

# An AI-based Approach to Destination Control in Elevators

Jana Koehler and Daniel Ottiger  
Schindler Lifts Ltd.  
Research and Development  
CH-6031 Ebikon, Switzerland \*

## Abstract

Not widely known by the AI community, elevator control has become a major field of application for AI technologies. Techniques such as neural networks, genetic algorithms, fuzzy rules, and, recently, multi-agent systems and AI planning have been adopted by leading elevator companies not only to improve the transportation capacity of conventional elevator systems, but also to revolutionize the way in which elevators interact with and serve passengers.

In this paper, we begin with an overview of AI techniques adopted by this industry and explain the motivations behind the continuous interest in AI. We review and summarize publications that are not easily accessible from the common AI sources. In the second part, we present in more detail a recent development project to apply AI planning and multi-agent systems to elevator control problems.

## 1 Challenge of Efficient Vertical Transport

Like many other industries, the elevator industry is currently facing two main challenges: (1) The continuous pressure to lower construction costs of buildings requires that the core space occupied by an elevator installation be reduced and that transportation performance be significantly improved. (2) Increasing competition requires a diversification strategy to provide new and individually tailored services to passengers.

Most of the physical components of an elevator (e.g., drives and shaft installations) are invisible to the passenger and can be exchanged without even being noticed, but the available services and their quality determine how an elevator installation is perceived by the customer.

Not surprisingly, elevator companies have developed a demand for new technologies and are adopting AI techniques to address the aforementioned goals. To understand the underlying control issues, let us recall how elevators usually work.

### 1.1 Machines Interacting with Humans

Most buildings are equipped with an elevator group installation comprising two to eight cars. Humans interact with these systems by pressing a call button, which issues a *pickup call* at the floor in question. In many cases, there are two call buttons—up and down—for humans to indicate their desired travel direction. In larger buildings, a display may give the waiting passenger information about where the cars are currently located in the building. More commonly, passengers do not know where the elevators are, in which direction the cars are going, nor which car will serve them. Thus, while waiting, users typically scan the elevator doors. If made to wait too long, they often

---

\*The views in this article represent the personal views of the authors and not necessarily the official views of Schindler Lifts Ltd. Correspondence address and current affiliation of the first author: IBM Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland, email: koe@zurich.ibm.com

become impatient and press the call button again, sometimes even in both directions in their uncertainty whether the system has recorded their call. People have been observed to press all possible buttons again and again, apparently in the irrational belief that this will make an elevator arrive sooner.

When the elevator arrives, passengers enter the car and press the desired floor buttons, which issues a *cabin call* from that particular car. Sometimes passengers change their minds and press additional floor buttons, or they might hold doors open for other passengers, or get off at a different floor than originally intended.

Passengers throughout the building interact with each other when using elevators, but they are usually not aware of this interaction. For example, holding doors open is a courtesy to a particular passenger, but it is not cooperative behavior towards the passengers waiting on other floors for the same elevator. From a passenger's point of view, a short *waiting time* and a short *journey time*, i.e., the time to reach the destination floor with as few intermediate stops as possible, are desirable whatever the demand for elevator transport may be.

Elevator control software faces the same scenario, but from a very different perspective. It considers the cars, each having a number of pickup calls and cabin calls to serve, the current floor position, and the current travel direction of each car. Some of the floors to be served may lie "ahead" of the car given the current travel direction, others may lie "behind" it, requiring the car to turn around and head in the opposite direction. Usually, the time of a pickup call is registered, which allows predictions to be made as to the waiting times of passengers at that particular floor, but the control software typically does not know how many passengers are waiting there and to which floor these passengers want to travel. Based on this limited information, the control has to *dispatch* cars, i.e., to select a particular car to serve a particular pickup call and redispach cars when the traffic situation changes. In any case, floors should be served as quickly as possible in order to minimize a passenger's waiting time. However, it is common knowledge in the elevator industry that minimizing waiting times alone is not sufficient to obtain a good control algorithm.

Two main criteria are used to evaluate an elevator control algorithm. First, the so called HC5% value specifies the handling capacity of a group of cars within five minutes in terms of the percentage of the building population that is served. It is either measured empirically in a simulation run or computed directly using agreed-upon analytical methods [4, 39]. A good installation should be able to transport at least 14% of the building's population within five minutes. For example in the case of 2000 inhabitants, a group of cars should be able to transport at least 280 passengers within five minutes. Second, the average and maximum waiting and journey times are determined (again either empirically or analytically), and this in turn determines the quality of service, which is also directly perceived by the passengers. The lower the resulting times, the better a control algorithm is rated. Short journey times relate directly to a high HC5% value, but humans care much more about short waiting times. As for waiting times, it is not important to reduce average waiting times from 32 to 28 seconds, for example, but it is psychologically very important to avoid long waiting times of up to 60 seconds and more.

In this scenario, any technology that can address the following questions is of interest to elevator companies:

- What is the objective function for a group dispatching algorithm? Almost no information is published by companies about the objective functions they use in their control algorithms. Usually, a vague "combination of waiting and journey times" is minimized, but which function would yield the best possible results still seems to be an open question.
- How can a control system acquire additional information about passenger needs? In particular, how can it find out how many passengers are waiting at a floor, how fully loaded a car is, and where the passengers want to go?
- How can the performance of the controller be improved? Is it possible to detect and predict patterns of traffic based on the currently available information and/or previously learned

patterns? How could such information be exploited by a control algorithm?

- How can passenger interfaces be improved beyond the simple buttons? How can passengers with special needs be better served?

In the following, we give an overview of AI-based approaches that have been explored by elevator companies in the past to address these issues.

## 1.2 How many Passengers are Waiting?

The *acquisition of traffic information* is the attempt to capture precise data about the entry and destination floors of passengers [24]. In the simplest case, manual surveys are conducted by having human observers count passengers going in and out of cars. The observers are either placed in the cars or at the main lobby. Alternatively, videotaped recordings can be manually analyzed or a counting unit can be linked to the control system recording pickup and cabin calls, which would allow some conclusions to be drawn about the relationship between entry and exit floors. In addition, accurate weighing devices in the car floor would allow the cabin load to be determined before and after a stop and thus a more or less accurate count of passengers to be derived from the measured loads. More advanced computer vision technology has been proposed to detect the number of persons waiting at a particular floor, but tracking individuals moving on and off the scene proved to be a tricky issue as the lighting conditions vary from building to building and waiting passengers are much less static in their behavior than originally assumed. Vision-based approaches reported an accuracy of 80–85% [28, 36], but are still too costly to be used regularly.

Counting and measuring attempts aim to calculate the *passenger arrival rate* at each floor, i.e., how many passengers arrive per minute averaged, for example, over the past five minutes, as well as the *probability distribution* over entry and destination floors for all possible floor pair combinations.<sup>1</sup> This information can be used directly in a control algorithm, e.g., by serving floors first that have a high arrival rate [32]. Another possibility is to calculate more abstract *traffic patterns* from this data.

## 1.3 Traffic Patterns, Expert Rules, and Learning

Traffic patterns attempt to capture the flow of passengers at a more abstract level. Three main patterns can be identified:

- *Up-peak*: Passengers enter at the lobby floor and request upwards transportation.
- *Down-peak*: Passengers request downward transportation from all floors to the lobby.
- *Interfloor*: Passengers request upward or downward transportation between floors, but not to or from the lobby.

Other patterns can be defined by specifying the proportions of the three basic patterns that constitute them, e.g., a prototypical *noon peak* pattern at lunch time can be defined consisting of 45% up-peak, 45% down-peak, and 10% interfloor traffic. Of course, traffic patterns vary throughout the day and a real traffic pattern will always be some combination of the basic patterns.

In the late 1980s, traffic patterns were an ideal starting point for the development of *expert systems* to dispatch elevators [1, 38]. Given (1) a set of predefined patterns, (2) passenger counts gathered over a certain period of time, and (3) rules acquired from human lift experts, the expert system would determine the currently predominant pattern and issue predefined dispatching decisions related to this pattern. Unfortunately, several problems prevented a wide adoption of

---

<sup>1</sup>This probability distribution represents information such as “currently, when picking up passengers at the main lobby, 80% want to go to the restaurant floor, whereas 5% each want to go to floors 4, 7, 8, and 10”.

expert systems. First, identifying a clear pattern proved to be nontrivial, as most buildings usually show a varying mix of different traffic flows. Second, the well-known problems of capturing expert knowledge, accounting for inconsistencies, and maintaining the underlying rule set made the approach difficult to use. A further difficulty arose from the fact that although certain traffic situations might look very similar, they can require very different dispatching decisions.

As a result of this experience, *fuzzy logic* and *fuzzy rules* became popular in the early 1990s. *Fuzzy logic* was used to describe traffic patterns at a more fine-grained level, and fuzzy rules were embedded into expert systems to implement more flexible inference mechanisms [40, 41]. A typical approach of how fuzzy rules are used for elevator control is described in [31, 34]. With this approach, the intensity of each traffic flow is described with fuzzy variables such as *high*, *low*, or *medium*. Fuzzy rules are then used to determine the predominant traffic pattern, which in turn influences the elevator control. For example, a simple rule such as “*if intensity is heavy, incoming traffic is high, outgoing traffic is low, and interfloor traffic is low, then traffic type is heavy up-peak*” would trigger a specific control algorithm that sends any idle elevators immediately down to the main lobby.<sup>2</sup>

The usefulness of abstract patterns to guide predefined dispatching decisions proved to be rather limited. Many experts believe today that it is impossible to make predictions about traffic flows that are accurate enough to provide useful information.

In the mid-1990s, *neural networks* became a popular way to allow for certain learning abilities of the control algorithm. In an early approach described in [37], neural networks were used to identify one out of five predefined traffic patterns after being trained on a set of simulated situations. This approach was still very similar to the earlier developments of expert systems and thus did not overcome the limitations of pattern-triggered rules.

A more interesting and much more comprehensive development based on neural networks has been conducted by OTIS [25, 42]. With this approach, the dispatching decision is based on the estimated *remaining response time* (RRT) of each car, i.e., the time a car still needs to travel before it reaches the pickup floor. This time is estimated based on the distance to travel and the intermediate stops a car has to execute before it reaches the desired floor. Uncertainty arises from the unknown number of passengers boarding and exiting during a stop, which makes it difficult to predict when a car will depart, and from the unknown destinations chosen by subsequent passengers, which may well result in additional (yet unknown) intermediate stops. The task of the neural network is to predict the RRT as accurately as possible. The network is trained in a simulation environment, which allows the predicted RRT to be compared with the actual one. The learning process attempts to minimize this error and hence improve the accuracy of subsequent forecasts.

The network design focused on two goals: First, the number of nodes in the input layer should be independent of the number of floors in a building [43]. Second, a set of factors was determined to enable the system to make accurate RRT forecasts. The developers selected 47 factors to be represented, yielding 47 input nodes. These nodes represent such parameters as the distance to the call floor, the estimated number of passengers, or the number of cabin calls, but the detailed design of the network was kept confidential—in particular no information about the activation functions and their thresholds is divulged. The output layer comprised a single node yielding the estimated RRT. Weights were calculated by training the network with simulated traffic data. Based on these design decisions, a *perceptron* was developed, which yielded predictions improved up to 20% on average.

Given that perceptrons are limited to learning linearly separable functions, the question arose whether perceptrons are adequate representations of the RRT estimation function. Consequently, the perceptron architecture was extended to various feedforward networks that differ in the structure of their hidden layer(s) and the activation functions used. The training of these networks proved to be difficult, and the authors report that they were unable to improve the results ob-

---

<sup>2</sup>Fuzzy logic is also commonly used at the mechanical level in drive controllers to allow for a smooth acceleration and deceleration of drive engines, but this application area is beyond the focus of our paper.

tained with the simple perceptron. Although sometimes slightly more accurate RRT forecasts were obtained, these did not automatically result in better dispatching decisions. Maintaining complex neural networks in real buildings also proved to be nontrivial as this requires an online training process. Actual building data, which can often be incomplete and inaccurate due to the given uncertainty of observations, have to be fed into the training algorithm. Online training processes are a difficult endeavor in general, and in this particular application, it was impossible to guarantee they would produce useful results.

In another approach, neural networks have been used in a *reinforcement learning* framework for elevator dispatching [10] in a building with ten floors served by four cars. The minimization of the squared waiting times was taken as the objective function to achieve short waiting experiences for passengers and to provide fair service. The neural network comprised 47 nodes in the input layer, 20 sigmoid nodes in a single hidden layer, and two linear output nodes representing one of the possible actions a car can take in a given situation: *stop at next floor* or *continue*. For each car, a neural network is needed. In the input layer, eighteen (nine pairs of) nodes were used to encode information about the nine “down” hall buttons. One node in each pair is a Boolean representation of whether the button was pressed, whereas the other node represents the time elapsed since the button was pressed. Sixteen units are used to represent the possible directions and locations of the car, and ten units represent the ten floors. Three more units are needed to represent whether the car is at the highest floor with a waiting passenger and where the passenger with the longest waiting time is located. Note that this network design depends on the size of the building, in contrast to the OTIS approach. The neural network was trained for 60,000 hours of simulated elevator operation using a down-peak traffic pattern of varying intensity. The algorithm was then compared with other well-known control algorithms using pure down-peak and mixed up- and down-peak patterns, for example (i) a static *zoning* algorithm (see below), (ii) an algorithm that serves the highest unanswered floor first, (iii) an algorithm that attempts to maintain an equal load among cars, and (iv) an algorithm that serves the longest-waiting passenger first. These algorithms are interesting from a theoretical point of view, but much too simplistic in order to achieve a high transportation performance, i.e., it takes not too much to beat them.

The results showed a reduction of average waiting times from 21 seconds for the worst algorithm to 14 seconds for the reinforcement learning approach on the pure down-peak. A similar improvement was obtained for mixed traffic, but average waiting times remained about 20 seconds. It is unknown whether this approach has ever been put into practice. The considerable training effort and the building-dependent network design are clearly drawbacks of this solution.

To address the difficulties associated with the design, training, and maintenance of neural networks, *genetic algorithms* have recently been explored [22, 27, 44]. The chromosomes represent possible solutions to a given dispatching problem with random or expert-generated solutions as initial seeds. *Fitness* is evaluated by calculating the waiting times for passengers based on each solution. The stochastic search employed in genetic algorithms often produces better dispatching solutions than those generated by a predefined set of dispatching rules [33], but thus far improvements have only been reported for simulated traffic scenarios. Development is apparently ongoing.

The uncertainty of information about a traffic situation and how it will develop calls for approaches that can *reduce* the degree of *uncertainty*. A first step in this direction is to define various levels of service for each floor [27, 44]. Using genetic algorithms, the fitness function is extended to a weighted sum of waiting time, journey time, and estimated passenger load of each car. The weights for each factor are set differently for the various floors, thereby assigning higher priority to selected floors. At the same time, the genetic algorithm is biased towards certain solutions. As practical experiments have shown that weights have to reflect changing traffic conditions in a building, genetic algorithms are also used to determine appropriate weights for dynamically adjusting fitness functions [44]. The usefulness of genetic algorithms is not yet clear. Finding the right combination of specific crossover, mutation, and selection methods yielding good dispatching decisions poses a challenge in this domain.

Experience in this industry shows how difficult it is to implement intelligent, reliable, and self-adaptive autonomous systems in the real world with the currently available AI techniques. As Crites and Barto [10] put it, “*The elevator domain poses a combination of challenges not seen in most RL (reinforcement learning) research to date.*”

However, the difficulties in achieving considerably better dispatching decisions appear to emerge primarily from the inherent uncertainty associated with the problem. Assumptions about the unknown destinations of passengers, the number of passengers currently using the system, and the evolution of the traffic in the immediate future must by nature remain extremely vague. It is also doubtful whether a clear theory will ever be proposed that is able to categorize the huge number of possible traffic situations in a building and map them to a fixed set of rules defining how the control system should react. Recent developments, therefore, focus on technical solutions that help reduce the amount of uncertain information by imposing a more disciplined behavior on passengers. We will discuss these development trends in more detail below.

## 1.4 Combinatorial Optimization of Travel Routes

As early as 1970, elevator dispatching was characterized as a combinatorial optimization problem, which could be addressed using heuristic search techniques [9]. However, at that time, the available computing power did not allow the investigation of more than toy examples, and the assumption of complete knowledge of passenger destinations made such approaches rather unrealistic. An even more unrealistic assumption was made in [19], which proposed algorithms to generate optimal policies for uncertain passenger destinations, but with elevators of infinite capacity, i.e., there is always sufficient space to accommodate all passengers desiring transport.

Assuming comprehensive information about the floors to be served appears to be realistic provided that the controller immediately redispaches cars the moment new information becomes available. Genetic algorithms are proposed in [33] as an appropriate stochastic search method to find near-optimal solutions under these assumptions. Alternatively, an algorithm using *minimin lookahead search* [17] with *alpha pruning* is proposed in [8]. This search algorithm by Richard Korf, which adapts the minimax algorithm for two players to the single-agent case, looks forward a fixed number of moves and backs up the minimum cost value of each frontier node. Once the backed-up values of the children of the current state have been determined, a single move is made in the direction of the best child, and the search process is repeated. With a monotonic cost function used for heuristic evaluations of interior nodes, pruning of frontier nodes with branch-and-bound is solution-preserving and yields an enormous acceleration of the search because not all frontier nodes require visitation. Although no details are given in [8], one can imagine how this search algorithm could be used to compute dispatching decisions online: Given a number of cars, information about their current travel routes, and a set of unanswered calls, lookahead search could be used to assign an optimal sequence of floors to each car. In this sequence, only the first floor (the “first move”) will be executed, then the algorithm would be called again to reschedule the remaining floors, taking into account fresh information about the current position of cars and new incoming calls.

To reduce uncertainty regarding destinations of passengers, two approaches are currently pursued: *dynamic zoning* and *destination control*. In dynamic zoning, elevators serve only a restricted range of floors, e.g., 7 – 14, and passengers are supposed to board the elevator serving the zone in which their destination floor is located. Zones have to be dynamically adjusted depending on the traffic flow in a building, which requires passengers to carefully observe zone displays located above car entrances. In [7, 23], genetic algorithms are used to establish rules about how to arrange zones depending on the traffic flow observed, whereas [26] proposes a systematic, but incomplete search method to determine the best arrangement of zones. In order to obtain more accurate information about destinations and the number of passengers desiring transport, passengers are requested to indicate their destination via an input terminal in [2, 13]. The zones are adjusted according to the destinations registered, but passengers still have to watch for the correct elevator to take. Practical experience with zoning shows increased transportation capacity during periods of massive up-peak

traffic, but users do not seem to accept this system readily. Moreover, this approach is difficult to extend to heavy interfloor traffic, which requires zoning information to be computed and displayed at all floors, not only at the main lobby.

In contrast to *zoning*, which can work with or without preregistered destination information, *destination control* always requires passengers to register their destination floor before they get on board. Based on the destination information, the control *allocates* the passenger to a particular car, but cars are not limited to serve particular zones.<sup>3</sup> The allocation of a passenger to a car is fixed and can no longer be changed, in contrast to conventional elevators where the controller can redispach cars at any time. The immediate allocation of a passenger to a particular car is similar to the so called *early car announcement* feature that has become popular with conventional dispatching. Early car announcement will tell a passenger immediately which car has been dispatched for pickup. The main motivation behind this feature is to enhance passenger convenience, but with unknown destinations, early car announcement prevents the redispaching of cars, thereby downgrading the system performance very easily.

When destinations are known in advance and no floor buttons can be pressed by passengers traveling inside a cabin, nearly complete and reliable information about a given traffic situation is available, making the dispatching problem much more amenable to combinatorial search techniques. With reduced uncertainty about the traffic situation, trying to calculate optimal travel routes for elevators makes much more sense. Although the notion of destination control has existed in the elevator industry for so long it is difficult to trace it back to a specific inventor, designing a user interface that is not only accepted by humans but also amenable to the development of appropriate allocation algorithms remains a challenging endeavor. Despite these difficulties, having passengers register their travel wishes in advance allows attractive new features to be added to elevator systems. Thus far, only one solution has found its way onto the commercial market, which is described in the following.

## 2 Destination Control Systems

The first destination control system, Miconic-10<sup>TM</sup>, was introduced by Schindler to the market in 1996. Until today, more than 1500 elevators have been equipped with Miconic-10<sup>TM</sup>. They are operating successfully worldwide, particularly in large buildings with thousands of inhabitants and multiple elevator groups where they sometimes can achieve up to twice the HC5% value of a conventional dispatching algorithm.

A ten-digit keypad is installed in front of the elevator group where passengers indicate the floor to which they wish to travel, e.g., 22, cf. figure 1. Upon receiving the destination, the elevator control system selects an elevator to transport the passenger using a heuristic allocation algorithm [12]. Given the entry and destination floor of this passenger, the algorithm attempts to fit the new passenger into the current travel routes of all cars at the earliest time possible. For example, let us assume a passenger wishes to go downwards from floor 5 to floor 2. Two elevators are available: car A is currently passing floor 5 heading upwards to serve floors 7, 9, and 10, while car B is heading downwards to floor 0 and is currently passing floor 7. Car B could obviously stop on its way down and pick up the new passenger immediately, whereas car A first has to finish its upward travel until it can turn and serve this new passenger in the downward direction. The control system predicts the waiting time of the new passenger and the possible delay of other passengers allocated to this car if the current travel route were to be modified to serve the new passenger. Using this information, the car with the shortest combined waiting time and delay is allocated to the new passenger. *Capacity limitations* and *direct travel* are elementary constraints satisfied by any allocation, i.e., the allocation scheme guarantees that passengers are allocated to a car only if

---

<sup>3</sup>We adopt the term *allocation* instead of *dispatching* because the process is somewhat orthogonal; cars are not dispatched in response to pickup calls; rather, passengers are allocated to cars traveling dynamically recomputed routes.



Figure 1: A telephone-like ten-digit keypad allows passengers to enter their destination before they enter the elevator. A display informs passengers of the elevator to which they have been allocated.

enough space is available (based on the number of previously registered and allocated calls to this car), and that passengers will never travel opposite to their desired travel direction.

The elevator identifier, usually a capital letter such as A, B, C, ... is displayed on the input terminal to advise the passenger which elevator to take. Instant allocations of passengers within less than one second are necessary for two reasons: First, a passenger becomes impatient if the terminal does not respond immediately and might even wonder whether the system has broken down. Second, it is desirable for the passenger to move away from the terminal as quickly as possible to free it for other arriving passengers. Note that only a few terminals are available per floor. An indicator in the door frame of the car confirms the destinations this elevator will serve, i.e., upon boarding passengers can assure themselves that the car will stop at the desired floors. An alternative interface based on touch screens was proposed recently in [11], but it requires that passengers first select the zone to which they want to travel and then touch the desired floor name or number displayed next.

By using identification devices such as smart cards, pin codes, or modern cell phones, passengers can even be recognized on an individual basis and more individually tailored services can be offered in future destination control systems:

- *Access restrictions:* Modern buildings are often occupied by very different types of tenants, e.g., there may be shopping and entertainment zones, housing areas, and offices spread over various floors in a building. For safety and privacy, it is desirable that elevator controls implement access restrictions to certain floors, e.g., some floors are not served as long as unauthorized passengers are traveling in the car.
- *VIP service:* A passenger can be identified as a VIP to be served with highest priority by the elevator system. For example, fire fighters or emergency medical personnel would be entitled to VIP service.
- *Separation of passenger groups:* Some groups of passengers should not share an elevator. For example, the room service staff delivering breakfast and a housekeeper emptying trash bins should not meet in the elevator of a hotel for hygienic reasons. This means that, if a passenger who belongs to a particular prespecified group requests transportation, the control must choose an elevator such that no encounters between conflicting passengers can occur inside a car or during boarding.

Given this information, the elevator control software has to allocate the passenger to an elevator such that all requirements be satisfied. Today, elevator systems can offer these services only

to a very limited extent by permanently or temporarily restricting the use of cars. For example, today’s VIP service is implemented by taking an elevator out of standard service, then sending this elevator to the VIP passenger, and—after the passenger has arrived at the desired destination—returning the elevator to standard operating mode. This restricted usage of elevators dramatically impairs the transportation performance of an elevator group. The algorithmic methods used in the elevator industry so far have not allowed these functionalities to be integrated directly into the normal operation of a group of elevators. The dispatching decisions become much more complicated as more constraints have to be observed, which could not easily be added to the currently used allocation scheme. Furthermore, based on heuristics no optimal dispatching decision can be constructed. Thus, a new allocation algorithm for destination control was required, which we will present in the following section.

## 2.1 Destination Control: An NP-hard Online Problem

The development of the new destination control algorithm presented in this paper was driven by a formal approach. In studies conducted in 1998 and 1999, the complexity of the problem was investigated and the NP-hardness of the problem was proven, even for the case that no additional service constraints such as space restrictions, direct travel, or the separation of passenger groups are imposed, [30].

The *allocation problem with destination calls* can be defined as follows: Given a number  $n$  of destination calls with boarding floor  $b$  and exit floor  $e$  as  $(b_1, e_1), (b_2, e_2), (b_3, e_3), \dots, (b_n, e_n)$  we wish to compute a totally ordered sequence of stops  $s_1, s_2, \dots, s_k$  such that each  $s_i$  corresponds to a given boarding or exit floor (no unnecessary stops should be contained in the sequence) and where each  $b_i$  precedes each  $e_i$  (passengers must obviously be picked up first and then delivered to their destination).

If we wish to find stop sequences of minimal length, one can easily prove the problem to be NP-complete by a reduction from feedback vertex set [30]. Another NP-hard graph-theoretical problem closely related to ours is the *minimum point-to-point connection*, which has been proven to be efficiently approximable. One can find several graph-theoretical problems as well as TSP variants and vehicle-routing problems that address certain aspects of our problem, but none of them meets all requirements exactly. Capacity limitations of cars and the fact that grouping passengers together changes the individual transportation costs for each passenger due to longer door opening times or additional intermediate stops make this problem different from all graph-theoretical problems to our knowledge.

Based on these results, a comparative analysis has been conducted, which modeled destination control in terms of a *planning* problem, a *scheduling* problem, and a *constraint satisfaction* problem [29]. Modeling the problem from a planning perspective seemed to be the most natural approach. The initial state of the problem is described by the current distribution of passengers and elevators in the building. The goal state is any state satisfying that all passengers have been delivered to their destination floors. The set of actions specifies what an elevator can typically do: stop at a floor, travel up or down, open and close doors. The service constraints are modeled in the preconditions of the actions [16].<sup>4</sup>

There are two subproblems to be addressed when developing a new allocation algorithm: The first is the *static, offline optimization problem* for one elevator, which requires an optimal sequence

---

<sup>4</sup>The domain model was published using PDDL, the Planning Domain Definition Language used in the planning competitions [21, 3] to define precisely the services with which we wished to augment the destination control system. As PDDL is a first-order language without function symbols, not all relevant properties of this application could be represented. For example, capacity constraints of cars and cost information reflecting waiting and journey times of passengers had to be omitted. Thus, this representation was not suitable for developing a domain-specific planning algorithm, but it was used in the AIPS-00 planning systems competition and it has also been considered in topological and complexity-theoretical investigations of planning benchmarks, where it was found to be one of the hardest domains currently available [15, 14]. The complete domain can be downloaded from <http://www.informatik.uni-freiburg.de/~koehler/elev/elev.html>. Details about the planning competition 2000 can be found at <http://www.cs.toronto.edu/aips2000>.

of stops to be computed for a given, fixed traffic situation in a building, i.e., we do not yet consider the problem that the traffic situation is constantly changing and the sequence of stops has to be revised accordingly. The second is the *dynamic, online allocation problem* for several cars, which must cope with the immediate and unknown changes of traffic situations. In the following subsections we will review these two problems in more detail and present our solutions. We focused on achieving the following two goals:

1. An algorithm should be developed that allows new services to be added to destination control. It should compute allocations of passengers to cars such that the resulting control achieves a quality of service (HC5%, waiting and journey times) as least as good as the original Miconic-10™ allocation scheme—but additionally offering the new service functionalities.
2. The algorithm should be extendable to deal with multi-deck elevators, which serve several floors simultaneously. The allocation algorithm not only has to decide to which elevator a passenger should be allocated, it must also decide at which deck the passenger will board. The deck information is hidden from the passenger and can be revised until the elevator arrives at the pickup floor, but the control has to make sure that each passenger boards the correct deck. For example, a double-deck car sent to pick up a passenger at floor 5 on its upper deck has to stop with its lower deck at floor 4, thus serving floors 4 and 5 at the same time. Prior to our development, no technical solution for destination control with multi-deck systems was available. A multi-deck elevator offers interesting new solutions to the problem of separating passenger groups, which we of course exploited. With two or more decks available, passengers can be separated by transporting them on different decks of the *same* car.

## 2.2 Offline Problem: A Case for AI Planning

The offline optimization problem for one elevator is given by a particular traffic situation in a building. This includes the traveling direction and current location of the elevator, the unanswered destination calls of passengers waiting in the building, and the destination calls that have already been serviced and whose passengers are already traveling in the elevator towards their destination. A state in the search space represents the traffic situation at a particular moment in time. The search space contains all possible traffic situations reachable from the initial traffic situation by moves of the elevator from one stop to the next in order to pick up or deliver passengers. A careful choice of data structures to implement the state representation had to be developed to allow a fast update of state changes and to make backtracking less costly.

In practice, one is interested in finding stop sequences that yield a high service level for arbitrary traffic patterns and buildings. Given the one-car problem, it is commonly agreed that one way of achieving this is to compute an optimal travel route for this car serving all passengers currently known to the system. As discussed above, the optimization criterion will usually be a combination of waiting and journey times. This means that, whenever a new call is registered, we try to compute a sequence of stops that serves all “old” calls plus the “new” call and that minimizes, for example, the total waiting time of all passengers or the overall time passengers spend with the elevator system from the moment they place a call until they reach their destination. For the practical solution of the optimization problem, the following observations are important:

- The size of the problem instance is not determined by the number of passengers, but by the number of stops to be added to the solution. For example, ten passengers traveling from floor 1 to floor 10 require only the simple stop sequence  $1 \rightarrow 10$ , assuming the elevator can accommodate ten persons. As good elevator controls must ensure short waiting times for all passengers, the length of “good” solutions is bounded.
- Although the search-space depth is reasonably bounded, the real-time requirements are quite demanding. The optimal solution to the one-car problem will later be used to calculate

instant allocations of passengers to cars. Furthermore, each car itself must be able to quickly recompute its travel route in order to react to the constantly changing traffic in a building. We therefore set an upper limit for the computation time of approximately 100 milliseconds.

Studying the application in more detail, one also finds that computing *suboptimal* stop sequences for a single car can result in very long traveling routes owing to the unnecessary detours it would have to make. Thus, our planner is to avoid generating suboptimal stop sequences.

A fast domain-specific planning algorithm was developed that constructs an *optimal* sequence of stops for a single elevator, e.g.,  $1 \rightarrow 3 \rightarrow 5$ , which serves all passengers registered for this car and obeys all active constraints. In other words, we do not need to verify, for example, that access restrictions are satisfied if all passengers have access to all floors in a particular building. The search algorithm is based on a combination of several search techniques. The core is a depth-first, branch-and-bound search algorithm, which has been augmented with forward-checking techniques adopted from constraint reasoning. Forward-checking allows states that violate service requirements such as *direct travel*, *separation of passenger groups*, *access restrictions* to be pruned from the search space without computing them explicitly, cf. figure 2.

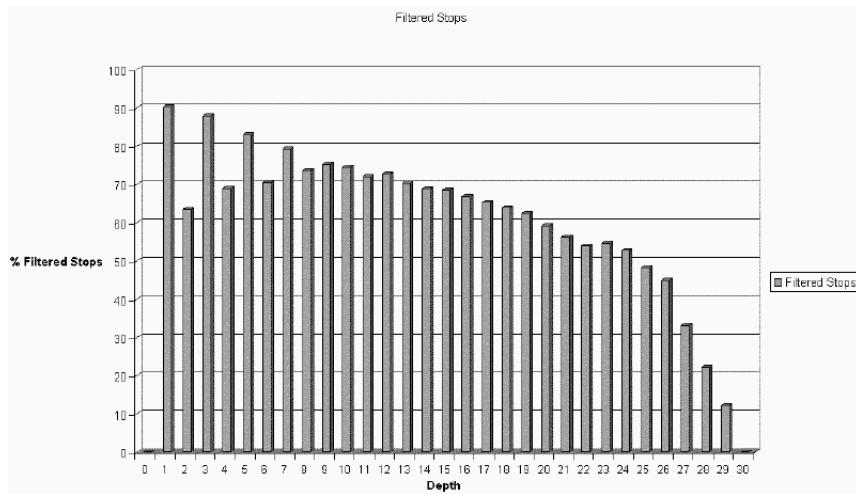


Figure 2: Effectiveness of forward-checking to prune states that violate the elementary constraints of *elevator capacity*, i.e., allocating passengers to overcrowded cars, and *direct travel*, i.e., allocating passengers to cars heading in the opposite direction. Invalid states can be pruned without expanding them, thereby significantly accelerating the algorithm. For example, at depth 14, approx. 68% of the nodes are pruned, which means that 231,732 out of 336,937 nodes were removed from the search space at this depth without being expanded.

The search algorithm is able to compute optimal stop sequences for cars with an arbitrary number of decks. For example, in a double-deck car, a passenger can in principle board on the lower or upper deck. The search algorithm has to consider all possible permutations of passengers and decks, which can double the branching factor of the search space compared to the single-deck case.

Each step added to the plan contributes to its costs. The distance to the goal state in which all passengers have arrived at their destination is measured using a *domain-specific admissible heuristic function*. For example, to minimize the total waiting time of all passengers, the cost function will take the sum of the waiting times of passengers who have been picked up based on the stop sequence constructed so far (the current costs). The corresponding heuristic function will determine the shortest waiting times for passengers still waiting at the various floors (the estimated costs). If the current costs plus the estimated costs exceed the costs of the best solution generated

so far, the entire branch can be pruned from the search tree.

We designed an admissible heuristic function, which allows as many nodes as possible to be pruned from the search space without affecting its completeness. It works independently of the architecture of a building (e.g., number of floors, floor height) and of the elevator characteristics (e.g., speed, acceleration). As a result, the branching factor is significantly reduced. Very often, only one-third of the branches remain for inspection. A typical scenario often found in these search spaces is shown in figure 3. This reduction is extremely important for accelerating the algorithm because typical search spaces can have an average branching factor of about ten successors for each node.

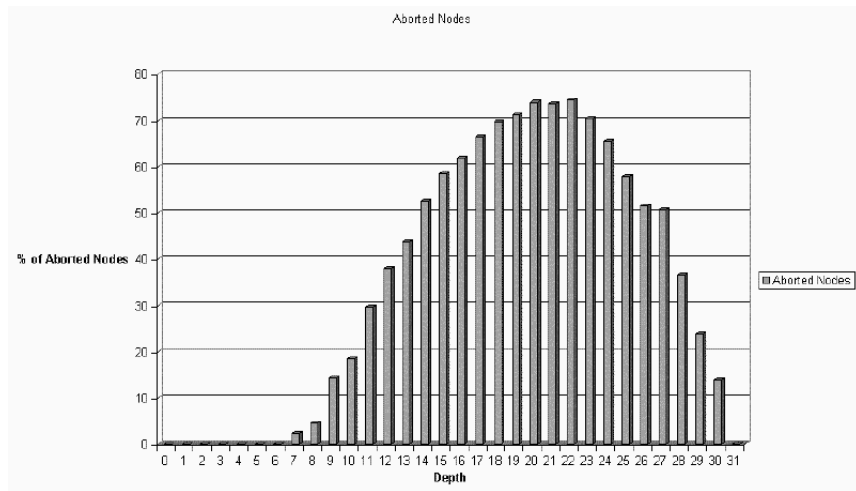


Figure 3: Effectiveness of the heuristic function illustrated by the percentage of nodes that are cut at each search depth when generating a plan of length 31. The heuristic becomes increasingly effective with increasing search depth, reaching its peak at depths 19 to 22, where it can prune more than 70% of the nodes, which corresponds to 35,917 out of 50,446 nodes at depth 19, for example.

Currently, the planning system searches more than 200,000 states per second (implemented in Delphi 5 and running on a 500 MHz IntelliStation under Windows NT). This allows optimal plans of up to a length of 15 to 25 stops to be found in less than 100 milliseconds. Data from real buildings show search spaces containing between  $10^{12}$  and  $10^{15}$  states in peak traffic situations. With these runtime properties, the algorithm scales to very high dimensions of traffic and building sizes, i.e., even the largest elevator system has long exceeded its available transportation capacity before the algorithm runs into combinatorial explosion. The execution of the plans requires the stop sequences to be translated into the much more fine-grained level of elevator control commands, which is described in the following section.

### 2.3 Online Problem: Interleaved Planning and Execution

Thus far we have been able to compute the optimal stop sequence for a single elevator to serve a given set of registered calls. In a real building, a set of destination calls has to be served by a group of elevators, making a reasonable sharing of calls amongst cars desirable. In the dispatching scenario for conventional elevator systems, pickup calls are redispached amongst cars all the time. In a destination-control system, the situation is quite different: First, remember that there are no longer any buttons available inside the cars, i.e., once passengers have boarded the car they cannot influence the travel route of this car. Thus, the travel route is solely determined by the destination calls allocated to this car. It is therefore fully predictable by the control software because the main

reason for redispaching conventional elevators—namely the uncertainty over which cabin buttons a passenger might press—has been eliminated. Second, remember that passengers withdraw from the terminal after they have received their allocation. This means that, once passengers have been allocated to a particular car, this allocation cannot be changed to another car. Although it might sometimes be desirable to reallocate passengers, it is impossible to communicate a new allocation to them. Reallocating passengers while they are waiting would also add a significant amount of confusion for many individuals. Thus, each car serves a subset of the destination calls, which is fixed—only the order in which the calls are served by each car can be changed. It follows immediately that we cannot compute the globally optimal allocation of calls to cars and the resulting minimal travel routes unless we are able to reallocate passengers.

Thus, when a new call is received by a terminal, the best the controller can do is to send this call to each elevator and request a revised, optimal stop sequence accommodating this new call. The revised plans are compared, for example based on the time a car could pick up the new call, and the passenger is allocated to the car that has submitted the best-ranked plan. This process is known as an *auction*, which is commonly used in multi-agent communication. The auction models a greedy search at the global level.

Consequently, we embedded our planning system in a *multi-agent system* implementing the control software on each single car. The agents communicate via *asynchronous messaging* supporting *publish/subscribe* mechanisms and allowing *peer-to-peer* communication between lift components such as drives, doors, and terminals. New agents can dynamically register with the communication network, which also informs other interested agents about the presence of a new agent, i.e., the communication layer supports *ad hoc* networking. Agents representing physical components or logical functions of the control can either send messages directly to each other, in which case they know the recipient of the message, or they can publish information, in which case they do not need to be aware of the subscribers.

### 2.3.1 Auctions Allocate Passengers

For each elevator, a so called *job manager* implements its controller. The job manager is a *holon of agents* responsible for various tasks in the control. A holonic agent system [5] comprises a group of cooperating agents, which appear as a single agent when communicating with other agents. One of the agents from the group will act as the “head” of the holon and represent the group in contacts with external agents. In our model, this central role is played by the brokering agent, which handles the communication between terminals and elevators.

Passenger calls are received via terminals spread throughout the building. Each terminal receives information about the building configuration from the so called *configuration manager*, which also maintains a database of passenger profiles if the individual identification of passengers is available. Thus, upon receiving a call, the terminal checks which elevators can serve the entry and destination floor of this passenger and verifies whether the passenger has access to the desired destination. If the verification was successful, the terminal requests an offer from all available elevators by contacting the corresponding brokers. The broker initiates a planning process and calculates an offer based on the revised optimal stop sequence. This offer is sent to the terminal, which selects the elevator with the best offer and sends an order to it. Upon receiving the order, the broker checks whether its offer is still valid, because the traffic situation may have changed during the elapsed time, and if so, it confirms the order. The passenger is now allocated to this elevator, and the terminal displays the car allocation. Figure 4 illustrates the two-levelled *contract net protocol* [35] underlying the communication process.

The individual brokering of passenger calls also makes the system more tolerant of failures. Passengers can be allocated as long as at least one broker remains active, which makes our design similar to the *adaptive agent architecture* [18]. Brokers can take over allocations that would have otherwise been assigned to another elevator, but they cannot actively bring up new brokers, i.e., new elevators, which is a natural limitation in this application.

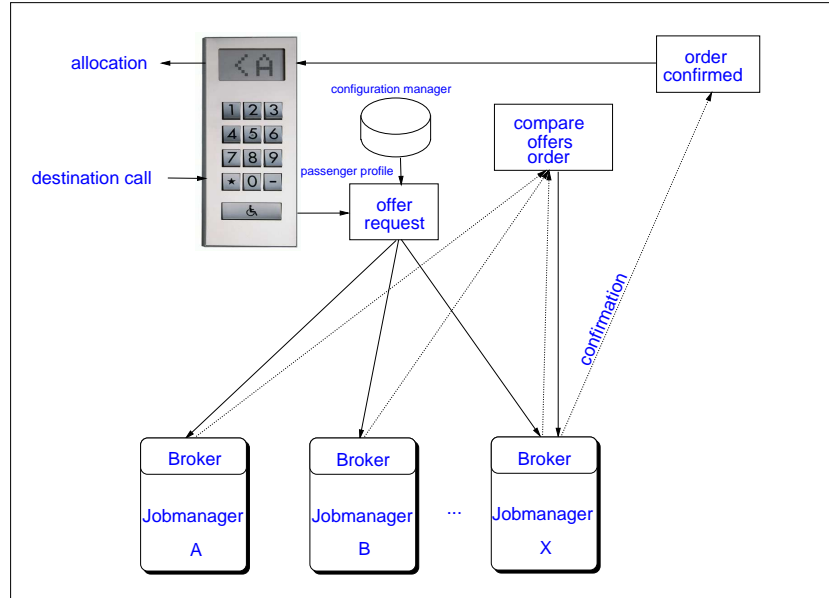


Figure 4: Allocation of passengers to elevators via an auction model. The underlying contract net protocol has been developed in cooperation with the DFKI Saarbrücken [6].

### 2.3.2 Job Manager as a Holon of Specialized Agents

The job manager integrates all components for the logical control of an elevator and communicates with the physical elements such as the doors and drive, see figure 5. The holon of agents forms a persistent team that continues to exist even when team members change or all goals have been achieved, i.e., when there is no traffic and all elevators are idle. Similar to the *open agent architecture* (OAA) [20], the behavior of agents is triggered by events that are transmitted as messages between the agents, but whereas in OAA we need facilitator agents to mediate between agents representing distributed applications, our agents communicate directly with each other using the peer-to-peer communication layer.

In the following, we will discuss the role of each of these agents in the job manager system. Note that there is one job manager per car and that they are completely independent of each other. There is no communication between the job managers or coordination of activities amongst them. The group control results from the allocation of passengers to the best-bidding car in the auction.

The **broker** receives offer requests from the terminals and adds these new calls to the *world model* of the planner, which represents the initial state representation for the planner. The *world model* implements a *blackboard*-like data store, which allows various agents to communicate with each other their knowledge concerning the status of passengers. After having added the new passenger call to the world model, the broker initiates the planning process and evaluates the stop sequence returned by the planner. The broker evaluates how the new passenger affects those passengers already allocated to this car and how long the new passenger has to wait until being picked up.

The **car driver** is responsible for executing the plans. Given an abstract sequence of stops, the car driver maps these into a fine-grained temporal sequence of activities, e.g., accelerating, moving, landing, opening doors. It calculates the door opening times depending on the information about the number of boarding and alighting passengers that it can expect at a given stop and sends appropriate commands to the doors and drive. When the plan has been executed, it releases the elevator such that it can be parked, for example.

Whenever a new passenger has been allocated, the broker tells the car driver to update its trip

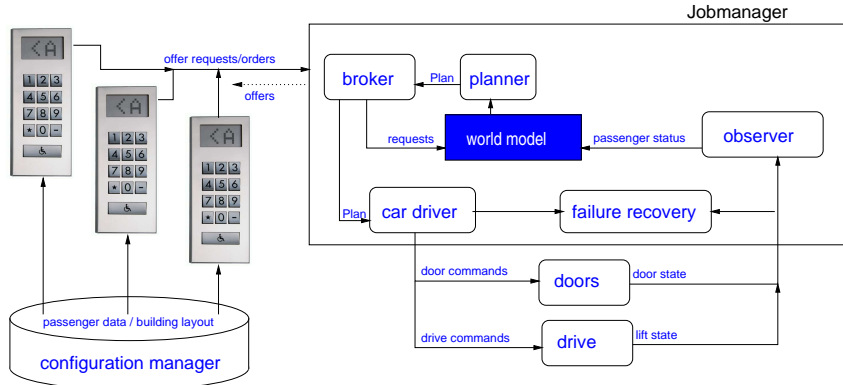


Figure 5: Job manager as a holonic agent system implementing interleaved planning and execution for a single elevator.

plan immediately. In fact this means that the current plan is deleted and replaced by the new plan. Exchanging plans is a tricky issue in this domain because not all actions can be aborted immediately. For example, when the elevator is driving full speed from floor 0 to the next stop at floor 20 and is currently passing floor 5, then it can easily accommodate a stop at floor 10 on the way and plan update is not a problem. The transfer process with passengers boarding and alighting, cannot be aborted, doors closed, and the elevator sent elsewhere. We therefore distinguish between two *modes of execution* for activities: *loose execution* and *enforced execution*. A loosely executed activity can be disrupted immediately, whereas an enforced activity must be finished but can be modified, e.g., doors can be held open longer. Therefore, each car communicates its current activity mode to its associated planning system. This information is part of the initial state representation of the planning problem.

The **observer** continuously updates the world model of the planner according to the information it receives from the doors and the drive. For example, when the drive stops at a certain floor and the doors have been opened, it assumes after a certain delay that all traveling passengers whose destination corresponds to this stop have left the elevator and all waiting passengers have boarded. The observer knows nothing about the car driver—it only considers information available about the doors and drive of the car it is observing. This independent updating of the world model is very important to keep the world model and reality synchronized. For example, if a door does not open at a floor although the open-door command has been sent, the observer will not update the passenger status at this floor and not remove alighting passengers from the world model. When the planner is activated again, these passengers are still present in the car and have to be accommodated when a new plan is generated.

The **failure recovery** agent monitors plan execution, diagnoses problematic situations, and initiates recovery actions. It maps the activities of the car driver to the information it receives from drive and doors, and verifies whether the intended activities have indeed been executed in the physical layer. For example, the car driver sends an open-door command and publishes that it has done so. Now the failure recovery expects the respective door to open and sets *triggers* [20] that allow it to monitor external sensors and track the progress of plan execution. If the door does not open after a certain amount of time, the failure recovery agent becomes active. It follows a three-level approach. At the first level, it simply tells the car driver to replace its current plan, which triggers plan execution from the point where the error occurred. If the failure cannot be corrected this way, it enters the second level and tells the planner to replan for the current situation and the car driver to try to execute this new plan. If this also fails, it enters the third level, assumes control of the elevator and attempts to evacuate passengers.

The failure recovery agent implements a very flexible approach to deal with hardware failures,

with situations that make the world model and reality become drastically unsynchronized, and with passengers who interact with the system in an unforeseeable way, e.g., by blocking doors.

The **drive** executes start and stop commands and records the traveling times of the elevator between the various floors. This information is continuously updated in the planner, which uses it to compute action costs. The drive also publishes *static* information about how many decks it has and *dynamic* information about its current status and position. This becomes part of the initial state representation of the planning system.

The **doors** execute the opening commands that the car driver sends when a desired floor has been reached. Note that we can have several doors over several decks opening to various access zones on a floor. This can make the computation of door opening times and the order in which the doors should open while observing access restrictions quite tricky.

The **configuration manager** provides information about the building layout, i.e., the number of floors, access zones, passenger groups, access rights, and active services.

Each component in the job manager is a self-acting agent that initiates activities when certain events occur. This can trigger several agents simultaneously, whose activities then run in parallel. For example, the broker can receive several requests from various terminals before one of these requests results in an order. In another situation, a terminal may have allocated a passenger to an idle elevator, but when the car driver begins to execute the plan, its commands are ignored because the drive is busy executing parking maneuvers or because another agent, e.g., the cleaning service, has assumed control of the elevator and reserved the cabin. Our control algorithm is able to deal with such interfering events.

### 3 Empirical Results

Simulation runs with artificial traffic patterns or call profiles gathered from real buildings allow us to investigate empirically the dynamics of destination-control problems. The following figures show a *noon peak* traffic pattern (45% up-peak, 45% down-peak, 10% interfloor traffic) in a building with five fast elevators running at 8 m/s. Each car can accommodate up to thirteen persons and serves 25 floors. We compare three different traffic intensities: 90 passengers (low traffic for this building), 180 passengers (medium traffic), and 270 passengers (high traffic) arriving in a five-minute period during a simulation time of four hours. Figure 6 shows the lengths of plans, i.e., the number of stops in the *optimal* stop sequence returned by the planners in less than 100 milliseconds. With a 30% traffic increase, the average length of the plans that each elevator has to generate in order to serve all passengers can easily double. For low traffic, plans have an average length of 2.9 stops, medium traffic requires 6.1 stops, and for high traffic plans with an average length of 9.6 stops have to be generated in order to serve all passengers.

However, with increasing traffic, plans are replaced much more often, and long plans are unlikely to be fully executed. Figure 7 illustrates how many stops in a plan are actually executed before this plan is replaced. In low traffic, the average number of executed stops is 1.6, medium traffic results in 1.0 executed stops, dropping down to 0.8 stops in high traffic. There was no situation in which more than 6 stops of a plan were executed.

This allows us to draw a radical conclusion: To solve the *control problem* we could easily restrict the search function to a depth of 5 or 6 stops and simply generate optimal plan prefixes of this length as was proposed in [8] using *lookahead search*. Such a plan prefix would contain only the initial 5 or 6 stops and would only serve some of the registered calls, but could be extended to the optimal plan serving all passengers. So, why do we invest so much effort into generating plans to serve all passengers? Simply to keep the elevators running, it would be sufficient to determine the next stop they should serve. The answer lies in the *allocation* problem the brokers have to deal with: When the terminals request an offer for a new passenger, each broker has to determine how this passenger would be best served while maintaining the service level for the passengers already allocated to this elevator. Simply inserting the new passenger's stop into the current plan yields a

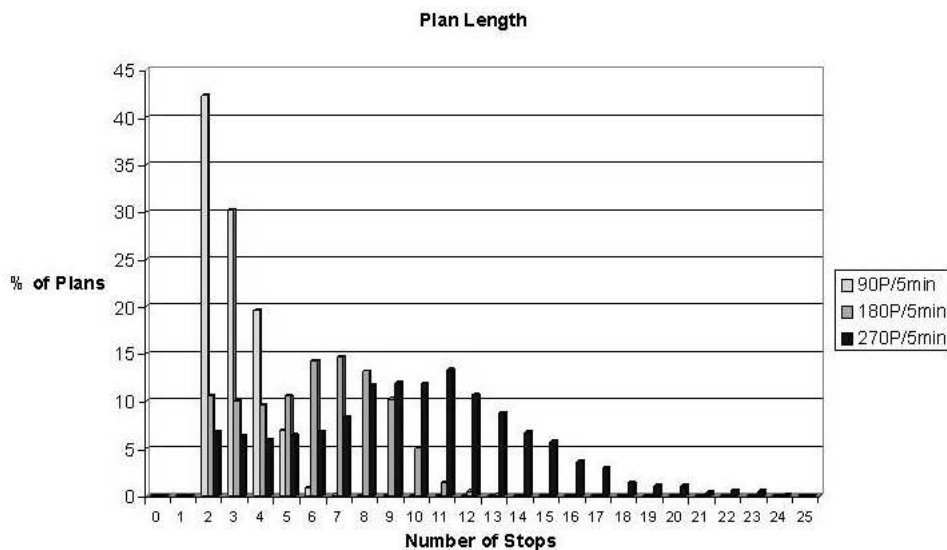


Figure 6: Length of generated plans for low (90 pass./5 min.), medium (180 pass./5 min.), and high (270 pass./5 min.) *noon peak* traffic situations.

satisfactory, but suboptimal solution in many traffic situations. Quite often, however, it turns out that a significant change in the service order of some passengers yields shorter waiting and travel times for most passengers. Therefore, the planner generates a comprehensive plan showing the impact of the new passenger on *all* other passengers allocated to this car. A plan prefix of length  $k$  would only show the impact of the new passenger on those passengers served by the first  $k$  stops. Thus, comprehensive plans are necessary in order to perform an in-depth analysis of the changed traffic situation of an elevator. Moreover, they allow the broker to make more informed decisions. A typical situation is an empty elevator that serves passengers traveling in both directions. It makes a significant difference to waiting passengers who want to travel upwards if the elevator starts to travel downwards first, and vice versa. In particular, the effect on *waiting times* requires to determine optimal solutions for each individual car.

We also compared the resulting control with the original Miconic-10<sup>TM</sup> algorithm and a conventional algorithm developed at Schindler. The following results were obtained across the board: Independent of the traffic pattern, the average waiting times are reduced by 10% when comparing our algorithm to the Miconic-10<sup>TM</sup> algorithm. This improvement is not perceptible to a passenger because it corresponds to an absolute reduction of one or two seconds. Moreover, the reduction of the average journey times depends on the traffic pattern. For up-peak, a reduction of only 10% is achieved, whereas for down-peak and noon-peak we observe improvements of around 35%. In larger buildings, this means that approximately 30 seconds per passenger trip is saved on average. This corresponds to one intermediate stop during a passenger's journey being eliminated by the controller. Maximum waiting times are not reduced, but maximum journey times are reduced up to 50%, a significant improvement.

This improvement is achieved by virtue of the algorithm's ability to completely replan the stop sequence of an elevator. For example, in low down-peak traffic, the optimal solution might be to collect passengers when traveling downwards with a stop sequence of 10-8-6-0. If more and more passengers register calls at 10, 8, or 6 such that each stop will entirely fill the car, the planner can replace it by the new optimal sequence 10-0-8-0-6-0, thus picking up all waiting passengers at one stop, shuttling them directly to the lobby, and then returning to the next floor in the plan. Such a flexible reordering of stops to accommodate changes in traffic is not possible with other control

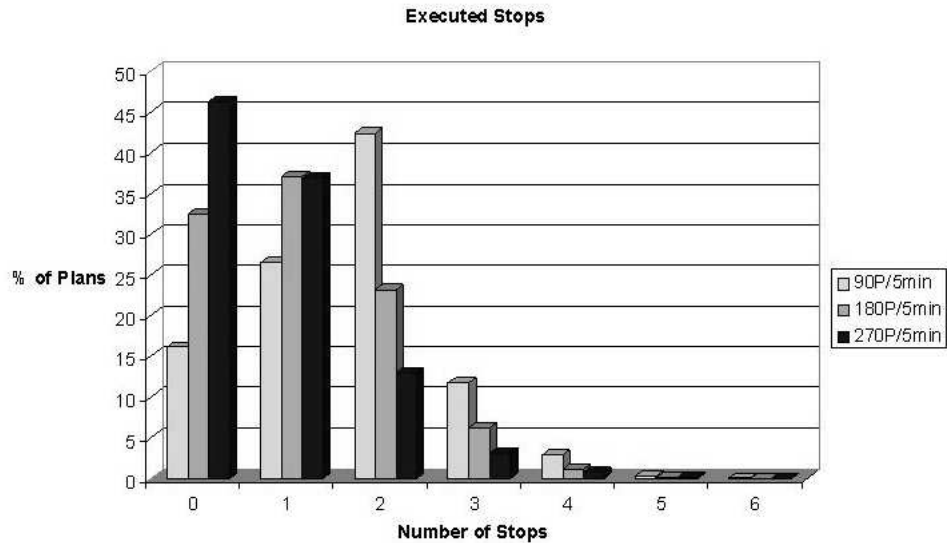


Figure 7: Percentage of plans with 0, 1, 2, . . . executed stops in low (90 pass./5 min.), medium (180 pass./5 min.), and high (270 pass./5 min.) *noon peak* traffic situations.

algorithms. Conventional controllers struggle with the problem of avoiding stops at every floor during heavy down-peak traffic.

By replacing a group of single-deck cars with the same number of double-deck cars, the transportation capacity during up-peak traffic can be doubled. The financial benefits of the new control algorithm come from reducing the number of cars needed to achieve a required service level. For example, instead of installing five cars with conventional control, three cars with destination control can handle the same amount of traffic. Or the other way round, with the same number of cars, up to 30% more passengers can be transported—thus reducing the costs of constructing, installing, and maintaining an elevator system. A further benefit comes from saving the time of passengers, which can easily cumulate to several hundred hours per month in a large office building.

Much more important than improving performance is the ability of the system to offer custom-tailored services that can be incorporated seamlessly into the standard control. Access control, VIP service, and separation of passenger groups are achieved without taking an elevator temporarily out of service, which keeps the performance of an elevator group high. For example, instead of having different elevators serving different zones in a building with low traffic, simply one group can suffice if the control temporarily grants service to the various zones by planning the travel routes for the elevators such that zones can only be accessed by authorized passengers.

The event-triggered activation of agents also makes the control open to the addition of future services as well as being much more robust against failures. For example, if a shaft door at a certain floor is blocked, the elevator can still serve the remaining floors as long as the safety of the passengers is guaranteed. Similarly, if a cabin door at the second deck of a double-deck cabin malfunctions, the job manager could still safely operate this elevator as a single-deck cabin. This implements a useful form of *graceful degradation* that improves the availability of elevator systems.

## 4 Summary and Discussion of Open Issues

As we have discussed in this article, elevator control is a major field of application for AI technologies. Starting with expert systems in the late 1980s, elevator companies have explored a large variety of AI developments such as neural networks, genetic algorithms, fuzzy rules, and—

recently—multi-agent systems and AI planning in their search for intelligent elevator controls. AI technologies stand not only to increase the transportation capacity of conventional elevator systems, they can also improve the way in which elevators interact with and serve passengers. The first part of this paper reviewed past approaches to improve conventional elevator systems.

In the second part, we presented a novel elevator system based on so called destination control, where passengers specify their desired destination at terminals located outside the elevator. An auction process executes the allocation task, where each car bids for the passenger requesting transportation. In order to process a bid, a planning system computes an optimal stop sequence serving a set of destination calls. The planner uses a hybrid search algorithm combining depth-first branch-and-bound with constraint-propagation techniques. This search algorithm is embedded in a multi-agent architecture, which implements an interleaved process of plan generation and plan execution. The result is robust and fault-tolerant control software, which improves elevator performance significantly and offers novel customer-tailored services.

We can identify two major areas where AI technologies could become even more important in the future. First, communication between passengers and elevators could be improved significantly. Although simple buttons suffice to implement vertical transport, new services and new elevator systems require new channels of communication. For example, imagine a large building with a sky lobby on the 50th floor. A passenger who wants to travel to floor 49 could take the express car to the sky lobby and change there to a downward-traveling elevator. Our destination control system could easily compute this solution by having the next generation of job managers cooperate to determine the fastest routes for passengers. But how can a more complicated travel plan of this kind be communicated to passengers in a simple manner? Another example is that of disabled passengers or persons unfamiliar with the use of destination-control terminals. How could the terminal detect a confused passenger and provide assistance for correct usage? Today, for example, blind passengers are directed to their allocated car by a unique tone played by the terminal and the arriving elevator. Are there better communication methods? Although elevators are highly interactive devices, they often fail in truly interacting with people in a responsive way. The challenge lies in finding cheap, easy-to-use, intelligent interfaces that facilitate new forms of communication between elevators and humans. AI research to develop solutions for multimodal communication in various settings could help address these issues.

Second, current elevator systems constitute a significant waste of space. Typically, a huge shaft is used by a single car. Initial attempts have been made to put several cars into one shaft or to extend the disconnected vertical shafts to a system of connected horizontal and vertical transportation channels occupied by several autonomous vehicles. The design and control of such *multimobile systems* require issues to be addressed that are currently being studied in such subfields of AI as robotics, planning, multi-agent systems, and reasoning under uncertainty.

## References

- [1] A. Alani et al. Performance optimisation of knowledge-based elevator group supervisory control system. In G. Barney, editor, *Elevator Technology*, volume 6, pages 114–121. IAEE, 1995.
- [2] M. Amano et al. The latest elevator group supervisory control system. In G. Barney, editor, *Elevator Technology*, volume 6, pages 88–95. IAEE, 1995.
- [3] F. Bacchus. The AIPS'00 planning competition. *AI Magazine*, 22(3):47–56, 2001.
- [4] G. Barney. *Elevator Traffic - Analysis, Design and Control*. Peter Peregrinus Ltd., 1977.
- [5] H.-J. Buerckert, K. Fischer, and G. Vierke. Holonic transport scheduling with teletruck. *Applied Artificial Intelligence*, 14(7):697–525, 2000.

- [6] H.-J. Bürckert and G. Vierke. Agent models for elevator control. internal report, unpublished, 2001.
- [7] W. Chan and A. So. Dynamic zoning in elevator control. *Elevator World*, XLV(3):136–139, 1997.
- [8] P. Chenais. Method and apparatus for assigning calls entered at floors to cars of a group of elevators. Schindler US Patent 5,612,519, 1997.
- [9] G. D. Closs. *The Computer Control of Passenger Traffic in Large Lift Systems*. Phdthesis, University of Manchester Institute of Science and Technology, 1970.
- [10] R. Crites and A. Barto. Improving elevator performance using reinforcement learning. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 8, pages 1017–1023. MIT Press, 1996.
- [11] J. Forcht. Rufeingabevorrichtung für Aufzugsanlage (call input device for elevator installation). Thyssen European Patent 1,006,070, 1998.
- [12] P. Friedli. Group control for lifts with immediate allocation of destination calls. Schindler European Patent 0356731B1, 1989.
- [13] K. Hattori et al. Group-controlled elevator system. OTIS European Patent EP 0,810,176, 1997.
- [14] M. Helmert. On the complexity of planning in transportation domains. In *Proceedings of the 6th European Conference on Planning*, LNAI. Springer, 2001.
- [15] J. Hoffmann. Local search topology in planning benchmarks: an empirical analysis. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 453–458. Morgan Kaufmann, San Francisco, CA, 2001.
- [16] J. Koehler and K. Schuster. Elevator control as a planning problem. In S. Chien, S. Kambhampati, and C. Knoblock, editors, *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, pages 331–338. AAAI Press, Menlo Park, 2000.
- [17] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42:189–211, 1990.
- [18] S. Kumar, P. Cohen, and H. Levesque. The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the 4th International Conference on Autonomous Agents*, pages 459–466. ACM Press, 2000.
- [19] D. Levy, M. Yardin, and A. Alexandrowitz. Optimal control of elevators. *International Journal of Systems Science*, 8(3):301–320, 1977.
- [20] D. Martin, A. Cheyer, and D. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1):91–128, 1999.
- [21] D. McDermott. The 1998 ai planning systems competition. *AI Magazine*, 21(2):35–55, 2000.
- [22] A. Miravete and N. Tolosana. Genetics and intense vertical traffic. *Elevator World*, XLVII(7):118–120, 1999.
- [23] M. Nakamura, K. Yoneda, and A. Togawa. Elevator control system with modified display when operating mode changes. Hitachi UK Patent GB 2,311,148, 1997.
- [24] R. Peters et al. Lift passenger traffic patterns: Applications, current knowledge and measurement. *Elevator World*, IIL(9):87–94, 2000.

- [25] B. Powell, D. Sirag, and B. Whitehall. Artificial neural networks in elevator dispatching. *Lift Report*, 27(2):14–19, 2001.
- [26] Z. Qun et al. A direct-search approach to dynamic zoning optimization-problem of elevator group control systems. *Elevator World*, IIL(2):136–140, 2001.
- [27] Z. Qun, X. Hua, and Y. Jun. A multi-objective dispatching method in elevator group control systems. *Elevator World*, XLVIII(12):164–168, 2000.
- [28] A. Schofield, T. Stonham, and P. Mehta. A machine vision system for counting people. In A. Lustig, editor, *New Methods and Technologies in Planning and Construction of Intelligent Buildings*, pages 50–59. IB/IC Intelligent Building Congress, Israel, 1995.
- [29] B. Seckinger. Synthesis of elevator controls based on constraint-based search. Master’s thesis, Albert-Ludwigs-Universität Freiburg, 1999. in German.
- [30] B. Seckinger and J. Koehler. Online synthesis of elevator controls as a planning problem. In *13th Workshop on Planning and Configuration, Technical Report, University of Würzburg*, pages 127–134, 1999.
- [31] M. Siikonen. Elevator group control with artificial intelligence. Technical Report A67, Helsinki University of Technology, 1997.
- [32] M. Siikonen. Procedure for control of an elevator group consisting of double deck elevators, which optimizes passenger journey time. Kone PCT Patent WO 98/32683, 1998.
- [33] M. Siikonen. On traffic planning methodology. *Lift Report*, 27(3):24–29, 2001.
- [34] M. Siikonen and T. Korhonen. Defining the traffic mode of an elevator, based on traffic statistical data and traffic type definitions. Kone US Patent 5,229,559, 1993.
- [35] G. Smith. The contract-net protocol: High-level communication and control in a distributed system. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.
- [36] A. So et al. A comprehensive solution to computer vision-based group supervisory control. In G. Barney, editor, *Elevator Technology*, volume 5. IAEE, 1993.
- [37] A. So et al. Elevator traffic pattern recognition by artificial neural network. In G. Barney, editor, *Elevator Technology*, volume 6, pages 122–131. IAEE, 1995.
- [38] A. So and S. Liu. An overall review of advanced elevator technologies. *Elevator World*, XLIV(6):96–103, 1996.
- [39] G. Strakosch. *The Vertical Transportation Handbook*. Jon Wiley and Sons, 1998.
- [40] H. Ujihara et al. Application of expert systems to elevator group control. *Lift Report*, 15(6):46–48, 1989.
- [41] Y. Umeda et al. Fuzzy theory and intelligent options. *Elevator World*, XXXVII(7):86–91, 1989.
- [42] B. Whitehall and B. Powell. Adjustment of elevator response time for horizon effect, including the use of a simple neural network. OTIS US Patent 5,936,212, 1999.
- [43] B. Whitehall, D. Sirag, and B. Powell. Elevator control neural network. OTIS US Patent 5,672,853, 1997.
- [44] K. Yoneda et al. Multi-objective elevator supervisory-control system with individual floor-situation control. *Hitachi Review*, 46(6):266–274, 1997.