Available online at www.sciencedirect.com
ScienceDirect
Control Engineering Practice 15 (2007) 219–230
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

Prioritised A* search in real-time elevator dispatching

Muna Hamdi[a], D.J. Mulvaney[b,*]

[a]School of Computer Science, Cardiff University, Cardiff CF24 3AA, UK
[b]Department of Electronic and Electrical Engineering, Loughborough University, LE11 3TU, UK

Received 13 August 2004; accepted 21 June 2006
Available online 7 September 2006

Abstract

Under the typical operating conditions of an elevator system, there is insufficient time to consider all dispatching alternatives and the major elevator companies normally adopt empirical techniques to reduce complexity and achieve acceptable performance. The current work has been able to demonstrate that in practical circumstances an optimal solution to the real-time elevator-dispatching problem can be obtained. The elevator dispatching problem is formulated as a heuristic search and is implemented using a novel extension of the popular A* search, termed prioritised A*, that retains the desirable admissibility and monotonicity of A*. PA* makes best use of the limited time available by ensuring the dispatcher considers the most important aspect of the problem first, namely to give each elevator its first assignment. In a manufacturing process, this is equivalent to ensuring that each machine is immediately given its first job, while the determination of the detailed order of the remaining jobs is refined later. This study has obtained access to extensive data records collected from installed elevator systems and their analysis has led to the identification of new passenger models able to deliver suitable high quality predictive data to improve the operations of the dispatcher.

Keywords: Scheduling algorithms; Heuristic searches; Real-time AI; Path planning

1. Introduction

Modern high-rise buildings are normally serviced by collectively controlled groups of elevators. The elevator traffic patterns and their intensities depend greatly both on the building function and on the time of day, as in the example for an office building shown in Fig. 1.

In an office building, employees arriving for work in the morning will generally produce a high demand at the terminal floor resulting in *up peak* traffic. For most of the remainder of the morning and during the afternoon there is relatively uniform movement between all floors, termed *interfloor* traffic. During this time it is possible for elevator traffic patterns to change significantly and rapidly (arising, for example, as a result of conference meetings or fire drills), leading to unpredictable fluctuations in demand. Over the lunch period, there is greater demand to travel to terminal and restaurant floors and subsequently there may be another period of up peak traffic as staff return to office floors. At the end of the day, staff take the elevator to the terminal floor forming the *down peak* traffic. Elevator dispatchers normally attempt to detect the onset of up peak and down peak traffic conditions and modify their control behaviour accordingly. Note that the data gathered to produce Fig. 1 was obtained from a dispatcher that did not implement down peak, so such a scheduling state is not shown.

The purpose of an elevator dispatcher is to determine the schedule of elevator movements that best meets a specified goal. The most commonly adopted goal is to minimise a function involving the time passengers wait for elevators to arrive, but may also involve other factors such as reducing energy usage. Improving the performance of elevator dispatchers is important for two main reasons. Firstly, in the design of new buildings, it may be possible to reduce the number of elevators required to service the building and so increase the useful floor space available for occupants. Secondly, in an existing building, journey time for the current occupants can be reduced. Another desirable feature of a dispatcher is that it should be able

*Corresponding author. Tel.: +44 1509 227042; fax: +44 1509 227014.
E-mail address: d.j.mulvaney@lboro.ac.uk (D.J. Mulvaney).

0967-0661/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2006.06.005

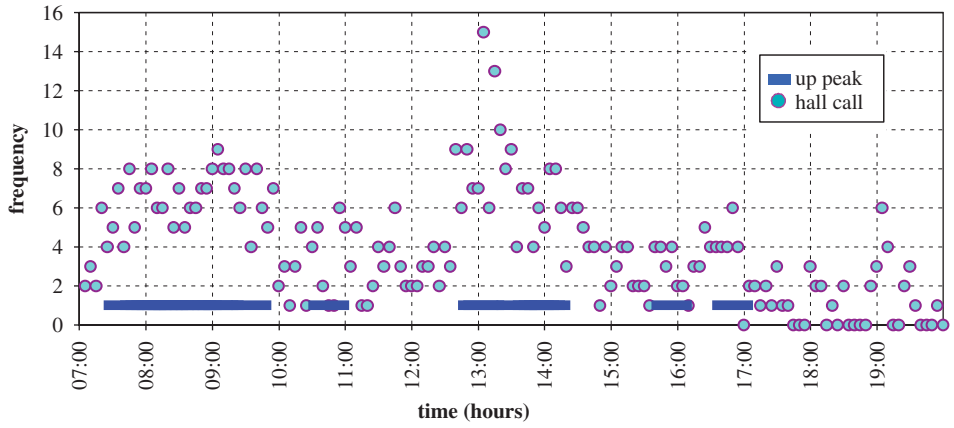

Fig. 1. Hall calls calculated every 5 minutes at the terminal floor between 7 am and 8 pm in a 18 floor London office building with a population of approximately 900 workers.

to adapt automatically both to short-term changes, such as the onset of up peak traffic, and to long-term changes, such as alterations to the nature of building occupancy.

The future state of an elevator system is influenced by new hall calls (produced by passengers requesting elevators), car calls (produced by passengers in elevators selecting their destinations), elevator movements and the schedule produced in the current state. Although the incremental updating of a schedule may be achieved by performing repair and backtracking operations, such an approach is rarely adopted as the resulting implementation greatly increases complexity, adversely affecting maintainability. Consequently, the approach normally taken is to perform the search for a solution from afresh in each scheduling cycle.

The quality of the decisions that can be taken by a dispatcher depends critically on the quality of the data it receives (Mulvaney & Sillitoe, 1992). A dispatcher has direct access to frequently updated data regarding the locations of elevators, hall calls and car calls, but data relating to the positions and movement of individual passengers are not available explicitly. One of the main achievements of the current work that has already been reported (Hamdi & Mulvaney, 1998), has been to use those data directly available to provide estimates of passenger data and this has been shown to improve significantly the elevator dispatcher performance. Estimates of when passengers are likely to arrive and their probable destinations can be used not only to improve the performance of installed elevator systems, but also to enhance prediction software for use by designers of proposed installations. Further refinements to data quality (Hamdi & Mulvaney, 1996) include statistical models of elevator events (such as estimating the time taken for a given numbers of passengers to enter or exit an elevator) and models of the dynamic movements of elevators (that allow accurate predictions of how long an elevator will take to deal with calls it is already committed to answer before becoming free to respond to new hall calls).

Elevator dispatching falls into the category of NP-complete problems (Garey & Johnson, 1979), namely those whose solution time increases exponentially with complexity. In a building with n elevators and p hall calls, there are n^p possible scheduling solutions. For a group of six elevators, if there are four hall calls to be served, over 7000 alternative solutions are available; for 10 hall calls this increases to over 60 million possible solutions. Moreover, to provide an adequate service to passengers, the commands given to the elevators that specify their next operations need to be recalculated in a time interval typically no greater than 200 ms (Hamdi, 2001). The selection of the time interval is a delicate balance: its value needs to be both sufficiently large that there is time for the dispatcher to generate a high quality schedule, yet sufficiently small to allow the dispatcher to be responsive to passenger requests. Considering such requirements, it clearly cannot normally be guaranteed that an optimal solution can always be found within the critical time required to produce a schedule.

One common way of representing such problems is to phrase them as a search task. Normally, the deeper the search that can be performed, the better the quality of the schedule produced, but the longer is the computation time. In real-time problems, an approach needs to be found that will reach a suitable solution in the time available. Due to commercial constraints, the detailed internal workings of installed elevator dispatchers are not published in the literature (Crites & Barto, 1998); but the general principles of the approaches adopted are available. A more detailed discussion of the principal drawbacks of these approaches is provided in the following section, but in brief, these are that they may not always make full use of the time available, or that they cannot guarantee to find an adequate solution in the time available. To overcome these drawbacks, an alternative solution, termed prioritised A* (PA*), is proposed in which the solutions to the most important parts of the problem are tackled first and the solution is continually refined until the time limit is

reached. In the practical application of PA* to installed elevator systems, it is demonstrated that it is normally possible to achieve an optimal solution in the time interval available. In the remainder of the paper, candidate solutions for real-time search are discussed, PA* itself is described, and its operation is first introduced for a simplified elevator system before being applied to a practical elevator installation.

2. Real-time search and its application to scheduling

Search techniques are able to find suitable paths that map an initial state to a goal state, while heuristics can be applied to limit the extent of the search and so control its execution time (Hamdi & Mulvaney, 1996). A* and its variants are probably the most popular best-first searches (Hart, Nilsson, & Raphael, 1968). The *best path* is determined from the values returned by a heuristic state evaluation function, f, that is the sum of the cost of reaching the current point in the search tree, g, and the estimated cost of attaining the final goal from the current point, h. If h is never overestimated, the solution will be optimal in terms of g (Korf, 1988). Although, in practice, significant numbers of undesirable solutions are pruned, the number of nodes in an A* elevator scheduling search tree will grow exponentially with the number of hall calls and so the determination of the optimum schedule is likely to take longer than the time interval available. To guarantee deadline compliance, there is normally a trade-off between the complexity of the search that is undertaken and the quality of the resulting schedule (Hamidzadeh & Atif, 1996). To solve real-time problems, enhancements to basic search approaches have been proposed by a number of authors. These enhancements normally fall in two main categories discussed below, namely limiting the search horizon and allowing the search to be interrupted at the end of the time interval; but a small number of other approaches that are also relevant to the current work are discussed.

2.1. Limiting the search horizon

One approach to ensuring real-time operation is to restrict the search horizon to ensure that a known, limited number of alternatives will have been investigated in the time available. The main drawback of this type of approach is that not all of time interval available will be used to perform the search, as, to ensure the search completes, the selection of heuristics needs to take into account a margin for error in the calculation time.

A form of A* search was used by Clark, Mehta, and Prowse (1994) in an elevator dispatcher which used a computer vision system to estimate the number of passengers waiting for an elevator to arrive. The cost evaluation function was chosen to reflect the overall travel time of passengers and the search was directed towards the highest calculated cost. The best move was decided using rules to calculate the actual and estimated cost and a look-ahead search. The authors found that it is not possible to calculate the costs of the large number of alternative search paths in the time available and recommended that heuristics be devised to prune the search tree.

Another approach to limiting the search horizon is provided by an adaptation of A* termed learning real-time A* (LRTA*) described by Korf (1990). Since decisions are often based on limited information, an initially promising direction may appear less favourable once it is explored, thus motivating a return to the previous choice point. The horizon is limited by how far the search can progress at any given time, but LRTA* is able to make progress towards the goal state without having to plan a complete sequence of solution steps in advance. The key difference from A* is that LRTA* assesses the merit of every node relative to the current scheduler state; consequently the initial state is irrelevant and the g value represents the cost from the current state to the next state rather than from the initial state. The LRTA* family of algorithms is probably most useful when a single goal is defined (for example a robot navigating to a target location (Shah-Hamzei & Mulvaney, 1999), and where a policy of re-evaluation after committing only to small steps may be more advantageous than continuing to greater depths. However, in elevator systems many decisions cannot be undone, for example occupied elevators are constrained not to change direction, and so their scheduling is probably better addressed by greater consideration before execution than by a policy of least commitment. Also, in the current work, a robust and reliable real-world solution was required and on-line learning (such as that available in LRTA* to improve performance on repeated presentation of the same problem) was rejected in favour of a statistical approach to the modelling of passenger movements.

2.2. Allowing the search to be interrupted

This technique allows the search to operate in its normal way, but continually keeps the best solution encountered so far. This means that no matter at what point the search is interrupted, the best solution encountered so far is made available. An early adoption of this approach was in the elevator dispatching system described by Schindler in Barney and dos Santos (1985), where each branch in a search tree corresponded to an assignment of an elevator to a hall call. Following the action of a collection of rules drawn from previous experience, the search provided an initial solution that was successively refined using alpha pruning for a period of time determined by the configuration of elevators and hall calls. The solution was then moderated by an analysis of the configuration resulting from a simulation of the future states, for example the solution would be rejected if the elevators become closely grouped. More recently, Mouaddib and Gallone (1996) used a progressive scheduling technique to generate rapidly an initial estimate for subsequent successive refinement.

The main drawback of this approach for the current work is that it cannot be guaranteed that, in the time available, sufficient nodes of the tree will have been examined to produce the first destinations for each of the elevators.

2.3. Other approaches

Another search-related approach applicable to elevator dispatching is iterative deepening A* (IDA*) described in Korf (1998). A principal advantage of IDA* is that only a comparative operation is needed when determining the next state, reducing the computational overhead compared with A* which needs to perform sorting, insertion and deletion operations (Korf, 1988). However, one critical drawback of IDA* in its application to real-time search is that if the calculation of the solution takes longer than the allowed scheduling period and the scheduler then needs to be interrupted, A* is able to give a partial schedule while IDA* is not guaranteed to do so.

Elevator dispatching has been investigated using a range of methods other than search. Crites and Barto (1996) developed a Markov-based solution that used a multi-agent approach (one per elevator) and considered two alternative implementations, namely where the agents shared a network and where they operated in their own separate networks. Kim, Seong, Lee-Kwang, and Kim (1998) selected a suitable mode of elevator operator by applying fuzzy rules whose inputs were obtained from parameters in the elevator environment (such as the number of passengers wishing to travel up or down the building), Gudwin, Gomide, and Netto (1998) recorded linguistic fuzzy variables, such as which hall call is waiting for service and the time each elevator expects to attend outstanding hall calls, and then, by adopting suitable membership functions, applied fuzzy rules to determine the dispatching schedule. Further improvements were demonstrated by context adaptation, whereby the fuzzy membership functions were altered according to the immediate traffic density. Beielstein, Ewald, and Markon (2003) employed a genetic algorithm (GA) to optimise the performance of a dispatcher operating under the control of a neural network The main drawback of these approaches is that, at the level of abstraction adopted, it is not clear how the simplifying assumptions will affect performance. Sorsa, Siikonen, and Ehtamo (2003) considered the optimal routing of double-deck elevators, by formulating the dispatching as an integer programming problem and solving using a GA. Pepyne and Cassandras (1997) represented up-peak dispatching as a batch service queuing system and used dynamic programming to show that minimizing the waiting time can be optimised by a threshold policy. However, in the elevator environment such thresholds are in constant flux, requiring frequent re-application of the dynamic programming equations. So and Chan (1997) investigated the application of GAs to determine dynamically the extent of zones (a set of adjacent floors serviced by a group of elevators). Cortes, Larrañeta, and Onieva (2003) showed that, in comparison with the THV dispatcher developed at UMIST (Barney & dos Santos, 1985), the waiting times during lunch time traffic could be reduced by the application of a GA with fixed-length chromosomes.

A rigorous, fundamental approach has been taken in the current work, namely to develop a suitable A* model able to represent accurately the underlying nature of the problem by including timings of elevator movements and statistical models of both passenger and elevator events. Combined with appropriately formulated heuristics, the results show that this approach is able to solve the elevator dispatching problem in an existing building.

3. PA* and its application to elevator dispatching

To overcome the drawbacks of the existing approaches discussed above, an alternative method was sought which is based on a sound understanding of the elevator scheduling problem, is able to use the full time interval available for calculations and, on interruption at the end of this time, is able to deliver a schedule providing the first destinations of the elevators. As the immediate goal is to give each elevator its first assignment, it is important to be able to find a search tree structure that provides each elevator with a hall call assignment without the need to search exhaustively.

Prioritised A* has been so named by the authors as the search tree is reorganised such that a useable solution can be obtained from a minimal depth of search. In general, if there are p tasks to be allocated to a set of n resources, PA* gives priority to making sure that the resources are each given their first assignment. In a manufacturing process, this would be to ensure that each machine is immediately given its first job; in elevator dispatching, each elevator is given its first destination. Such reorganising of the search involves ensuring that the first n levels in the tree correspond to the resources that need to be allocated tasks. This sets a minimum performance requirement for the search, namely that it must be able to expand the tree at least as far as the nth level (equal to the number of elevators) in the allotted time. As long as this can be achieved, the search can be halted at any future time, knowing that the best solution encountered so far is available. The algorithm for PA* is shown in Fig. 2. It is important to note that the reorganisation of the search in this way does not affect the properties of admissibility and monotonicity, and PA* inherits both of these features from A* (Korf, 1990). One advantage of PA* with respect to A* is that, because the open set contain nodes only from one level of the expanded tree, only nodes in that level of the tree need be sorted in each generation, thereby avoiding the need to sort the whole set of open nodes as is required by A*.

When the number of hall calls is relatively large, the time to compute the complete search is likely to be longer than the calculation time available. To reach the n^{th} level with the minimum of backtracking, heuristics need to be

Initialize

(1) Let L (where $L \geq 1$) be the number of 'levels' (sometimes termed ply) in the tree, not including the level with the root node.

(2) Create a 'closed set' of nodes that have been searched $C=\{\}$, and the $M_i=\{\}$, $i=1,\ldots,L$ are the sets of 'open nodes' (those yet to be searched) for each level i in the tree. M_0 contains the root node. Before entering the generate phase, set $i=0$.

Generate

(1) Generate a set of nodes n_{i+1} that are the descendants of the node m_i that has the lowest cost in M_i. For the elevator system, nodes are generated for each hall call not already assigned at any of the levels 1 to i and also for the case of no hall call assignment at level $i+1$.

(2) Add the nodes n_{i+1} to M_{i+1}.

(3) Sort the nodes M_{i+1} in increasing order of their $\hat{f}$ values. If there is a tie for minimum $\hat{f}$, the relevant nodes are sorted according to their $\hat{g}$ values.

(4) Move the node m_i to C.

Test

If the first node in M_{i+1} is the goal state, then finish. Otherwise choose i according to the M_i containing the mode of lowest cost and return to the **generate** stage.

Fig. 2. PA* search algorithm. It is assumed that the search forms a tree with no duplicate nodes. If the number of hall calls is less than the number of elevators, then the algorithm terminates once all hall calls are assigned, otherwise the search may either terminate when all elevators have their first assignment or when all hall calls have been assigned following successive applications of the above algorithm.

applied. As is generally the case in search problems, its satisfactory solution depends critically on selecting suitable heuristics that give an accurate indication of the most suitable path, yet are computationally inexpensive. Such accuracy is important in PA*, as, if the search needs to be interrupted at the end of the available period, the scheduler is more likely to have either the optimal assignment or one close to the optimal. Employing computationally expensive heuristics implies that more time will be dedicated to investigating the relative merits of following proposed branches at the expense of expanding fewer nodes.

In elevator dispatching, the most common goal is to minimise the mean time for hall calls to be answered, termed *waiting time*. Suitable heuristics are required to balance the different factors affecting the assignment of hall calls to achieve the minimum waiting time; this is the credit assignment problem (Korf, 1990). For example, independently choosing the shortest path to one hall call can increase the time taken to answer the remaining hall calls; it is the schedule having the smallest combined cost for answering all hall calls that needs to be found. An alternative, but far less popular parameter that could be considered when producing an objective assessment of the performance for an elevator system is journey time (Barney, 2003), namely the sum of the waiting time and the time spent travelling in an elevator car. However, as the time spent awaiting the arrival of an elevator is psychologically more apparent to passengers than that the travel time while in the car itself, waiting time is often considered by elevator engineers as more important measure. Consequently, we chose to use the waiting time rather then the journey time in all our implementations. Further, the installed dispatcher we used for comparison purposes to benchmark our results also used waiting times, and consequently our own use of waiting times makes possible a fairer appraisal. The heuristic functions the authors considered most suitable for the elevator-dispatching problem are described in the next section.

4. Worked calculation of a simplified elevator schedule using PA*

To help in the understanding of the operation of PA*, this section describes its application to a simplified elevator system. The next section considers the application of PA* search under realistic operating conditions.

To make clear the differences between PA* and existing search techniques, the search tree is first computed using A*. Fig. 3 shows an example of a building with two elevators already moving to answer car calls. The search tree shown is that generated by the application of a standard A* search. At each level, one node is generated for each elevator and hall call combination. The hall call is assigned to that elevator having the smallest cost of reaching of reaching the call, f. A method is needed to obtain the cost estimates required in the search tree. In this work, as the interfloor travel times are known because of the fixed construction of a building, these can be kept in tables that would only need to be modified should the performance of the installed elevator system be altered, perhaps following an upgrade. Other cost values are computed in look-up tables immediately prior to performing a search. The first of the tables holds the costs, taking into account any current commitments, of trips from an elevator's current position to answer its first hall call. Secondly, a set of tables, one for each elevator, is used to hold the costs of trips between pairs of hall calls while taking into account any car call commitments. The tables for the example elevator system shown in Fig. 3 are given in Tables 1 and 2. In Table 1, the values (in arbitrary units) are those for answering the first hall call, where, for example, for elevator 0 to answer hall call 0 (HC0) first, it must first deal with its current car call by moving to the relevant floor (3 units of time) and answering the call (4 units), before moving down to the ground floor (7 units). As an example of the calculation of the entries in Table 2, consider the case where, somewhere in its schedule, elevator 1 needs to answer HC1 immediately before HC0. On answering HC1 (4 units of time) at floor 3, then, assuming a passenger enters, elevator 1 will have a car call to answer. In the worst case this will be the top floor (as HC1 is an upwards call), which takes 5 units of time for travel and 4 to answer. Finally, to move all the way to the bottom floor requires an additional 8 units of time.

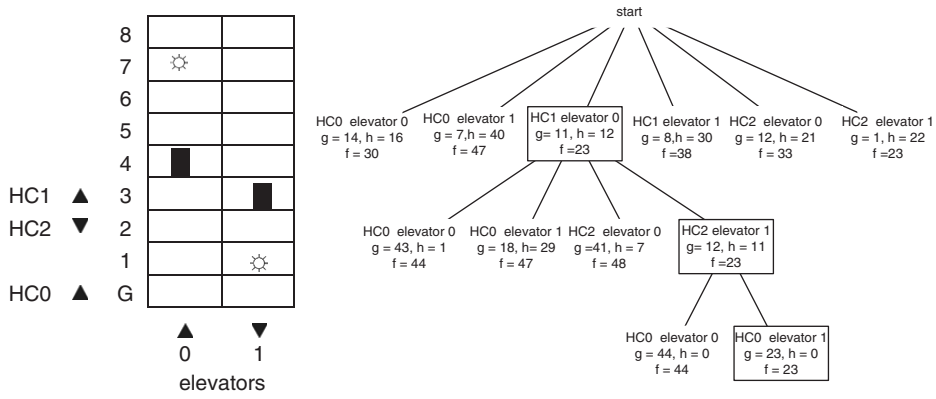

Fig. 3. A simplified A* search example, where, once at the appropriate floor, answering either a hall call (HC0, HC1 or HC2) or a car call (☼) takes four units of time and the transit time between adjacent floors takes a single unit of time. The elevator positions are indicated by ▮, while the arrow symbols (▲ and ▼) indicate the directions of elevator movements and hall calls.

Table 1
Costs of trips from current elevator positions to hall calls for the example of Fig. 3

First hall call	Elevator 0	Elevator 1
HC0	14	7
HC1	11	8
HC2	12	1

Table 2
Costs of trips between pairs of hall calls for the example of Fig. 3

	Previous hall call		
	HC0	HC1	HC2
Elevator 0 next hall call			
HC0		21	6
HC1	7		13
HC2	22	19	
Elevator 1 next hall call			
HC0		21	10
HC1	7		17
HC2	22	19	

The values of g and h for each node in the search tree of Fig. 3 are obtained as follows. At a given level in the search tree, if P_n is the number of hall calls p_i that are assigned to elevator n, then, for each hall call m assigned to the elevator, the cost $g_n(m)$ of answering the hall call currently under consideration is given by

$$g_n(1) = d_n(p_1), \qquad P_n = 1,$$
$$g_n(m) = d_n(p_1) + \sum_{i=1}^{P_n-1} s_n(p_i, p_{i+1}), \qquad P_n > 1, \quad m = 1, ..., P_n, \tag{1}$$

where d_n are the costs to answer the first hall call and s_n are the costs of the trips between hall calls in the order assigned. The value of g at the node is then the sum of the g_n. For the example in Fig. 3, the values of d_n and s_n are shown in the bodies of Tables 1 and 2, respectively. At a given level in the search tree, the candidate values h_n for each elevator to answer one of the Q unassigned hall calls q_j are found using

$$h_n(k) = \min[d_n(q_j)], \qquad P_n = 0, \quad j = 1, \ldots Q,$$
$$h_n(k) = \min\left[\sum_{m=1}^{P_n} g_n(m) + s_n(p_{P_n}, q_j)\right], \qquad P_n > 0, \quad j = 1, \ldots Q, \tag{2}$$

where k is the number of unassigned hall calls temporarily assigned at the current node. The smallest value from the $h_n(k)$ is selected as the value of $h(k)$ and the corresponding q_j is assigned to elevator n for the remainder of the calculations for this node, modifying temporarily its values of P_n and g_n. Eq. (2) is then applied repeatedly until all unassigned hall calls have been used in the calculations. The value of h is then the sum of the $h(k)$.

Using the search tree shown in Fig. 3, an example calculation of the values of g and h is now given. Consider the highlighted (boxed) node immediately below the root node. Here, the assignment of elevator 0 to HC1 is being proposed. The cost of reaching HC1 from the current position of elevator 0 is the value of g, here given by

$$g_0(1) = d_0(1) = 11, \tag{3}$$

where the value of $d_0(1)$ is that for elevator 0 to answer HC1 as found in Table 1. To compute the h values to represent the cost remaining, it is necessary to consider the assignments of HC0 and HC2 to the elevators. The costs of the two possible next assignments of elevator 0 are obtained from Eq. (2)

$$h_0(1) = \sum_{m=1}^{1} g_0(1) + s_0(1, 0) = 11 + 21 = 32$$

or

$$h_0(1) = \sum_{m=1}^{1} g_0(1) + s_0(1, 2) = 11 + 19 = 30, \tag{4}$$

where the values of $s_0(1,0)$ and $s_0(1,2)$ can be found from Table 2. Similarly, the costs of the two possible next assignments of elevator 1 are given by

$$h_1(1) = d_1(0) = 7$$

or

$$h_1(1) = d_1(2) = 1. \quad (5)$$

The smallest value of $h(1)$ is 1, namely when HC2 is assigned to elevator 1. Finally, the assignment of HC0 still needs to be determined and the values of $h(2)$ are found for each of the elevators.

$$h_0(2) = \sum_{m=1}^{1} g_0(1) + s_0(1,0) = 11 + 21 = 32$$

or

$$h_1(2) = \sum_{m=1}^{1} g_1(2) + s_1(2,0) = 1 + 10 = 11. \quad (6)$$

The smallest value of $h(2)$ is 11, when HC0 is assigned to elevator 1. The value of h is then the sum of the smallest values of both $h(1)$ and $h(2)$, this being the total waiting time to answer HC0 and HC2 (assuming elevator 0 answers HC1). It is vitally important to note that such assignments during tree exploration are for calculation purposes only and are by no means guaranteed to be the optimal; however, as the search tree is explored deeper, the estimates are continually refined and the corresponding assignment substantiated.

As the number of hall calls increases, there is an approximately exponential increase in the number of nodes in the tree and the longer it will take to calculate the value of f. When the number of hall calls exceeds the number of elevators, the immediate goal of the scheduler should be to find the first assignment for each elevator, although, using the method described earlier, the estimates of the costs of answering the remaining hall calls are taken into account in the calculation of h. The basis of the PA* approach is to ensure the first assignments are available as soon as possible in the search. This is achieved by prioritising the search tree according to elevator usage rather than being concerned that all hall calls are assigned to elevators. In the PA* search, each level of the tree involves exactly one of the elevators and considers its possible assignment to all the hall calls not already assigned at higher levels of the tree. Each level also needs to include the possibility of no assignment to that elevator, as it is perfectly possible that elevators to be considered at lower levels in the tree are better placed to answer the outstanding calls. This also means that even where the number of hall calls is fewer or equal to the number of cars, the PA* dispatcher is at liberty to assign more than one call (or even all the calls) to any given car. If no assignment is chosen, then, as long as the search remains within that branch of the tree, the elevator will remain free of further assignment until the end of search. If only one elevator is available for assignment then no free node is generated for that elevator; for example, only one node is generated for elevator 1 at the final level of the tree in the PA* example in Fig. 4. The node calculations are the same as those for the A* search, except that only the elevator being considered at that level of the tree is used in the application of Eq. (1). One advantage of PA* is that the order of implementation of the schedule is clear from the search tree and is that found when following the set of expanded nodes from the start of the search. For example, in the search tree in Fig. 4, the assignments to elevator 1 are simply that it should answer hall call 2 followed by hall call 0. The time the passenger who issued hall call 0 needs to wait for the arrival of an elevator is simply the sum of the times taken for elevator 1 to answer first hall call 2 and then hall call 0.

As the tree structure forces the assignment of elevators to hall calls rather than vice versa, it is possible to stop the

first hall call

lift	call	g
0	HC1	11

second hall call

lift	call	s_n	$h_n(1)$
0	HC0	21	
0	HC2	19	
1	HC0	7	
1	HC2	1	1

third hall call

lift	call	s_n	$h_n(2)$
0	HC0	21	
1	HC0	10	11

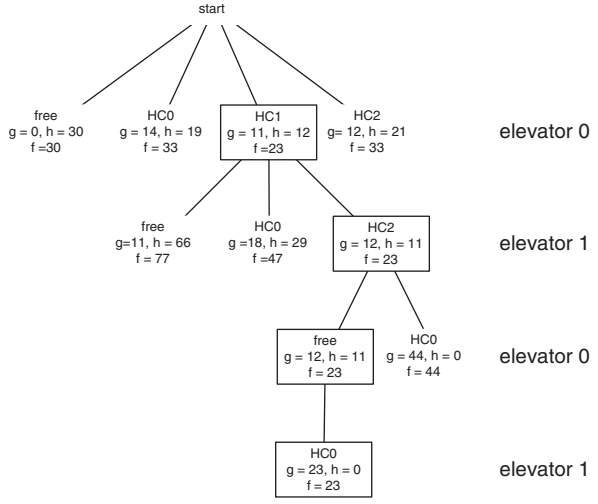

Fig. 4. Prioritised A* search tree for the example of Fig. 3. The tables show the values of s_n and $h_n(k)$ that are obtained on examining the node that is expanded at the first level of the tree. Note that in this example, two fewer nodes are expanded in the PA* search tree compared with the corresponding with the A* tree, although the same assignment results.

Table 3
The results of testing the prioritised A* scheduler for the example in Fig. 3 over a morning up peak period (during which time over 50 000 schedules are calculated) using three different methods of calculating h

Method of calculating h	Scheduling time (ms)	Number of nodes	Number of backtracks
Using both d_n and s_n	0.6	10	0
Using d_n only	1.7	24	7
$h = 0$	2.0	49	15

Note that in all cases, identical schedules were produced.

search when the number of tree levels is equal to the number of elevators. It is a matter of design to ensure that, for the elevator-dispatching problem, this can be achieved in the time available assuming some maximum number of hall calls. In practice, the authors have found that this requirement is easily met with the equivalent of modern desktop processing power in a range of installed elevator systems evaluated during the peak periods of operation (see the following section).

It is important to consider backtracking, as the A* search technique requires the search to return to a previously visited node in the tree should the current path being followed prove not to be the most promising. Clearly, the time available for the search could be exceeded if backtracking occurs and in such a case the elevators' first assignments may be unavailable if the n^{th} level has not been reached. However, by adopting the method shown in Eq. (2) for the calculation of h, which includes initial assignment estimates for all elevators, a first assignment is available even if the nth level of the tree had not been reached in the time available to the search. To reduce the possibility of the problem occurring, two enhancements were implemented. The first is to retain the solution found immediately before backtracking, although this is only strictly needed if the backtracking occurs to one of the first n levels. The second is to ensure that the quality of the calculated value of h is sufficient to avoid excessive backtracking. Additional computational effort is normally required in order to improve the quality of the estimate of h, but the solution adopted in the current work is to determine and store in look-up tables the latest calculations of the costs of each elevator answering hall calls before the search is carried out. At present, the costs are found by the same processor that calculates the schedules, but additional computational power could easily be brought to bear on the problem by determining the look-up table values on a second parallel system.

For the simplified elevator system example shown in Fig. 3, the calculation time and backtracking performance of PA* were compared with those obtained from an optimal search (where $h = 0$) and those obtained from a search using only the d_n values from Eqs. (1) and (2), thereby significantly worsening the quality of the estimates. The results are shown in Table 3 and demonstrate that by using the values of both d_n and s_n, a good estimation of h is obtained, directing the search to its goal state without any backtracking.

5. Results of the application of PA* to an installed elevator system

The performance of the PA* scheduler was assessed using data gathered over a number of days from installed elevator systems (Hamdi, 2001). To be more representative of the specification of installed legacy embedded systems that are common in installed elevator systems, were simulated on a Pentium II processor operating at 133 MHz. This extensive study has demonstrated that it is normally possible to find an optimal schedule using PA* in the critical time interval available. One such elevator system, originally commissioned in the mid 1990s by a leading UK elevator company, is shown in Fig. 5. To advance from the simplified scheduler described in the previous section to a full dynamic system able to cope with the scheduling of a real elevator system, a complete test bed was developed as shown in Fig. 6 (Mulvaney & Hamdi, 2003). To ensure the elevator dispatcher has the high-quality data it needs to provide suitable schedules, static and dynamic elevator models that together provide a complete elevator simulation, as well as a passenger model were developed and these are described in detail below.

1. *Static elevator model.* From a detailed statistical investigation of the data obtained from real elevator installations, the simplifying assumptions made in the previous section can now be replaced by empirical values. Table 4 shows a selection of the values used for the 18-storey building shown in Fig. 5. Note that commonly used constraints were also incorporated into this model, such as an elevator must answer all car calls before reversing direction and that fully loaded elevators are marked as unavailable.
2. *Dynamic elevator model.* In order to provide better estimates of trip times, the velocity, acceleration and jerk profiles of the elevator were modelled. When the assignment of an elevator to a hall call is being considered, but where the elevator is about to pass the floor at which the call was issued, times taken to halt or reverse need to be taken into account. Also, as the cost of an elevator moving to a hall call also depends on the number of car calls between the elevator initial position and the destination hall call, the elevator trip time to and from each car call is calculated rather than simply using the direct trip time to the hall call floor.

In addition, dynamic simulations of door movements and passenger entry and exit times were used in place of static values.

3. *Passenger model*. Information relating to passenger movements are not available directly from observations of the data collected from elevator systems, but can be estimated from observations of car call destinations, timings of hall call, elevator locations, door opening times and the activations of photocells mounted on car doors. How this information was used to develop statistical models of individual passengers and their intended movements through the elevator system is described in detail in Hamdi and Mulvaney (1998). The passenger model stores the arrival rates of passengers and their variations on a daily and weekly basis as gathered from a real installed system, and applies a Poisson distribution to generate the data for the model. Similarly, the destinations of passengers arriving at each of the floors in the installed system are recorded and then used to generate the car calls in the model with a probability of target floor proportional to its popularity in the real system. The passenger model is used to estimate the number of passengers both waiting for elevators and travelling in the cars, as well as their likely destinations. Statistical models are also used to predict whether new passengers will arrive in a time scale that will affect the current schedule being calculated, in which case the search can be amended to take into account both the hall calls and car calls that are likely to result. The model has also been adapted for use off-line in order to simulate the arrival of passengers according to appropriate statistical distributions. Such a model of passenger arrivals allows comparisons of the performance of alternative scheduling techniques (Mulvaney & Hamdi, 2003).

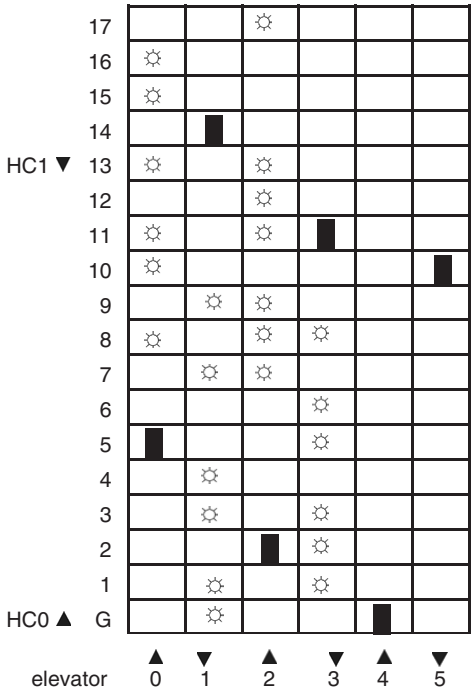

Fig. 5. Example of an installed elevator system in a central London office building with 18 floors, six elevators and a population of approximately 900 staff.

A typical PA* search tree for the elevator system is shown in Fig. 7. The PA* scheduler was transferred from simple to real-world examples by replacing, one at a time, the assumed values by those obtained from the static and dynamic elevator models. The resulting effect on the length of the passenger queues is illustrated in Fig. 8. It can be seen that, compared with the existing installed dispatcher, the PA* approach significantly reduces the number of queues containing two or more passengers, indicating that PA* answers hall calls more quickly and before queues

Table 4
Typical values provided to the scheduler for the building shown in Fig. 5 for some of the most frequently occurring passenger movements

Elevator operation	Duration (s)	Number of passengers
Answer a hall call	8.9	1
Answer a car call	6.4	1
Answer both a hall call and a car call	9.4	2
Door closing	2.7	0

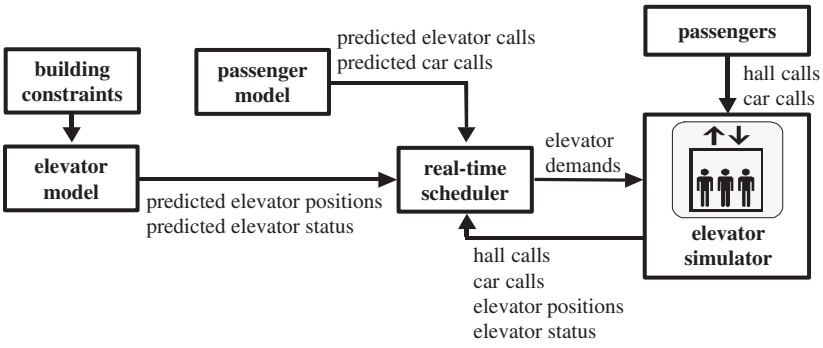

Fig. 6. Test bed for the real-time elevator dispatcher.

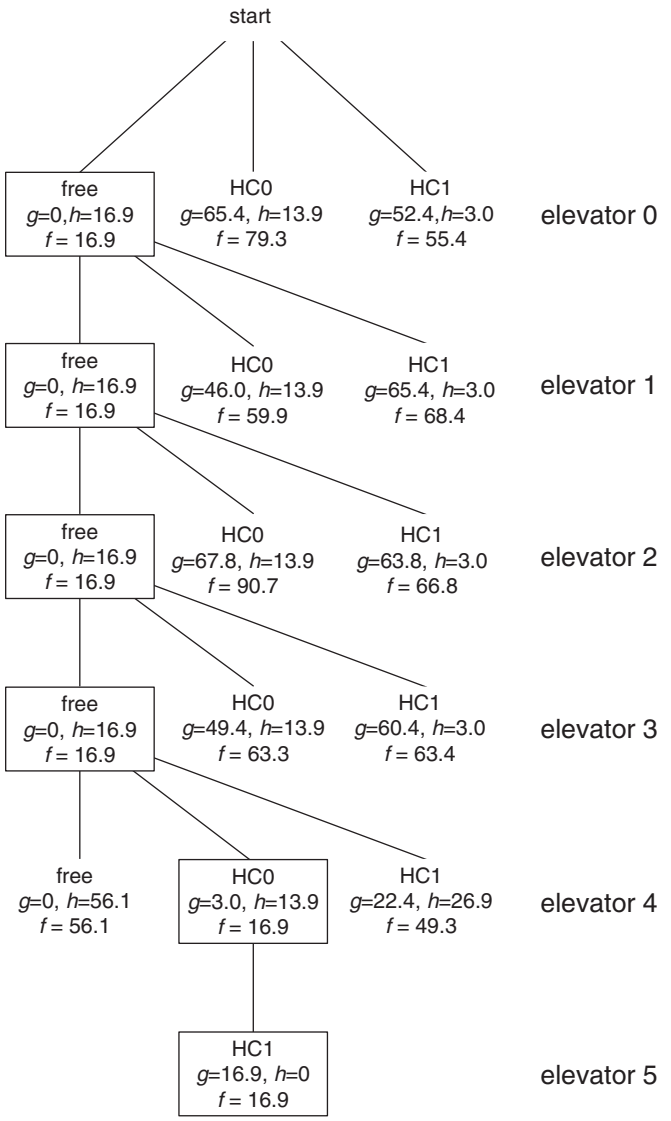

Fig. 7. A typical PA* search tree for the elevator system in Fig. 5.

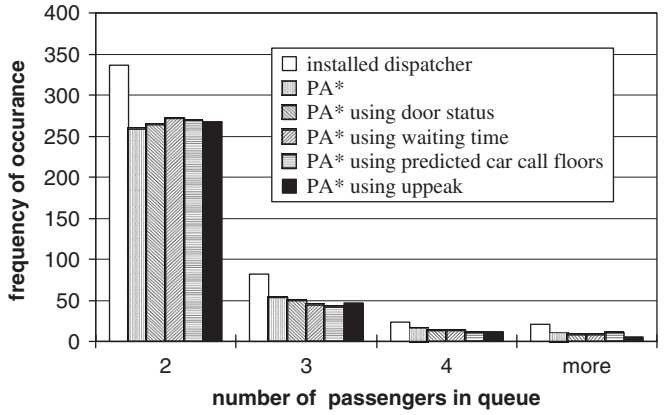

Fig. 8. Comparison of queue frequencies for all floors between 7 am and 8 pm for two weeks' of data, as a series of modifications is introduced.

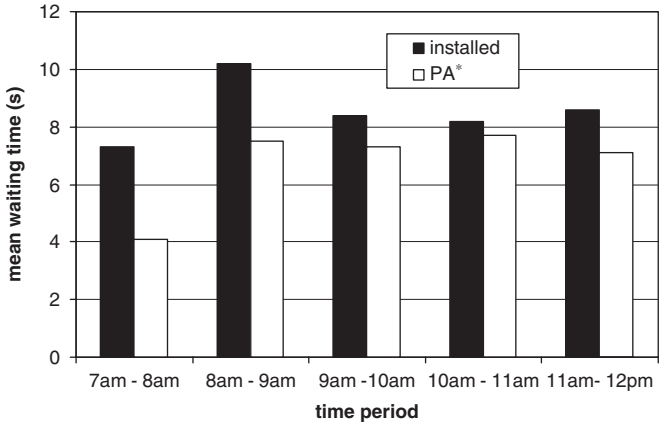

Fig. 9. Comparison of mean waiting times for the installed and PA* schedulers during the morning period. The reduction in waiting time achieved by the PA* is most apparent during the up-peak period when there is a significant influx of passengers at the ground floor.

have time to develop. Fig. 8 also shows the results of the investigations into the change in performance as further scheduler refinements were added; but no significant effect on in the scheduler's performance was detected. One possible reason is that the values of the times provided by the models are generally small compared with the elevator travel times, and hence improvements in their estimation do not significantly influence the performance of the scheduler overall. In Fig. 8, the first simulation run used the existing installed dispatcher with up-peak and fixed parking policies. The second used PA* with a fixed door delay; predicted car calls are assumed to be at the furthest floor in the direction of the elevator; waiting time is not included in the elevator trip duration to a hall call and no up-peak policy is used. The third run replaced the fixed door delays with actual elevator door delays, the fourth run adds hall call waiting time to the trip duration and the fifth uses predicted car call floors. The sixth run uses an up-peak policy similar to that used in the installed dispatcher.

Fig. 9 shows the differences in recorded waiting times for the installed and the PA* schedulers for a typical working morning. The difference in waiting time was most marked during the busiest main morning up-peak period between 7am and 10am where the waiting time was reduced by around 24% from 8.7 s for the installed scheduler to 6.6 s for the PA* scheduler. Over the whole working day, the reduction was around 13%, with the PA* scheduler achieving a passenger mean waiting time of around 7.8 s at the terminal floor, which compares favourably both with the installed scheduler's mean waiting time of 9.0 and the 15.0 s mean waiting time normally considered acceptable in a business type building (Barney & dos Santos, 1985). Fig. 10 shows that the reduction in waiting time is apparent for all floors, with the PA* scheduler achieving a reduction both in the number of passenger waiting time intervals greater than 15 s and in the number of queues of two

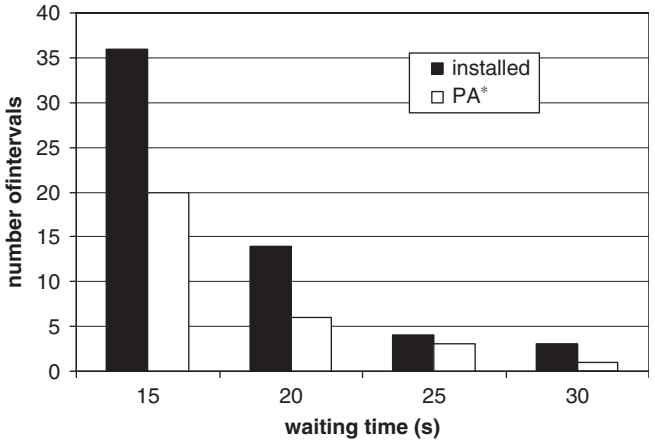

Fig. 10. The graph shows the number of waiting time intervals that are longer than the durations as shown on the horizontal axis. It can be seen that, compared with the solution produced the installed scheduler, PA* significantly reduces the frequency of longer passenger waits.

passengers. When comparing the schedules produced by PA* using both d_n and s_n values with PA* using only d_n values, then, over a full day, typically less than 0.05% of the schedules were not identical.

6. Conclusions

A real-time search method based on A* has been introduced and its operation described in this paper. PA* search is applied to the elevator dispatching problem, giving priority to the assignment of elevators to hall calls, ensuring that each elevator is given its first assignment and has been shown to operate successfully in a realistic real-time environment. In its application to an installed elevator system, the results demonstrate that its performance improved when data were made available regarding both the predicted passenger movements and the current state of the elevator system, such as hall call waiting time, passenger entry and exit times and elevator journey time.

Choosing to use good quality estimates for the remaining path length allows PA* to converge to the goal state with a minimum of backtracking. Since the scheduler described suffers little from backtracking, it is generally possible to truncate the search after reaching that number of levels equal to the number of elevators and still produce a suitable partial assignment. Moreover, for a given number of elevators and hall calls, the number of nodes required to produce the first assignment of each elevator is known and consequently it is possible to determine the maximum time the schedule will take to calculate. This is a particular advantage of PA*, since, as shown in the survey of previous work, schedulers cannot normally be interrupted and still be expected to give the most suitable assignment. A further advantage of PA* is that only nodes in one level of the tree need be sorted in each generation, thereby avoiding the need to sort the whole set of open nodes as is required by A*.

To demonstrate the general applicability of the PA* scheduler requires further tests on a wide range of building function, with different patterns of occupancy and under varying traffic conditions. Since the assignment is often suitable even when truncated, the search method will be appropriate in other scheduling problems where more than one assignment is required for each resource, particularly where the search needs to be completed within a critical time period.

Acknowledgement

This work was supported in part by the Express Lift Company, Northampton, UK (a former subsidiary of Otis Elevator Company).

References

Barney, G. C. (2003). *Elevator traffic handbook*. London: Spon Press; 2003.

Barney, G. C., & dos Santos, S. M. (1985). *Elevator traffic analysis design and control* (2nd ed). London: Peter Peregrinus.

Beielstein, T., Ewald, C. -P., & Markon, S. (2003). Optimal elevator group control by evolution strategies, *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago.

Clark, G., Mehta, P., & Prowse, R. (1994). Knowledge-based elevator controller. *Control 94*, Coventry, UK (pp. 42–47). March.

Cortes, P., Larrañeta, J., & Onieva, L. (2003). A genetic algorithm for controlling elevator group systems. *Seventh internationalwork-conference on artificial and natural neural networks*, Menorca, Spain (pp. 313–320). June.

Crites, R., & Barto, A. (1998). Elevator group control using multiple reinforcement learning agents. *Machine Learning*, *33*, 235–262.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman.

Gudwin, R., Gomide, F., & Netto, M. (1998). A fuzzy elevator group controller with linear context adaptation. In: *Proceedingsof the 1998 IEEE International Conference on fuzzy systems*, Anchorage, USA (pp. 481–486).

Hamdi, M. (2001). *Intelligent real-time lift scheduling system*. Ph.D. thesis, Loughborough University, UK, 2001.

Hamdi, M., & Mulvaney, D. J. (1996). Visual interactive lift simulator, *Elevcon 96*, Barcelona, Spain, (pp. 87–94).

Hamdi, M., & Mulvaney, D. J. (1998). Simulation of lift systems and modelling of passenger movement. *International Journal Elevator Engineering*, *2*, 1–18.

Hamidzadeh, B., & Atif, Y. (1996). Time controlled dynamic scheduling of aperiodic real-time tasks. *second IEEE International Conference on Engineering Complex Computers*, Montreal, Canada (pp. 323–330).

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for heuristic determination of minimum cost paths. *IEEE Transactions On System, Science and Cybernetics.*, *SSC-4*(2), 100–107.

Kim, C., Seong, K. A., Lee-Kwang, H., & Kim, J. O. (1998). Design and implementation of a fuzzy elevator group control system. *IEEE Transactions On System, Science and Cybernetics.*, *28*, 277–287.

Korf, R. E. (1988). Search: A survey of recent results. In H. E. Shrobe (Ed.), *Exploring artificial intelligence* (pp. 197–237). Los Altos, CA: Morgan Kaufmann.

Korf, R. E. (1990). Real time heuristic search. *Artificial Intelligence*, *42*(2–3), 189–211.

Mouaddib, A.-I., & Gallone, J.-M. (1996). Progressive scheduling for real-time artificial intelligence tasks. *second IEEE International Conference on Engineering Complex Computers*, Montreal, Canada. (pp. 95–98), 1996.

Mulvaney, D. J., & Hamdi, M. (2003). *Real-time dynamic scheduling and its application to lift scheduling*, *Lift report* 1, (pp. 24–29). January.

Mulvaney, D. J., & Sillitoe, I. P. W. (1992). *An assessment of new techniques for lift scheduling*. Technical report, Loughborough University, UK, August.

Pepyne, D. L., & Cassandras, C. G. (1997). Optimal dispatching control for elevator systems during uppeak traffic. *IEEE Transactions on Control Systems Technology*, *5*, 629–643.

Shah-Hamzei, G. H., & Mulvaney, D. J. (1999). On-line fuzzy decision tree learning for global path planning. *Journal of Engineering Applications of Artificial Intelligence*, *12*(1), 93–109.

So, A., & Chan, W. (1997). Dynamic zoning in elevator control. *Elevator World*, *XLV*(3), 136–139.

Sorsa, J., Siikonen, M.-L., & Ehtamo, H. (2003). Optimal control of double-deck elevator group using genetic algorithm. *International Transactions in Operational Research*, *10*(2), 103–114.