Real-Time Destination-Call Elevator Group Control on Embedded Microcontrollers

Benjamin Hiller*
and Andreas Tuchscherer**

Zuse Institute Berlin, Takustraße 7, D–14195 Berlin, Germany
{hiller,tuchscherer}@zib.de

Summary. We introduce new elevator group control algorithms that can be implemented to be real-time compliant on embedded microcontrollers. The algorithms operate a group of elevators in a destination call system, i. e., passengers specify the destination floor instead of the travel direction only. The aim is to achieve small waiting and travel times for the passengers. We provide evidence, using simulation, that the algorithms offer good performance. One of our algorithms has been implemented by our industry partner and is used in real-world systems.

1 Introduction

Algorithmic control of elevator systems has been studied for a long time. A suitable control should achieve small average and maximal waiting and travel times for the passengers. The waiting time and the travel time of a passenger is the time span between the release of the call and the arrival of the serving elevator at the start floor and destination floor, respectively.

Recently, the paradigm of destination call elevator control has emerged. In destination call systems, a passenger enters the destination floor (and possibly the number of passengers traveling to this floor). Such a destination call system is very interesting from an optimization point of view, since more information is available earlier, which should allow improved planning.

In this paper we report on elevator control algorithms designed for Kollmorgen Steuerungstechnik, our partner from industry. The algorithm designed is supposed to run on embedded microcontrollers with computation times of at most 200 ms using not more than 200 kB of memory. Thus computational resources are very scarce.

* Supported by the DFG research group "Algorithms, Structure, Randomness" (Grant number GR 883/10-3, GR 883/10-4).

** Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

Related Work Many elevator control algorithms have been proposed, but only few of them seem to be used in practical systems. Moreover, there is not much literature on destination call systems yet. Tanaka et al. [4] propose a Branch&Bound algorithm for controlling a single elevator, which uses too much computation time to be implemented on an embedded microcontroller. Friese and Rambau [3] developed an algorithm for cargo elevator group control with capacity one based on Integer Programming. Although the algorithm runs in real-time on a PC, it is still too time-consuming for embedded microcontrollers. The book of Barney [1] deals mainly with engineering aspects.

Contribution We introduce new destination call control algorithms suited to run on an embedded system offering very scarce computing resources. Since exact optimization is not feasible on such hardware, the algorithmic approach is an insertion heuristic using a non-trivial data structure to maintain an elevator tour. We assess the performance of our algorithms by simulation. We also compare to algorithms for a conventional system and a more idealized destination call system. This gives an indication of the relative potentials of these systems.

2 Modeling the Destination Call System

The real-world destination call system envisioned by Kollmorgen works as follows. Upon arrival, a passenger enters his destination floor (issues a *call*) and is immediately assigned to one of the elevators of the group. The passenger is supposed to go to the corresponding elevator and board it as soon as it arrives and indicates the correct leaving direction. If the designated elevator arrives and the cabin is full so that a passenger cannot enter, he is supposed to reissue his call.

The anticipated operations of the elevator group can be described by tours for each elevator, specifying the order floors are visited. These tours are used to predict the waiting and traveling times of the passengers, thus allowing to evaluate different control alternatives.

The tours have to fulfill some requirements modeling the real system. (a) Tours need to respect the assignments fixed so far. This requirement differs from the assumptions of Tanaka et al. since there the assignment is done on arrival of an elevator at a floor. (b) A tour must not contain a turn for a passenger, i. e., a passenger must never move in the wrong direction. (c) For each floor and leaving direction, we assume that *all* passengers assigned to an elevator enter the cabin at its first stop at this floor with the corresponding leaving direction. The rationale for this rule is the following. The elevator control has no way to detect which passengers enter at a certain floor (there are no panels in the cabin). Therefore it does not know which stops are really required by the loaded passengers and the elevator has to stop at all registered destination floors. In fact, this is equivalent to assuming that all waiting passengers enter the elevator and thus the capacity of the elevator is ignored for the planning.

We make some other reasonable assumptions as discussed by Tanaka et al., e. g., that if no passenger has to be served by an elevator, the elevator stays at its last floor and the cabin cannot stop or reverse direction halfway between floors.

Note that due to requirement (c) there may be *phantom stops*, i. e., stops for dropping a passenger who is not really in the cabin. Phantom stops and the immediate assignment are features which might alleviate the advantages of a destination call system since both restrict the optimization potential.

3 Algorithms

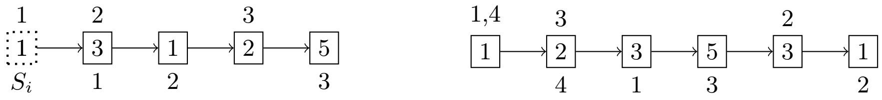
The results of Tanaka et al. [4] and Friese and Rambau [3] suggest that it pays off to use thorough optimization for computing new schedules.

However, the scarce computing resources available on embedded micro-controllers make exact optimization methods infeasible. We therefore propose insertion heuristics. Insertion heuristics are well-known for the Traveling Salesman Problem. However, the structure of a tour for an elevator is much more complex, making the insertion operation particularly non-trivial.

A tour T is a list of stops $T = (S_0, \ldots, S_k)$, where each stop is described by its halting floor, its scheduled arrival and leaving times, and the sets of calls picked up and dropped at this floor. Moreover, we also store the set of currently loaded calls (after dropping and picking up) at each stop. A new call c can be inserted into an existing tour T via the operation $\textsc{AddCall}(T, S_i, c)$, where S_i indicates the insertion position. If the floor of S_i does not match the start floor of call c, a new stop is created before S_i. The insertion position for the drop stop is uniquely determined by S_i, the remainder of T, and the no-turn requirement. It may be necessary to split an existing stop into two new stops to avoid direction changes for passengers. Of course, not every choice of S_i is feasible for insertion but it has to be ensured that a feasible tour is obtained afterwards.

AddCall is non-trivial due to the cases that may arise. For instance, consider the tour in Figure 1(a) and suppose we want to insert the new call $4\colon 1 \to 2$ at the first stop at floor 1 (which is the only feasible insertion position). We need to create a new stop at floor 2, leaving upwards due to call 1. But then call 3 will enter and we need to go to floor 5 before we can leave floor 3 downwards. Therefore we need to adjust the tour, keeping it close to the original one. There are more complex cases of this *repair* operation to take into account.

We use the insertion procedure AddCall to set up a group elevator algorithm Best Insertion (BI) as follows. Every time a new call enters the system, BI inserts the new call at all feasible positions in the already scheduled tours. The call is assigned to the elevator and insertion position where it causes a



(a) Original tour. (b) Tour after insertion.

Fig. 1. Inserting call $4: 1 \to 2$ at S_i needs repair. A square represents a stop at a floor. Numbers above/below a stop indicate calls picked up/dropped.

minimum cost increase. The cost function captures waiting and travel times for all calls. This way, the algorithm balances stability of the plans for old calls with good service for new calls.

In order to achieve real-time compliance and to avoid deferment of single calls leading to high maximum waiting times, we selected a suitable subset of insertion positions. The algorithm CBI (controlled BI) eventually implemented by our industry partner works like BI, but using just this restricted subset of insertion positions.

4 Evaluation and Computational Results

We now use simulation to evaluate our algorithms and compare them to algorithms for a conventional system and a more idealized destination call system. To measure the quality of a control algorithm we use quantiles. We look at the 50%, 75%, 90%, and 100% quantiles of the waiting and travel times.

Simulation Model and Instances The precise rules of the simulation are as follows. At each stop of an elevator the passengers enter the cabin in first-come-first-served (FCFS) manner. Of course this is only relevant if the cabin capacity does not suffice to pick up all waiting passengers.

We consider a building with an elevator group serving 16 floors. The passenger data used in our experiments came from the software tool *Elevate* [2]. We look at eleven templates defined by Elevate that represent different traffic patterns. These include up traffic (U), down traffic (D), and interfloor traffic (I), as well as combinations of different traffic patterns, e. g., UDi denotes a situation with up and down traffic and a little interfloor traffic (indicated lower case "i"). Instances with changing predominant type of traffic are indicated by a "*". For each template we compute the quantiles over ten samples.

Real Destination Call System For the destination call system described in Section 2, we compared our algorithms BI and CBI to an adapted version of the algorithm employed by Kollmorgen for conventional systems, which is based on a complete round trip of an elevator. The criterion for assigning calls to elevators aims to minimize the required waiting time. Our results (omitted due to space restrictions) show that BI seems to be superior to the straightforward adaption of the control algorithm of Kollmorgen, in particular for the travel times and the higher quantiles of the waiting times.

System Comparison For comparison, we also studied two different elevator control systems. In the conventional system we have no information about the destination floor of a call until the passenger has entered the cabin. On the other hand, it is not necessary to assign each call to an elevator immediately. Passengers again enter a cabin in FCFS order. For this setting we implemented an algorithm called CGC [1] designed to perform well in most traffic situations.

Moreover, we consider an idealized destination call system. Here we have complete boarding control, i. e., at each stop of an elevator the control algorithm determines which passengers are picked up. Consequently, the capacity of the elevator cabin is taken into account in this model. We compared BI to another adapted variant of the Kollmorgen algorithm. Similar to the real destination call system, BI performs better in most situations.

Finally, we compare the best algorithms of the different systems with each other. These are CGC (conventional system), CBI and BI-FCFS (real destination call system), and BI-planned (idealized destination call system).

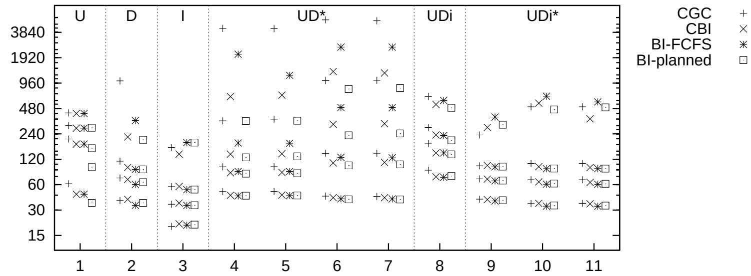
Figures 2(a) and 2(b) show the quantiles of the waiting times when operating three and four elevators. Using three elevators BI-planned gives the best results in most situations. For four elevators (less load per elevator) the situation looks different. BI-planned does not perform predominantly anymore. CGC now performs best for all I and UDi* instances.

The results for travel times are given in Figures 2(c) and 2(d). BI-planned outperforms the other algorithms with three elevators, while CGC achieves the worst travel time quantiles. CBI performs similarly to BI-FCFS, but CBI always achieves a smaller maximal travel time. Using four elevators, CBI yields similar results as BI-planned and gives an even better maximal travel time on most instances. CGC performs quite well on the I and UDi* instances.

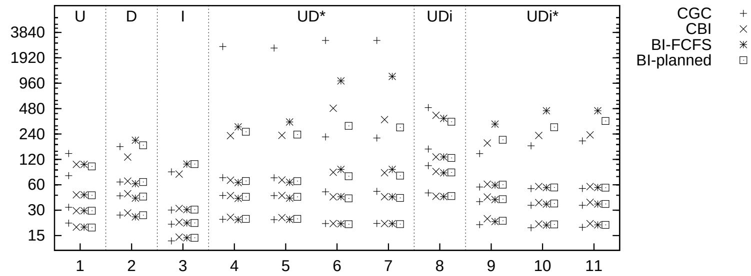
We summarize the most important results. CBI is almost as good as BI-FCFS and even better for maximal waiting and travel times. Destination call systems seem to be superior to conventional systems in high load situations, the opposite seems to hold for low load at least for the waiting times. Moreover, control about the passengers entering the cabin at a stop pays off for destination call systems.

References

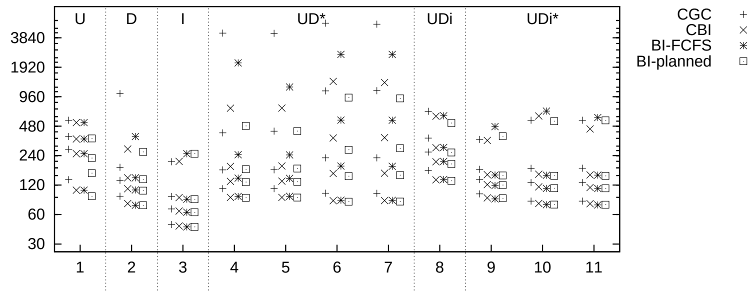
1. G. C. Barney. *Elevator Traffic Handbook: Theory and Practice.* Taylor and Francis, 2002.
2. Elevate – dynamic elevator simulation. Version 6.01. Peters Research, 2004. `www.peters-research.com`.
3. P. Friese and J. Rambau. Online-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models. *Discrete Appl. Math.*, 154(13):1908–1931, 2006. Also available as ZIB Report 05-03.
4. S. Tanaka, Y. Uraguchi, and M. Araki. Dynamic optimization of the operation of single-car elevator systems with destination hall call registration: Parts I and II. *European J. Oper. Res.*, 167(2):550–587, 2005.



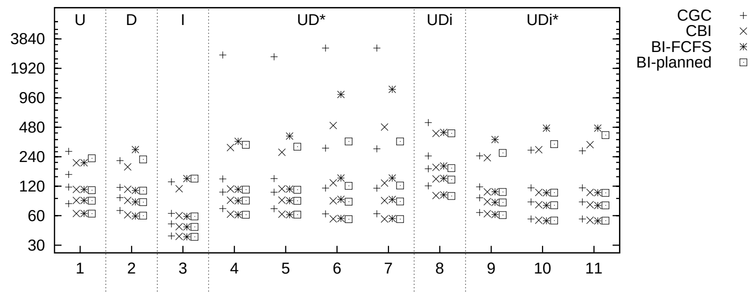
(a) waiting time, 3 elevators



(b) waiting time, 4 elevators



(c) travel time, 3 elevators



(d) travel time, 4 elevators

Fig. 2. Simulation results. Shown are the quantiles of waiting and travel times (in seconds) achieved for 3 and 4 elevators on each of the 11 call templates.