (6) subject to passenger-to-car assignment constraints (1), passenger-to-trip assignment constraints (2), same direction constraints (3), and capacity constraints (4). The decision variables are the passenger-to-car assignment and the passenger-to-trip assignment for each car. It can be seen that our formulation is an integer programming formulation. The scale of our formulation (i.e., the number of decision variable and constraints) is much smaller than the two existing formulations. Moreover, the formulation above applies to various traffic patterns like the formulation in [6], since no specific assumptions are made about them.

IV. Solution Methodology

In view of the hierarchical problem structure and the car-wise additive property of the objective function, a two-level optimization framework is developed in Section IV-A, optimizing single car dispatching at the low-level, and optimizing passenger-to-car assignment at the high-level. Key problem structures are utilized to develop efficient algorithms for both levels. At the low-level, an effective trip-based heuristic is introduced in Section IV-B to optimize the single car dispatching problems for individual cars. At the high-level, a hybrid nested partitions and genetic algorithm method is developed in Section IV-C to optimize passenger-to-car assignment.

A. Two-Level Optimization Framework

Taking advantage of the hierarchical problem structure and the car-wise additive objective function, the original problem formulation could be rewritten as follows:

$$\min J = \min_{\{\delta_{ij}\},\{\{\zeta_{ik}\}_{i\in S_j}\}_j} \frac{1}{|S|}\sum_{j=1}^{N_c} J_j(S_j,\{\zeta_{ik}\}_{i\in S_j}) = \min_{\{\delta_{ij}\}} \frac{1}{|S|}\sum_{j=1}^{N_c} \min_{\{\zeta_{ik}\}_{i\in S_j}} J_j(S_j,\{\zeta_{ik}\}_{i\in S_j}) \quad (8)$$

subject to (1)–(4).

Since constraints (2)–(4) are only imposed on low-level decision variables, $\{\zeta_{ik}\}_{i\in S_j}$, the formulation above can be further converted into[1]

$$\min J = \min_{\{\delta_{ij}\}} \frac{1}{|S|}\sum_{j=1}^{N_c} J_j^*(S_j), \text{ subject to } (1) \quad (9)$$

where

$$J_j^*(S_j) = \min_{\substack{\{\zeta_{ik}\}_{i\in S_j} \\ \text{s.t.,(2),(3),(4)}}} J_j(S_j,\{\zeta_{ik}\}_{i\in S_j}). \quad (10)$$

From (9) and (10), a two-level optimization framework can be naturally derived. Given a high-level decision variable $\{\delta_{ij}\}$, i.e., $\{S_j\}_j$, the low-level optimizes the single car dispatching problems (10) for individual cars and provide the optimized performance to the high-level for the calculation of the objective function value in (9). Based on the evaluation from the low-level, the high-level is then to optimize passenger-to-car assignment (9).

B. Low-Level: Optimizing Single Car Dispatching via a Trip-Based Heuristic

To effectively solve the single car dispatching problem (10), a trip-based heuristic is developed in [27] to sequentially decide passengers to be served in each trip. The key idea is that the optimal dispatching strategy should be a tradeoff between two factors: 1) serving more passengers in one trip to take advantage of batch effect and 2) serving fewer passengers in one trip to avoid extra long waiting of passengers who arrive later. To determine which passengers to be served in the next trip (trip k), a list of passenger assignment candidates will first be selected from unassigned passengers. For each candidate $\{\zeta_{ik}\}_i$, constraints (3) and (4) can be easily verified, as the terms in the left-hand side of these two inequalities, i.e., the number of passengers who travel down, S_j^d, the number of passengers who travel up, S_j^u, and the number of passengers inside the car between any two floors, S_j^f, can be easily calculated. The candidates that satisfy both constraints will be evaluated by using a carefully designed objective which takes the two above factors into account. The assignment candidate with the minimum performance is selected as the next trip. The process repeats until the trip assignment of all the passengers is determined, implying constraint (2) is satisfied. Therefore, the passenger-to-trip assignment generated will be feasible. Based on car dynamics (5), the corresponding performance of the passenger-to-trip assignment can be obtained.

Given a high-level passenger-to-car assignment, the low-level will solve N_c individual single car dispatching problems and return the sum of the optimized performance values (divided by $|S|$) as the performance of the given high-level decision variable.

C. High-Level: Optimizing Passenger-to-Car Assignment via a Hybrid Nested Partitions (NPs) and Genetic Algorithm Method

On the basis of the evaluation from the low-level, the high-level is to optimize the passenger-to-car assignment by using nested partitions (NPs) which has been proved to be powerful for many difficult optimization problems including the assignment problem [28]. The key idea of this method is to systematically partition the feasible decision space or region into subregions, to identify the most promising subregion that is most likely to contain the optimal solution through sampling, and then to concentrate the computational efforts on this subregion. As the iterations proceed, the most promising region is gradually reduced by further partitioning, along with adjustment by backtracking if needed. "This method is simple, robust," and "converges to the global optimum in finite time with probability one" [28].

1) Traditional Nested Partitions Method: To illustrate the application of the traditional NP method at the high-level, an example with $N_c = 4$ elevators and $|S| = 12$ passengers (Example 2 in Section V) is used in the following. Nested partitions performs search in the entire feasible region Θ, i.e., all the $|S| \times N_c$ (12×4) binary matrices subject to passenger-to-car assignment constraints (1). To facilitate NP's application, each feasible passenger-to-car assignment matrix will be represented as a vector of length $|S| = 12$, where element i $(1 \le i \le 12)$ equals j $(1 \le j \le 4)$ if passenger i is assigned to car j. This representation automatically satisfies passenger-to-car assignment constraints (1), turns the original constrained search problem into an unconstrained one without enlarging the search space, and consequently helps improve search efficiency. Then, the feasible region Θ is equivalent to the set of all the vectors of

[1]This formulation is actually a bi-level programming formulation [25], [26]. The current research in bi-level programming mainly concentrates on solving linear problems. Most of the existing methods make use of the linearity of the problems, and hence cannot be extended to our case

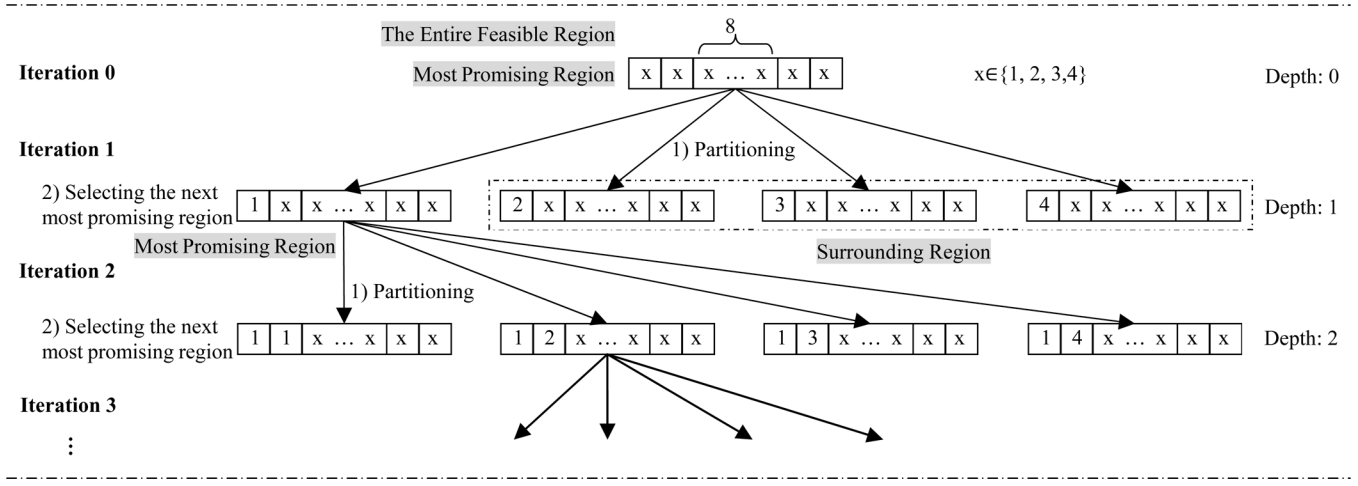

Fig. 4. Search tree of the traditional nested partitions method for a problem with four elevators and 12 passengers.

length 12, where each element can only take a value in the set $\{1, 2, 3, 4\}$.

Refer to Fig. 4 for the searching tree of the traditional NP method. In each iteration, there is a region $\sigma \subseteq \Theta$ called the "*most promising region*" where the optimal solution is most likely to locate, e.g., the entire feasible region $\{(x, \ldots, x)_{12\times1} | x \in \{1, 2, 3, 4\}\}$ at iteration 0 (since there is no information about the location of good solutions) and the region $\{(1, x, \ldots, x)_{12\times1} | x \in \{1, 2, 3, 4\}\}$ at iteration 1. The remaining region $\Theta\backslash\sigma$ is the "*surrounding region*." Each most promising region is associated with a parameter "*depth*" defined as the number of partitioning steps to obtain the region σ starting from the entire feasible region Θ without backtracking, e.g., the depths of the two aforementioned most promising regions are 0 and 1. The following two steps are performed at iteration k.

Step 1: Partitioning: The most promising region σ (k) is first partitioned into M_k subregions by using a partitioning scheme which decides how the region will be divided, e.g., partitioning the feasible region into four subregions by fixing the car assignment of the first passenger at iteration 1. Together with the surrounding region, there are M_k+1 disjoint subsets that cover the entire region.

Step 2: Selecting the Next Most Promising Region: Each of these $M_k + 1$ regions is sampled by using a random sampling scheme such as uniform sampling, and the performance values at those randomly selected points are used to estimate the promising index for each region, defined as the minimum performance value of all the samples within the region. Local search such as genetic algorithm can improve the estimation of the promising indices by using the samples as initial points and trying to find new samples with better performance [29]. If one of the M_k subregions has the best promising index, then it becomes the most promising region in the next iteration, $\sigma(k+1)$, and the remaining M_k-1 regions and the surrounding region are aggregated as the new surrounding region. If the surrounding region has the best promising index, the method backtracks to a new promising region $\sigma(k + 1)$ with a fixed backtracking rule. The region $\Theta\backslash\sigma(k + 1)$ becomes the next surrounding region.

The process repeats until the most promising region becomes a singleton or the computational time reaches its limit.

How to partition the most promising region and how to select the next most promising region are two important components of the nested partitions method. To make this method efficient, key problem characteristics of elevator scheduling will be incorporated in the design of these two components to develop a novel hybrid nested partitions and genetic algorithm method (HNPGA) next.

2) Nested Partitions and Genetic Algorithm Method:

a) Defining the Partitioning Scheme: The partitioning scheme determines how the most promising region will be partitioned. In view of the sequential decision nature of elevator scheduling [15], a natural partitioning step is to fix the assignment of the next N unassigned passenger. Then, a promising region σ of depth k is defined by fixing the assignment of the first $N \times k$ passengers. The subregions of this region are determined by assigning the next N unfixed passengers to the N_c cars, so there will be N_c^N subregions. Consider again the example in Section C1 for illustration. A straightforward choice of N is one, as shown in Fig. 4. Nonetheless, since generally the entire performance is not sensitive to the change of the assignment of only one passenger, this partitioning scheme may lead to difficulties in differentiating those subregions. Therefore, it will be difficult to find the subregion most likely to contain good solutions, leading to frequent backtracking, and thus low efficiency, as will be demonstrated in Fig. 5 in Section V. To avoid such difficulty, the partitioning scheme should be able to generate subregions with large performance variations so that NP can easily separate good subregions from bad subregions. Therefore, compared to the number of all the passengers $|S|$, N should be large enough to impact the entire performance, but not too large to avoid heavy computational burden for next step.

b) Identifying the Next Most Promising Region: With the given partitioning scheme, the next is to estimate the promising index for each subregion and to select the next most promising region based on those indices. The comparison of all the subregions and the surrounding region could be divided into two substeps: 1) selecting the best subregion $\sigma_b(k)$, which can be

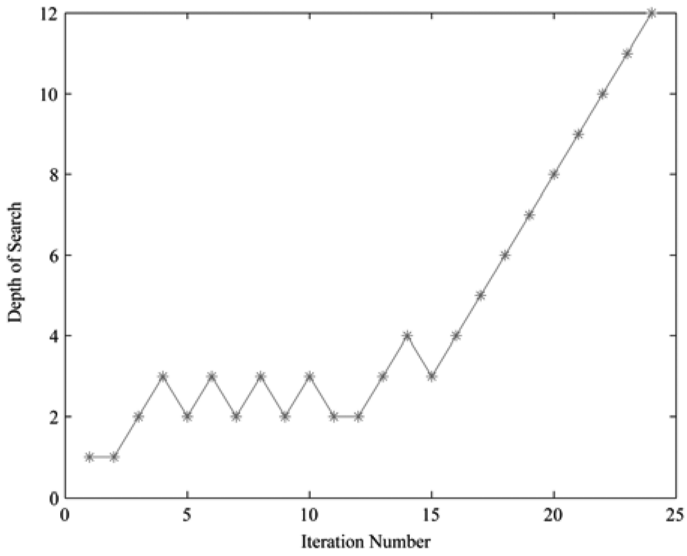

Fig. 5. Depth during the search in TNP.

viewed as "*local optimization*" and 2) comparing the best subregion $\sigma_b(k)$ with the surrounding region $\Theta\backslash\sigma(k)$, which can be viewed as "*global verification and correction*." Both steps are innovatively transformed to the optimization of the assignment of a certain number of passengers, and will be solved by genetic algorithm to take advantage of the "building block" property [30] of elevator scheduling, i.e., two assignments with good blocks/segments for different subgroups of passengers can be combined to generate a better assignment. For ease of presentation, the genetic algorithm based search is presented first, followed by the two steps to identify the most promising region.

C.2.2.1. Genetic Algorithm Based Search

Genetic algorithm is an important module in our hybrid nested partitions and genetic algorithm method. It is used twice in each iteration of NP, one for selecting the best subregion in C.2.2.2, and the other for the comparison of the best subregion and the surrounding region in C.2.2.3. For both cases, it is used to optimize the assignment of a group of $N_{\rm GA}$ passengers.

The chromosome is defined as a vector of length $N_{\rm GA}$, where element $i\,(1 \le i \le N_{\rm GA})$ equals to $j(1 \le j \le N_c)$ if passenger i is assigned to car j. The consistence between the definition of the chromosome and the new representation in NP as defined in C.1 makes genetic algorithm capable of being integrated into the nested partitions framework perfectly. The fitness of each chromosome is defined as the performance of the corresponding assignment (6).

Two kinds of mutation operators are designed: 1) random change of the car assignment of one passenger and 2) random swap of the car assignments of two passengers. The first mutation can balance the number of passengers assigned to each car. The second mutation can fine tune the group of passengers served by one car, while keeping the number of passengers unchanged. The standard single point crossover operator is used. Two assignments with good assignment segments for different subgroups of passengers can be combined by crossover to generate a better assignment. By applying these operators, any feasible passenger-to-car assignment could be achieved with a positive probability from another feasible one. Therefore, the ergodicity of the whole feasible solution space is guaranteed. Moreover, the new assignments generated by these operators are always feasible. This feature keeps the search within the feasible solution space and then improves search efficiency.

The main steps of genetic algorithm are as follows.

Step 1) Initialize population with randomly generated feasible passenger-to-car assignments and predetermined ones such as good solutions from previous iterations. Evaluate individual assignment's fitness.

Step 2) Expand population through mutation and crossover.

Step 3) Select the next population based on fitness.

Step 4) Repeat Steps 1 to 3 for a certain number of times.

C.2.2.2. Selecting the Best Subregion

As mentioned in Section IV-C1, in the traditional NP method, a group of N_p samples will be taken in each of the N_c^N subregions, and starting with those samples, genetic algorithm will run for N_g generations to find good solutions which can represent the performance (promising index) of the subregion. The subregion containing the best sample in the N_c^N final populations will be selected as the best subregion $\sigma_b(k)$. Since the number of subregions N_c^N is large, this substep could be time-consuming. Based on the convergence proof of NP, the only requirement for sampling is that each point in the entire solution space has a positive probability to be selected. Therefore, there is no need to take samples in each subregion. Instead, N_p samples will be taken uniformly from the composite region of all the subregions, i.e., the current most promising region $\sigma(k)$, so that each point could be selected with a certain positive probability. Starting with those samples, genetic algorithm in C.2.2.1 will run for N_g generations to find good solutions in the current most promising region, i.e., to optimize the assignment of the remaining $(|S| - N \times k)$ unassigned passengers. The subregion containing the best sample in the final population of genetic algorithm is selected as the best subregion $\sigma_b(k)$. Compared with traditional NP, the new best subregion selection substep in our method would result in a great reduction of CPU time with little expense of solution quality, as demonstrated in Example 2 in Section V.

At iteration k, this substep is to decide the assignment of the next N passengers based on the optimization of the remaining $(|S| - N \times k)$ unassigned passengers. It is clear that the dispatching of passengers who arrive early will affect passengers who arrive later and the impact decreases as the time interval between their arrivals increases. This observation leads to the idea that it might be sufficient to use part of future traffic information to effectively assign current passengers. It should be noted that decision made by using partial traffic information might not be consistent with the one from full information. Thus a further improvement is made in this substep. Only the assignment of the next N_t passengers is optimized by genetic algorithm to decide the assignment of the N passengers, where N_t is an integer between N and $(|S| - N \times k)$. Consequently, the search of genetic algorithm will be more efficient, as it is performed in a smaller space. The solutions generated by genetic algorithm are partial solutions in the sense that only the assignment of N_t out of the remaining $(|S| - N \times k)$ unassigned passengers are fixed in those solutions. The best partial solution in the final population is used to determine the best subregion. This substep can

be viewed as "*local optimization*" in the sense that the decision on the assignment of the next N passengers is made based on local information (the optimization of the assignment of the next N_t passengers). Due to the possible inconsistency between decisions made by using partial information and full information, the selection of the best subregion might be biased, as will be addressed in next.

C.2.2.3. Comparing the Best Subregion with the Surrounding Region

To compare the best subregion $\sigma_b(k)$ and the surrounding region $\Theta\backslash\sigma(k)$, samples from the two regions will be compared. The samples from $\sigma_b(k)$ are constructed by fixing the assignment of the next N_t passengers to be the best partial solution obtained in C.2.2.2 and randomly generating the assignment of the remaining $(|S| - N \times k - N_t)$ passengers. The samples from the surrounding region are taken by uniform sampling. It should be noted that the region $\sigma(k)\backslash\sigma_b(k)$ is also demanded to be sampled to avoid the negative impact from biased selection, in order that the true best subregion still has a certain positive probability to be selected. The samples from the three regions form the initial population of genetic algorithm in C.2.2.1, and genetic algorithm performs search in the composite region of the three regions, i.e., the entire feasible solution. The best solution in the final population of genetic algorithm is used to identify the next most promising region $\sigma(k+1)$ as follows. If the best solution lies within the current most promising region $\sigma(k)$, then the subregion containing this solution is selected as $\sigma(k+1)$. If the best solution lies within the region $\Theta\backslash\sigma(k)$, this method needs to backtrack to choose a region with depth less than k and containing the best solution as $\sigma(k+1)$, e.g., a region where all the solutions have the same assignment of the first $(k-1)$ passengers as that in the best solution[2]. This substep can be viewed as "*global verification and correction*" in the sense that local decision in C.2.2.2 will be verified and corrected by using global information (the optimization of the assignment of all the passengers).

c) Convergence Property: The convergence proof of the traditional NP method requires that "the stochastic process $\{\sigma(k)\}_k$ is a Markov chain" [28], implying no information in the current iteration can be passed down to the next. All the samples from previous iterations should be discarded, and new samples should be regenerated for the current iteration independently of previous iterations. The so called "*fresh starting*" problem would lead to loss of information between iterations and hence impair search efficiency [31]. Several NP variants with inheritance have been developed to address this problem [31], [32]. The downside of these methods is that the convergence cannot be guaranteed. Our method also adopts the inheritance idea, i.e., good samples of the previous iteration are inherited to the current iteration. Although the independent sampling requirement in the original convergence proof is violated, our method is still convergent by taking advantage of the global convergence property of genetic algorithm [33] which is used in each iteration of nested partitions to search in the entire feasible region.

[2]This backtracking rule will be used in all the NP methods in Section V.

d) Discussion: The hybrid nested partitions and genetic algorithm method (HNPGA) applies not only to the elevator scheduling problem, but also to those sequential decision problems which satisfy the two following characteristics. 1) The decisions in a short period only have a minor impact on the global performance, whereas the decisions in a reasonably long period will have a large impact on the global performance. 2) The earlier decisions have impact on the later, and the impact decreases as their time interval increases. It should also be noted that while genetic algorithm is an important module in the nested partitions framework, this framework is not restricted to genetic algorithm. Other algorithms which fit the characteristics of the problem to be solved can be integrated into this framework.

V. Numerical Results and Insights

The above method has been implemented in Matlab and tested on an AMD Atholon $3000 + 2$ GHz Windows PC with 1 GB RAM. Extensive numerical testing has been conducted. A building with ten floors is considered in the following three examples. The number of elevators varies from two to four and the length of the look-ahead time window varies from 30 s to 120 s.[3] To show the incremental contribution of our paper over [6], the planning horizon of all the examples are set to a single time window. Example 1 is to demonstrate the near-optimality of solutions, the value of future traffic information, and the value of the new door action control method. Example 2 compares our method with the traditional NP method and its several variants, and demonstrates values of the new NP features in our method. Example 3 compares our method with a pure standard genetic algorithm and demonstrates the value of the nested partitions framework.

Three traffic patterns are examined, including up-peak, down-peak, and interfloor. The generation of traffic data used in the following examples is based on the testing data in [6]. For all traffic patterns, arrival times are independent and uniformly distributed over the entire time window. For up-peak, arrival floors are set to the lobby, and destination floors are independent and uniformly distributed between the fourth and the ninth floors. For down-peak, the destination floors are set to 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 related parameters for elevators are taken from Elevate [34] in Table I. The objective function is the service time (i.e., $\alpha = \beta = 1$).

Example 1: This example is to demonstrate solution near-optimality under various traffic patterns and different traffic densities, the value of future traffic information, and the value of the new door action control method. The number of elevators in the building is two. Two data sets are tested: one is for a 120-s time

[3]The length of the look-ahead time window in the rolling horizon scheme [6] depends on information collection and/or prediction capability, service capacity, traffic load, and computational capability. Different lengths of time windows might be used under different problem settings and computational constraints. How to determine the window length, however, goes beyond the scope of this paper.

TABLE I
ELEVATOR CONFIGURATIONS

Speed (m/s)	2.5	Acceleration (m/s^2)	0.7
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
(DATA SET 1: 120 S, 1 PERSON/10 S)

Traffic Patterns		Up-Peak		Down-Peak		Inter-Floor	
Methods		BB	HNPGA	BB	HNPGA	BB	HNPGA
Avg. Service Times (s)	Mean	23.6	23.8	23.7	24.1	21.4	22.2
	STD	1.57	1.70	1.99	2.09	2.46	2.41
Avg. CPU Times (s)		247.9	3.6	648.4	2.2	4066.1	3.9

TABLE III
RESULTS FOR VARIOUS TRAFFIC PATTERNS (DATA SET 2: 60 S, 1 PERSON/5 S)

Traffic Patterns		Up-Peak		Down-Peak		Inter-Floor	
Methods		BB	HNPGA	BB	HNPGA	BB	HNPGA
Avg. Service Times (s)	Mean	29.3	31.6	30.9	33.7	31.5	34.5
	STD	2.06	2.28	2.46	2.68	3.98	4.68
Avg. CPU Times (s)		365.1	2.9	1299.0	2.0	13838.0	3.5

window with a constant arrival rate of one passenger per 10 s[4]; the other is for a 60-s time window with a constant arrival rate of one passenger per 5 s. Both data sets have the same number of passengers.

To demonstrate the near-optimality of solutions, our method (HNPGA) will be first compared with a trip-based branch and bound (BB) method which can generate optimal solutions [35]. The new door action control method is used in both methods. The parameters N and N_t for HNPGA are set to 3 and 6 based on our testing experience, respectively. The influence of varying N and N_t will be examined in Example 2. The population size N_p and the generation number N_g in the genetic algorithm module in HNPGA are set to 5 and 10, respectively. The crossover probability and the mutation probability of the genetic algorithm module for up-peak, down-peak and interfloor are set to 0.6 and 0.7, 0.7 and 0.5, and 0.6 and 0.9, respectively. The HNPGA method does not terminate until the most promising region becomes a singleton. The BB method does not terminate until it has found the optimal solution. Results from ten Monte Carlo simulation runs for the two data sets are summarized in Tables II and III.

For data set 1 (light traffic), with an average CPU time less than 4 s, the average gaps between the two methods for the three traffic patterns are 0.97%, 1.86%, and 3.74%, showing that near optimal solutions are obtained within rather short CPU times for various traffic patterns under light traffic. As the traffic pattern changes from up-peak to down-peak to interfloor, the gaps between the two methods increase because the scheduling problem becomes harder and harder, as demonstrated by the average CPU times for Branch and Bound to find the optimal solutions for the three cases. For both methods, the average service time for interfloor is the smallest among the three traffic patterns. The reason might be as follows. On the one hand, the average passenger travel distance for interfloor is the lowest among the three traffic patterns, as neither the origin floors nor the destination floors are restricted to the lobby. On the other hand, cars will have more stop floors under interfloor traffic, as the traffic demand for interfloor will have both multiple origin floors and multiple destination floors, causing delay in serving passengers yet to be delivered. For the light traffic, the delay caused by stop floors is minor because the interval of arrivals of passengers is larger, so that the benefit brought by shorter average distance dominates the downside caused by more stop floors.

For data set 2 (heavy traffic), with an average CPU time less than 4 s, the average gaps are 7.78%, 8.99%, and 9.53%, demonstrating near-optimality of solutions and computational efficiency for various traffic patterns under heavy traffic. Similar to the light traffic case, the gaps between the two methods increase as the traffic pattern changes from up-peak to down-peak to interfloor. As opposed to the light traffic, the average service time for interfloor is the largest, because the downside caused by more stops floors dominates the benefit brought by shorter average distance.

For the same traffic pattern, the average service time for data set 2 (heavy traffic) is larger than data set 1 (light traffic) because elevators will be busier in heavy traffic; the gap between the two methods for date set 2 is larger because the case under high arrival rate is harder to solve than the case with low arrival rate, as shown by the CPU times for Branch and Bound to obtain the optimal solutions.

To demonstrate the value of future traffic information and the value of the new door action control method, four algorithms will be compared, including the following.

1) HNPGA: the method presented in Section IV.
2) HNPGAD (HNPGA with dwell time): the same as HNPGA except enforcing that elevators cannot leave stop floors until the door dwell time expires.
3) Lagrangian Relaxation (LR): the method in [6] with the traditional door action control method.[5]
4) HNPGAT (HNPGA with three-passage heuristic): the same as HNPGA except that the single car dispatching strategy is replaced by a three-passage heuristic commonly adopted in certain real elevator systems [36], in which advance information is not utilized and the traditional door action method is adopted.

The second and fourth methods have the same settings for N, N_t, the genetic algorithm module, and the stopping criteria as those in the first one. Results from ten Monte Carlo simulation runs for the two data sets are summarized in Tables IV and V.

For data set 1, it can be seen that the methods using future traffic information (i.e., HNPGA, HNPGAD, and LR) outperform the one without using such information (i.e., HNPGAT). Our method HNPGA is 7.90% better than HNPGAD on average, showing the benefit of not restricting the elevators must leave the stop floors once the door dwell time expires in the new door action control method. Our method also outperformed the traditional action control based methods, 9.73% better than LR on average, and 22.79% better than HNPGAT on average. For data set 2, similar results can be obtained. The HNPGA is

[4]This data set is taken from [6] for later comparison with existing results.

[5]In [6], only the first data set is tested.

TABLE IV
RESULTS FROM DIFFERENT METHODS (DATA SET 1: 120 S, 1 PERSON/10 S)

	Methods		HNPGA	HNPGAD	LR	HNPGAT
Up-Peak	Avg. Service Times (s)	Mean	23.8	25.3	27.2*	28.1
		STD	1.70	1.89	/	1.64
	Avg. CPU Times (s)		3.6	3.1	842.6	1.3
Down-Peak	Avg. Service Times (s)	Mean	24.1	25.7	26.1*	31.9
		STD	2.09	2.01	/	1.94
	Avg. CPU Times (s)		2.2	2.3	751.9	1.7
Inter-Floor	Avg. Service Times (s)	Mean	22.2	25.1	24.4*	31.1
		STD	2.41	2.60	/	3.41
	Avg. CPU Times (s)		3.9	3.8	810.3	2.5

*the lower bound of the real avg. service time obtained by using LR

TABLE V
RESULTS FROM DIFFERENT METHODS (DATA SET 2: 60 S, 1 PERSON/5 S)

	Methods		HNPGA	HNPGAD	HNPGAT
Up-Peak	Avg. Service Times (s)	Mean	31.6	32.7	36.4
		STD	2.28	2.65	2.20
	Avg. CPU Times (s)		2.9	2.8	1.1
Down-Peak	Avg. Service Times (s)	Mean	33.7	35.0	40.0
		STD	2.68	2.66	3.98
	Avg. CPU Times (s)		2.0	1.9	1.4
Inter-Floor	Avg. Service Times (s)	Mean	34.5	36.6	42.0
		STD	4.68	5.01	4.94
	Avg. CPU Times (s)		3.5	3.5	2.2

TABLE VI
SCHEDULING DETAILS OF A SAMPLE RUN FOR UP-PEAK

Pass. ID	Arrival Floor	Arrival Time (s)	Dest. Floor	Elevator ID	Trip No.	Pickup Time (s)	Time to Close the Door (s)
1	0	2.15	6	2	1	0	4.15
2	0	11.26	4	1	1	0	23.59
3	0	21.59	4	1	1	0	23.59
4	0	31.20	7	2	2	32.11	35.61
5	0	42.11	3	1	2	49.05	54.55
6	0	51.25	3	1	2	49.05	54.55
7	0	60.58	8	2	3	65.57	71.85
8	0	69.85	6	2	3	65.57	71.85
9	0	79.39	4	1	3	77.99	81.49
10	0	89.82	6	1	4	105.45	110.95
11	0	100.70	4	1	4	105.45	110.95
12	0	111.11	8	2	4	113.17	116.67

4.27% better than HNPGAD on average and 15.60% better than HNPGAT on average. This example shows that one can gain significant benefit by using future traffic information and the new door action method.

To further illustrate the new door action method, the scheduling details from one of the Monte Carlo runs for up-peak is listed in Table VI. Elevator 1 stays at floor 0 at time 0 to serve passengers 2 and 3 in trip 1. After passenger 2 enters elevator 1 at time 13.26 (arrival time 11.26 plus loading time 2 s), elevator 1 will stay at floor 0 to wait passenger 3 to board even if door dwell time has already expired. Elevator 2 arrives at floor 0 at time 113.17 to serve passenger 12 in trip 4. Elevator 2 will immediately close the door after passenger 12 comes in rather than waiting until door dwell time expires.

Example 2: This example is to demonstrate the values of the new NP features of our method, including the partitioning scheme, the most promising region selection method, inheritance, and decision-making with local information. The number of elevators is four. The up-peak traffic is examined under two traffic densities with various look-ahead time windows. The arrival rate for the light traffic is one passenger per 5 s and the arrival rate for the heavy traffic is two passengers per 5 s.[6] The look-ahead time window varies from 30 to 120 s, with a constant step of 30 s. Our method (HNPGA) will be first compared with the traditional nested partitions method (TNP). The population size N_p and the generation number N_g in the genetic algorithm module in the two methods are set to 5 and 10, respectively. The crossover probability and the mutation probability of the genetic algorithm module are set to 0.6 and 0.7, respectively. Both methods do not terminate until the most promising region becomes a singleton. The detailed configuration of NP features for TNP and HNPGA are as follows.

1) TNP: $N = 1$, the traditional most promising region selection method (in which all the subregions are evaluated), no inheritance, $N_t = |S| - N \times k$ (the assignment of next passenger is decided through the optimization of all the unassigned passengers).
2) HNPGA: $N = 3$, the new most promising region selection method, with inheritance, $N_t = 6$ (the assignment of the next 3 passengers is decided through the optimization of the next six passengers).

Results from ten Monte Carlo simulation runs are summarized in Table VII.

TABLE VII
RESULTS FOR TNP AND HNPGA UNDER UP-PEAK TRAFFIC

Time Window	Traffic Density		Light		Heavy	
	Methods		TNP	HNPGA	TNP	HNPGA
30s	Avg. Service Times (s)	Mean	17.3	17.4	22.6	22.8
		STD	1.23	1.22	1.09	1.10
	Avg. CPU Times (s)		20.7	1.0	35.4	2.9
60s	Avg. Service Times (s)	Mean	19.8	19.7	29.5	28.1
		STD	0.71	0.62	0.94	0.75
	Avg. CPU Times (s)		37.2	3.3	339.1	12.7
90s	Avg. Service Times (s)	Mean	21.6	21.3	34.5	31.5
		STD	0.63	0.52	0.94	0.50
	Avg. CPU Times (s)		136.3	7.6	1283.0	32.2
120s	Avg. Service Times (s)	Mean	23.3	22.3	37.6	33.1
		STD	0.41	0.51	0.98	0.65
	Avg. CPU Times (s)		395.5	14.9	2697.6	63.8

It can be seen that compared with TNP, our method can generate better solutions, except slightly worse in the 30-s light and heavy traffic cases, within much shorter CPU times for various lengths of time windows and different traffic densities. The improvement of solution performance and the significant reduction of CPU times show the value of those new NP features. It should be noted that it is not meaningful to compare the results under different time windows for either method because the results are only for a snapshot problem instead of the overall problem, and the impact of the look-ahead time window length is out of the scope of this paper.

To further illustrate how individual features contribute to our method, three intermediate methods are constructed by adding new features one at a time, including the following.

1) TNP1: $N = 3$, the traditional most promising region selection method, no inheritance, $N_t = |S| - N \times k$.

[6]The settings of the arrival rates here are different from those in Example 1 due to the enlarged service capability (the increased number of elevators).

TABLE VIII
RESULTS FROM DIFFERENT METHODS FOR UP-PEAK

Methods		TNP	TNP1	TNP2	TNP3	HNPGA
Avg. Service Times (s)	Mean	19.8	19.7	20.2	19.8	19.7
	STD	0.71	0.55	0.66	0.73	0.62
Avg. CPU Times (s)		37.2	76.2	5.6	5.5	3.3
Maximum Depth		12	4	4	4	4
Avg. No. of Iterations		21.51	4.06	4.30	4.10	4.10
Avg. No. of Iterations / Maximum Depth		1.79	1.02	1.08	1.03	1.03

2) TNP2: $N = 3$, the new most promising region selection method, no inheritance, $N_t = |S| - N \times k$.
3) TNP3: $N = 3$, the new most promising region selection method, with inheritance, $N_t = |S| - N \times k$.

The three methods have the same parameter setting for the genetic algorithm module as TNP and HNPGA. The three methods together with TNP and HNPGA are tested on the up-peak traffic in a 60-s horizon, with a constant arrival rate of one passenger per 5 s. All the five methods do not terminate until the most promising region becomes a singleton. Results from ten Monte Carlo simulation runs are summarized in Table VIII.

The TNP method has the largest iterations depth ratio,[7] which indicates the frequent occurrence of backtracking, as depicted in Fig. 5, leading to low efficiency of NP and large amount of CPU time. By increasing N to be three in TNP1, the iterations depth ratio decreases to be close to 1, demonstrating the value of the new partitioning scheme. Nevertheless, the CPU time increases largely to 76.2 s. The reason is that as the number N increases, the number of subregions increases to 64 and the traditional most promising region selection method requires that all the subregions must be evaluated. By adopting the new most promising region selection method in TNP2, the CPU time decreases significantly to 5.6 s with slightly increase of the average service time, demonstrating the value of the new most promising region selection method. By introducing the inheritance feature in TNP3, the average service time slightly decreases from 20.2 to 19.8 s, showing the value of introducing this feature. By using local/partial information instead of whole information in HNPGA, the CPU time further decreases from 5.5 to 3.3 s. Of all the five methods, our method HNPGA has the smallest average service time and CPU time, and close-to-1 iterations depth ratio, demonstrating the values of the new NP features.

Example 3: This example is to compare our hybrid method HNPGA and the pure standard genetic algorithm in C.2.2.1 using the traffic data in Example 2. The parameter settings and stopping criteria for our method are the same as those in Example 2. The pure genetic algorithm (PGA) has the same population size as the genetic algorithm module used in our method. For each problem setting, PGA will spend the same amount time as HNPGA. Results from ten Monte Carlo simulation runs are summarized in Table IX.

Under light traffic, the relative performance differences of the two methods for the four time windows of lengths 30, 60, 90, and 120 s are, respectively, 0, 2.96%, 4.05%, and 5.51%. Under heavy traffic, the relative differences for the four windows of lengths 30, 60, 90, and 120 s are, respectively, 2.15%, 4.42%, 6.25%, and 8.06%. It can be seen that our hybrid method can improve the pure genetic algorithm by embedding it in the nested partitions framework, and the improvement increases as more information becomes available or the traffic becomes heavier. The results show the value of the nested partitions framework which concentrates the computational resources on promising regions that can be quickly identified by using local/partial future traffic information.

[7]The iterations depth ratio is defined as the ratio of the average number of iterations to the maximum depth of the searching tree.

TABLE IX
RESULTS FOR GA AND HNPGA UNDER UP-PEAK TRAFFIC

Time Window	Traffic Density		Light		Heavy	
	Methods		PGA	HNPGA	PGA	HNPGA
30s	Avg. Service Times (s)	Mean	17.4	17.4	23.3	22.8
		STD	1.18	1.22	0.97	1.10
	Avg. CPU Times (s)		1.0		2.9	
60s	Avg. Service Times (s)	Mean	20.3	19.7	29.4	28.1
		STD	0.76	0.62	0.9	0.75
	Avg. CPU Times (s)		3.3		12.7	
90s	Avg. Service Times (s)	Mean	22.2	21.3	33.6	31.5
		STD	0.62	0.52	0.86	0.50
	Avg. CPU Times (s)		7.6		32.2	
120s	Avg. Service Times (s)	Mean	23.6	22.3	36	33.1
		STD	0.70	0.51	0.90	0.65
	Avg. CPU Times (s)		14.9		63.8	

VI. CONCLUSION

One important trend to improve elevator systems is to use advance traffic information. It remains as an open and challenging issue to develop new scheduling methods that can effectively utilize advance traffic information. A two-level formulation is presented in this paper, with detailed car dynamics embedded in simulation models for performance evaluation. Taking advantage of advance information, a new door action control method is developed to increase the flexibility of elevators. In view of the hierarchy structure of this problem, a two-level optimization framework is developed, i.e., optimizing single car dispatching at the low-level with an effective trip-based heuristic, and optimizing passenger-to-car assignment at the high-level with a novel hybrid nested partitions and genetic algorithm method which can be extended to solve certain class of sequential decision problems. Numerical results demonstrate the near-optimality of solutions, computational efficiency, and values of advance information, the new door action control method, and new features of the hybrid method. Further improvement is needed to reduce CPU time for online implementation.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and Mr. B. Xiong of University of Connecticut and Prof. L. Shi of University of Wisconsin-Madison for their insightful comments and helpful advice.

REFERENCES

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

[2] J. Gale, "Destination control and tower top access in Belgium," *Elevator World*, vol. 10, pp. 45–49, 2002.
[3] T. Pfeifer, A. Micklei, and H. Hartenthaler, "Internet-integrated building control: Leaving the lab-robust, scalable and secure," in *Proc. 26th Annu. IEEE Conf. Local Comput. Netw.*, Nov. 2001, pp. 306–315.
[4] N. S. Shimbun, "Elevators get smart," *RFID in Japan* Dec. 2004.
[5] 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.
[6] P. B. Luh, B. Xiong, and S. C. Chang, "Group elevator scheduling with advance information for normal and emergency modes," *IEEE Trans. Autom. Sci. Eng.*, vol. 5, no. 2, pp. 245–258, Apr. 2008.
[7] M. L. Siikonen, "Procedure for Controlling an Elevator Group Where Virtual Passenger Traffic is Generated," U.S. patent 6,345,697, Feb. 12, 2002.
[8] F. Luo, Y. G. Xu, and J. Z. Cao, "Elevator traffic flow prediction with least squares support vector machines," in *Proc. 2005 Int. Conf. Mach. Learn. Cybern.*, Aug. 2005, vol. 7, pp. 4266–4270.
[9] M. Huang and Q. B. Zhu, "A novel predictive method for traffic flow of elevator system," *Dyn. Contin. Discret. Impul. Syst. Ser. A: Math. Anal.*, vol. 13, pp. 332–339, 2006.
[10] 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.
[11] G. Bao, C. G. Cassandras, T. E. Djaferis, A. D. Gandhi, and D. P. Looze, Elevators Dispatchers for Down-Peak Traffic ECE Dept., Univ. Massachusetts, Amherst, 1994, Tech. Rep..
[12] G. R. Strakosch*, Vertical Transportation: Elevators and Escalators*. New York: Wiley, 1998.
[13] G. C. Barney*, Elevator Traffic Handbook: Theory and Practice*. New York: Spon, 2003.
[14] A. T. So, J. K. Yu, and W. L. Chan, "Dynamic zoning based supervisory control for elevators," in *Proc. 1999 IEEE Int. Conf. Control Applicat.*, Aug. 1999, pp. 1591–1596.
[15] R. Crites and A. Barto, "Elevator group control using multiple reinforcement learning agents," *Mach. Learn.*, vol. 33, pp. 235–262, 1998.
[16] T. Tobita, A. Fujino, H. Inaba, K. Yoneda, and T. Ueshima, "An elevator characterized group supervisory control system," in *Proc. IEEE Int. Conf. on Ind. Electron.*, 1991, pp. 1972–1976.
[17] D. Nikovski and M. Brand, "Decision-theoretic group elevator scheduling," in *Proc. 13th Int. Conf. Autom. Plan. Sched.*, Trento, Italy, 2003, pp. 133–142.
[18] J. Lewis, "A Dynamic Load Balancing Approach to the Control of Multiserver Polling Systems With Applications to Elevator System Dispatching," Ph.D. dissertation, ECE Dept., Univ. Massachusetts, Amherst, 1991.
[19] M. L. Siikonen, "Elevator traffic simulation," *Simulation*, vol. 61, no. 4, pp. 257–267, 1993.
[20] P. Cortes, J. Larraneta, and L. Onieva, "A genetic algorithm for controlling elevator group systems," in *Artificial Neural Net, Problem Solving Methods*. Berlin, Germany: Springer-Verlag, 2003, pp. 313–320.
[21] H. Ujihara and S. Tsuji, "The revolutionary AI-2100 elevator-group control system and the new intelligent option series," *Mitsubishi Electr. Adv.*, vol. 45, pp. 5–8, 1988.
[22] N. Imasaki, J. Kiji, and T. Endo, "A fuzzy neural network and its application to elevator group control," in *Fuzzy Engineering Toward Human Friendly Systems*. Amsterdam, The Netherlands: IOS, 1992.
[23] S. Markon, H. Kita, and Y. Nishikawa, "Adaptive optimal elevator group control by use of neural networks," *Trans. Inst. Syst. Contr. Inform. Eng.*, vol. 7, pp. 487–497, 1994.
[24] R. Smith and R. Peters, "ETD algorithm with destination dispatch and booster options," *Elevator World*, vol. 6, 1996.
[25] J. F. Bard*, Practical Bilevel Optimization: Algorithms and Applications*. Dordrecht, The Netherlands: Kluwer, 1998.
[26] S. Dempe*, Foundations of Bilevel Programming*. Dordrecht, The Netherlands: Kluwer, 2002.
[27] Q. C. Zhao, J. Sun, and Z. Shen, Elevator Dispatching Performance Analysis 2005, United Tech. Res. Center Internal Report, unpublished.
[28] L. Y. Shi and S. Olafsson, "Nested partitions method for global optimization," *Oper. Res.*, vol. 48, no. 3, pp. 390–407, 2000.
[29] L. Y. Shi, S. Olafsson, and Q. Chen, "A new hybrid optimization algorithm," *Comput. Ind. Eng.*, vol. 36, no. 2, pp. 409–426, Apr. 1999.
[30] D. E. Goldberg*, Genetic Algorithm in Search, Optimization and Machine Learning*. Boston, MA: Kluwer, 1989.
[31] S. Al-Shibabi, "Ants for sampling in the nested partitions algorithm," in *Proc. 16th Eur. Conf. Artif. Intell.*, Valencia, Spain, 2004, pp. 11–18.
[32] J. Kim and S. Olafsson, "Two-stage NP method with inheritance," in *Proc. 2002 Winter Simulation Conf.*, 2002, pp. 279–284.
[33] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Networks*, vol. 5, pp. 96–101, Jan. 1994.
[34] *Elevate*, , 2002 [Online]. Available: www.elevate.peters-research.com, Peters Research, Ltd.
[35] Z. Shen and Q. C. Zhao, "A branch and bound method to the continuous time model elevator system with full information," in *Proc. SICE Annu. Conf.*, 2007, pp. 327–330.
[36] 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.

Jin Sun (S'05) received the B.S. degree in automation from Tsinghua University, Beijing, China, in 2004. He is currently working towards the Ph.D. degree in control science and engineering at the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University.

He is a visiting scholar at the University of Connecticut, Storrs. His main research interests include optimization theory and algorithms, scheduling and control, and discrete event dynamic systems.

Qian-Chuan Zhao (M'06–SM'08) received the B.E. degree in automatic control in July 1992, the B.S. degree in applied mathematics in July 1992, and the Ph.D. degree in control theory and its applications in July 1996, all from Tsinghua University, Beijing, China.

He is currently a Professor and Associate Director of the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Tsinghua University. He was a Visiting Scholar at Carnegie Mellon University, Pittsburgh, PA, and Harvard University, Cambridge, MA, in 2000 and 2002, respectively. He was a Visiting Professor at Cornell University, Ithaca, NY, in 2006. He is an Associate Editor for the *Journal of Optimization Theory and Applications*. His research interests include discrete event dynamic systems (DEDS) theory and applications, optimization of complex systems.

Peter B. Luh (S'77–M'80–SM'91–F'95) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, in 1973, the M.S. degree in aeronautics and astronautics engineering from Massachusetts Institute of Technology (M.I.T.), Cambridge, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, in 1980.

Since then, he has been with the University of Connecticut, and currently is the SNET Professor of Communications and Information Technologies and the Head of the Department of Electrical and Computer Engineering. He is also a member of the Chair Professors Group at the Center for Intelligent and Networked Systems, the Department of Automation, Tsinghua University, Beijing, China. He is interested in planning, scheduling, and coordination of design, manufacturing, and supply chain activities; configuration and operation of elevators and HVAC systems for normal and emergency conditions; schedule, auction, portfolio optimization, and load/price forecasting for power systems; and decision-making under uncertain or distributed environments.

Dr. Luh is Vice President of Publication Activities for the IEEE Robotics and Automation Society, an Associate Editor of *IIE Transactions on Design and Manufacturing*, an Associate Editor of *Discrete Event Dynamic Systems*, and was the founding Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING (2003–2007) and the Editor-in-Chief of IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION (1999–2003).