European Journal of Operational Research 252 (2016) 397–406

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Assignment formulation for the Elevator Dispatching Problem with destination control and its performance analysis

Mirko Ruokokoski[a,*], Janne Sorsa[a], Marja-Liisa Siikonen[a], Harri Ehtamo[b]

[a] KONE Corporation, Finland
[b] Aalto University, School of Science, Systems Analysis Laboratory, Finland

ARTICLE INFO

Article history:
Received 5 May 2013
Accepted 12 January 2016
Available online 18 January 2016

Keywords:
Linear programming
Elevator dispatching problem
Assignment formulation
Destination control
Collective control

ABSTRACT

This paper studies the Elevator Dispatching Problem arising in destination controls. In many of the presented methods to the problem a routing aspect is not considered; decision variables specify only request-to-elevator assignments and the service order of the requests is determined by applying a heuristic rule called the collective control principle. However, quality of this approach is rarely investigated. In this paper the approach is compared with a formulation defining explicitly the elevator routes. The average waiting time as well as average journey time are used as objective functions in the comparisons. Computational experiments with the former objective function on random instances arising during light and normal traffic conditions indicate that both approaches very often produce the same solution while with the latter one the situation is the opposite. Some well-known traffic patterns are also analyzed to identify cases in which the optimal solutions of these approaches are equal.

1. Introduction

Elevators in a building form a vertical transportation system designed to transport passengers from their origin floors to destinations safely, comfortably, and efficiently. A set of elevators is called an *elevator group* if they have a common set of request registration devices located in their vicinity on floors and their movements are planned in a coordinated manner. The task of designing elevator routes serving all transportation requests given by passengers is called the *Elevator Dispatching Problem* (EDP). The EDP is by nature an *online problem* as requests are gradually revealed in the course of time and dispatching decisions has to be made on the basis of incomplete data. A rolling horizon scheme is used: an *offline subproblem*, derived from the online version, is solved repeatedly. The task of the offline subproblem is to redesign the elevator routes using all currently known requests subject to all operational constraints.

This paper is concerned with the static offline subproblem arising in the so-called *destination control* (DC). Static means that all inputs of the problem are constants. In a DC registration devices are equipped with destination keypads. It is assumed that each passenger issues a *destination request* by inputting the intended destination floor (and possibly the number of passengers traveling along with him/her to the same destination floor) on a keypad in the elevator lobby. A destination request therefore consists of, in addition to the origin floor and the arrival time, the number of passengers and the destination floor.

A typical policy in a DC is *immediate assignment*. When a passenger registers a request, the offline subproblem is solved, the resulting elevator assignment is immediately announced to him/her on the display, and that request-to-elevator assignment remains fixed in the subsequent offline subproblems (see e.,g., Schröder, 1990 and Sorsa, Hakonen, and Siikonen, 2006). The immediate assignment policy means that the offline subproblem is solved every time a new request is registered. The subproblem can also be solved for each elevator separately in other times to just update the service order of its requests. The immediate assignment policy has also been applied in elevator systems with traditional up and down buttons (e.g., Hirasawa, Kuzunuki, Iwasaki, & Kaneko, 1978).

Another assignment policy used in elevator group controls is *delayed assignment*. In the delayed assignment subproblems are solved constantly, for instance twice a second, and a request-to-elevator assignment is allowed to be changed. The last moment after which the assignment cannot be changed anymore is in practice when the serving elevator has to start to decelerate for the request floor. At this moment the passengers are informed about the arriving elevator by a lantern and/or gong. The delayed assignment is mainly applied in elevator systems with up and down buttons

* Corresponding author. Tel.: +358 456783441.
E-mail address: mirko.ruokokoski@aalto.fi, mirko.ruokokoski@gmail.com (M. Ruokokoski).

http://dx.doi.org/10.1016/j.ejor.2016.01.019
0377-2217/

(e.g., Tyni and Ylinen, 2001 and Fernandez, Cortes, Munuzuri, and Guadix, 2014). It can also be applied in a DC with a destination-oriented passenger guidance system. When an elevator arrives at a floor to server passengers, the guidance system indicates the destination floors of the passengers who should board the car (e.g.,Tanaka, Uraguchi, and Araki, 2005 and Hiller, Klug, and Tuchscherer, 2014).

In many of the presented approaches to the EDP, elevators are assumed to obey a heuristic rule called the *collective control* (CC) (Tyni and Ylinen, 2001 and Smith and Peters, 2002). This rule is explained in the next section in detail. An interesting and significant feature of the EDP with CC (EDPCC) is that its offline subproblems can be modeled as an optimization problem with decision variables specifying only the request-to-elevator assignments. An optimal solution of it, however, may not necessarily be optimal for the EDP. The aim of this paper is to study the quality of the EDPCC.

Quality considerations of the EDPCC have been done in three studies with different setups. Closs (1970) considered a single-elevator DC. However, the available computing power at that time did not allow the investigation of more than toy examples. Inamoto, Tamaki, Murao, and Kitamura (2002) studied a multi-elevator DC but with very simplified settings. They also assumed that an elevator can pick up any subset of the waiting passengers. In practice this means that each passenger should be guided individually when to board the car. Tanaka et al. (2005) investigated a single-elevator DC with the destination-oriented passenger guidance system.

This paper compares the EDPCC with EDP in the case of a multi-elevator DC with a typical passenger guidance and with immediate assignment. That is, a request is immediately assigned to an elevator, that assignment is announced to the passenger on the display in the request giving device, and when the serving elevator arrives, all passengers assigned to it are assumed to enter it. In particular, an aim is to identify cases when the optimal solutions of the offline EDP and EDPCC are equal. To achieve these goals, the offline subproblem of the EDPCC is formulated as a mixed integer linear program with decision variables specifying requests-to-elevators assignments. We refer to this formulation as the *assignment formulation* (AF). The formulation is based on destination requests and therefore may not be applied as such in elevator systems with up and down buttons.

For modeling reasons, in the AF each elevator route is restricted to a certain length, a single round trip. The formulation of the EDPCC where all appropriate constraints have been given in an explicit mathematical form is now, according to our knowledge, introduced for the first time. Mathematical formulation of the offline EDP, which is used in the comparisons, is presented in Ruokokoski, Ehtamo, and Pardalos (2015). That formulation is based on arc-flow formulations, and is referred as the *routing formulation* (RF). In the RF the length of an elevator route is not restricted in any way.

This paper is organized as follows. In Section 2, a detailed description of the offline EDPCC is given. Mathematical formulation follows in Section 3. Section 4 analyzes some well-known traffic patterns to identify cases in which the optimal solutions of both formulations are equal. Branch-and-Bound algorithm used to solve the problem instances is described in Section 5. Computational experiments are reported in Section 6 and conclusions follow in Section 7.

2. Problem description

We begin with some definitions.

Let N be the set of destination requests in the offline subproblem at hand, and $|N| = n$. Requests can be divided into three disjoint subsets according to their states: *unassigned requests* N_1, i.e., requests that must be assigned to elevators; *assigned requests* N_2, i.e., requests that have assignments to elevators but their services have not started yet; and *on-board requests* N_3, i.e., requests whose passengers are already inside elevators. A request that belongs either to N_2 or N_3 is called a *fixed request* as its assignment cannot be changed.

In the CC (e.g., Barney, 2003), an elevator stops to serve requests assigned to it in its current travel direction in a floor sequence. Destination locations of the on-board requests cannot be bypassed in any case whereas assigned and unassigned requests can be bypassed only when the elevator is full. When no more requests are ahead in its current direction, the elevator moves to the furthest assigned request in the opposite direction, if any, reverses its travel direction and serves the requests in the new direction.

In the *reversal control*, which is obeyed by the most of the group control methods including the CC, movements in which at least one passenger travels in the opposite direction with respect to his destination floor after boarding the elevator are not allowed. Such movements are forbidden since they are in general considered psychologically undesirable for passengers (Levy, Yadin, & Alexandrovits, 1977). In practice reversal control means that an elevator is allowed to change its travel direction only when it is empty.

A *round trip* of an elevator e is a cyclic route in a building. It starts from the initial position of e and contains first some (or possible none) requests that can be served in the current travel direction of e. It then contains some (or possible none) requests that can be served after reversing the travel direction once, that is, requests in the opposite direction, and finally some (or possible none) requests that require two reversions for e before they can be served.

Waiting time of a passenger is the time from the instant that the passenger registers the request, until the instant the passenger can start boarding the elevator. *Journey time* of a passenger is the time from the instant the passenger registers the request, until the passenger can start exiting the elevator at the destination floor.

The formal description of the offline EDP considered is as follows. Given an elevator group and a set of requests, the objective is to design a minimum-cost set of elevator routes subject to side constraints. The reversal control principle must be fulfilled. For each elevator, the load of it must not exceed the capacity at any point and the route must start at its initial location. All requests must be served. Requests, except fixed ones, can be served by any elevator. A fixed request must be served by the elevator for which it has been assigned to. For each request, the origin location must precede the destination and both locations must be visited by the same elevator within time windows. The following four assumptions, which more or less reflect practical situations in elevator installations, must be satisfied.

Assumption 1. Waiting passengers at a landing cannot board the serving elevator before all on-board passengers who exit the elevator at that floor.

Assumption 2. If there is more than one passenger boarding the elevator at a landing, they board the elevator on a first-in first-out basis, i.e., in the ascending order of their arrival times.

Assumption 3. If an elevator starts to load passengers assigned to it at a floor, all of them should board, or as many as its capacity allows.

Assumption 4. If more than one passenger leaves the car at a landing, they exit the elevator on a last-in first-out basis, i.e., in the reverse order of their boarding order.

As said earlier, the offline EDPCC can be formulated as a problem of designing the assignment of requests to elevators. This comes from the fact that the CC principle together with Assumptions 1–4 define a unique elevator route for any set of

requests. Natural choice for the decision variables in the EDPCC is therefore the variables specifying the assignment of requests to elevators.

In this paper our aim is to model the EDPCC as a mixed integer linear program with the mentioned decision variables. Nevertheless, we have not found so far a way to write a relation between the assignment variables and time variables saying when a certain location is visited as linear inequalities in the case an elevator has to bypass some requests due to the finite capacity of the elevator. To overcome this issue, we require that the length of each elevator route is at most one round trip, that is, there is no bypassing. The disadvantage of this restriction is that it may rule out the optimal solution of the offline EDP even during a light traffic and make the problem infeasible during heavy traffic since multiple round trips need to be considered for each elevator.

It should be mentioned that when planning an elevator group, sufficient transportation capacity and reasonably short waiting and journey times should be guaranteed. For example, it is recommended that for an excellent service level in office buildings 98 percent of all waiting times should be less than 60 seconds and 75 percent less than 30 seconds (Barney, 2003). Therefore in practice long elevator routes should not often occur.

The input parameters to the EDP considered in this paper are the set of homogeneous elevators, E, and the set of requests, N. Each request $i \in N$ has several parameters: arrival time, origin floor, destination floor, number of passengers, state, and walking time. Walking time is needed in an elevator installation where the request devices are not in the vicinity of the elevator group. Common parameters for the requests are: maximum waiting time, maximum journey time, and transfer time. Transfer time means the time it takes from a passenger to enter or exit an elevator. Each elevator $e \in E$ has several parameters: state, assigned requests, and on-board requests. The state of e tells whether it is vacant, serving requests at some floor, closing doors, opening doors, accelerating, moving at full speed, or decelerating. Common parameters for the elevators are related to door operations and kinematics of the elevators: acceleration, nominal speed, jerk, capacity Q, door opening time, and door closing time.

We define the AF on a directed acyclic multi-graph $G = (V, A, E)$, where V is the vertex set and $A \subset V \times V \times E$ is the arc set. We next describe how the vertices and arcs are defined.

Each request $i = 1, \ldots, n$ is associated with a *pickup vertex* i and a *delivery vertex* $n+i$ that correspond to the origin and destination floors of request i, respectively. Here we assume, without loss of generality, that requests are indexed in the ascending order of their arrival times. Let the sets of pickup and delivery vertices be denoted by P and D, respectively. According to the states of the requests, both of these two sets can be divided into unassigned, assigned, and on-board subsets, denoted by P_1, D_1, P_2, D_2, and P_3, D_3, respectively. Notice that the vertices of P_3 can be omitted as for on-board requests the origin locations have already been visited. Let $e(i)$ denote the elevator for which vertex $i \in P_2 \cup D_2 \cup D_3$ has been fixed to.

With each elevator $e \in E$ is associated a *depot vertex* $2n + e$ representing its initial location at the moment the offline instance is solved. Let the set of depot vertices be denoted by T. Vertex set V is union of these disjoint sets P, D, and T.

Arcs represent the possible movements of the elevators. Let A_e denote the set of arcs for elevator $e \in E$. The set of all arcs is therefore $A = \cup_{e \in E} A_e$. Some of the arcs can be eliminated as they are not used in any feasible route due to the constraints of the problem. Instead of describing which arcs are eliminated, we describe the arc forming procedure. It consists of three phases.

In the first phase, for each elevator e a set B_e is constructed consisting of vertices it can visit and the depot vertex of it. Thus, $B_e = \{2n + e\} \cup P_1 \cup D_1 \cup \{i \in P_2 \cup D_2 \cup D_3 | e(i) = e\}$.

In the second phase, each B_e is ordered according to the CC principle and the Assumptions 1–4, but without taking into account the capacity of e. Let $O_e = [k_1, \ldots, k_{|B_e|}]$ denote the ordered B_e. In the ordering, the moving state of the elevator at the current moment is taken into account. For example, if an elevator is moving at full speed, say 6 meter per second, and the deceleration is 1 meter per second square, it takes 18 meters to make a full stop. So all the requests that are within that region cannot be served without changing the travel direction.

In the third phase arcs are formed. Set A_e contains arc (k_g, k_h), where $k_g, k_h \in O_e$, if and only if $g < h$ and elevator e can travel along arc (k_g, k_h) without violating any constraints or assumptions. To determine which arcs can be traveled by an elevator we follow the rules presented in Ruokokoski et al. (2015).

Each subgraph $G_e = (V_e, A_e)$ has the following properties. 1) It is acyclic. 2) The length of any feasible route is a round trip at the maximum. 3) For any pickup vertex $i \in V_e$ there is no path from vertex $n + i$ to i.

We define the RF on a directed graph $G' = (V' = V \cup \{0\}, A')$. Vertex 0 is an auxiliary terminal depot vertex at which each elevator ends its route. Arc set A' consists of an arc (i, j) if at least one elevator can travel along it. Therefore in the RF we associate with each arc (i, j) a binary decision variable x_{ij} being equal to 1 if some elevator travels along it, otherwise equal to 0.

The next example illustrates the graphs for both formulations and some concepts given above.

Example 1. Consider an example with two elevators, $e1 - e2$, and five requests, $r1 - r5$, of which $r1$ and $r2$ are on-board requests of $e1$. Elevator $e2$ is vacant and located at floor 1. Elevator $e1$ is standing at floor 2 and is about to move to floor 4. The example is visualized in Fig. 1. In the figure darts and circles represent origin and destination floors of the requests, respectively, and the arrows their directions. Squares represent elevator locations. Fig. 2 illustrates the graph of the AF. Vertices 1 and 2 are missing since they have already been visited. Solid arrows represent the arcs of $e1$ whereas dashed represent the arcs of $e2$. The ordered sets formed during the arc forming procedure are: $O_{e1} = [11, 3, 6, 7, 8, 5, 10, 4, 9]$ and $O_{e2} = [12, 4, 3, 8, 9, 5, 10]$. The numbers next to vertices are vertex

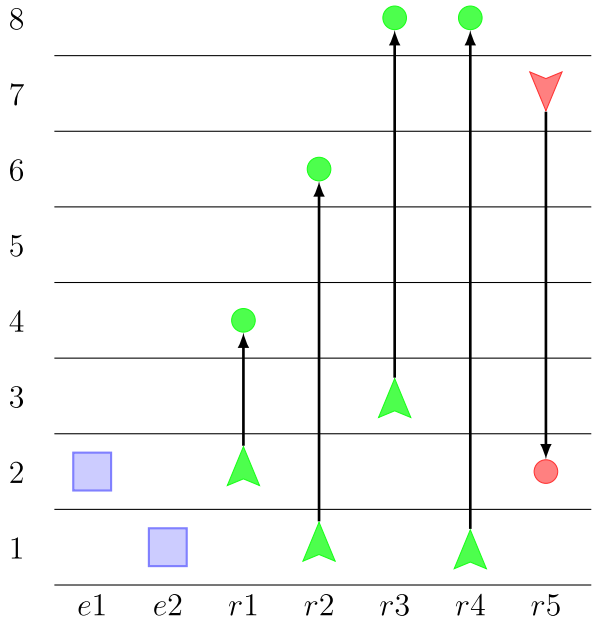

Fig. 1. An instance with two elevators, $e1 - e2$, and five requests, $r1 - r5$. Darts and circles represent origin and destination floors, respectively, while squares represent elevator locations.

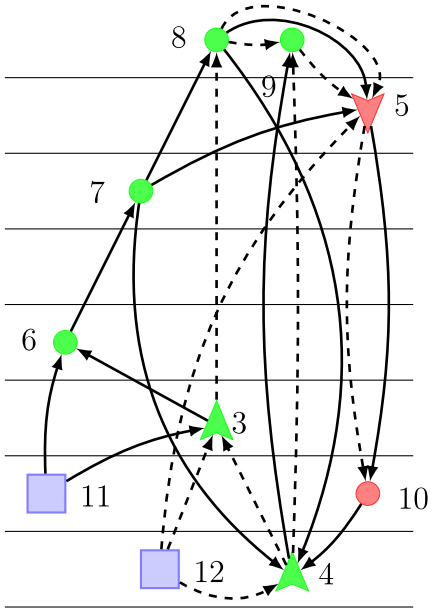

Fig. 2. Illustration of the AF graph. Vertices 11 and 12 represent elevators. Vertices 1 and 3 are missing since they have already been visited. Solid arrows represent the arcs of $e1$ whereas dashed represent the arcs of $e2$.

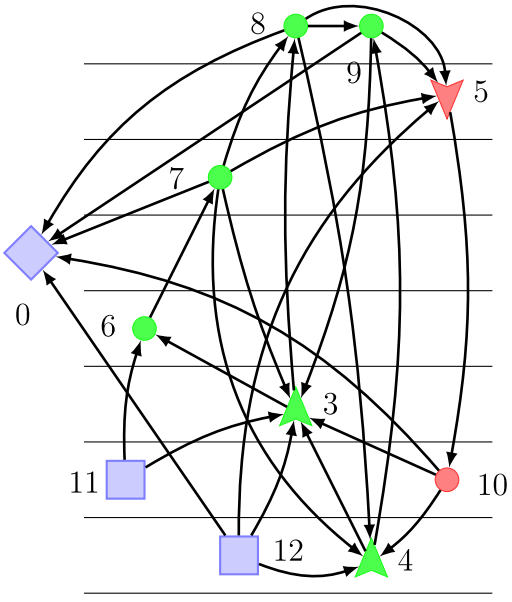

Fig. 3. Illustration of the RF graph. Vertex 0 represent an auxiliary terminal depot vertex at which elevators end their routes. Solid arrows represent the arcs.

indexes. Fig. 3 illustrates the graph of the RF. Vertex 0 is an auxiliary terminal depot vertex. Solid arrows represent the arcs.

3. Assignment formulation of the EDP

3.1. Notation

As described in the previous section, we define the AF on a directed acyclic multi-graph $G = (V, A, E)$. We now describe the parameters involved in it and give some definitions.

Each vertex $i \in V$ is associated with a demand ω_i satisfying for $i \in T$, $\omega_i \geq 0$, and for $i = 1, \ldots, n$, $\omega_i \geq 0$ and $\omega_{n+i} = -\omega_i$. For $i \in T$, ω_i represents the initial load of elevator i and for $i \in P$, ω_i represents the number of passengers of request i. Denote $\Omega(S) = \sum_{i \in S} |\omega_i|$, where $S \subseteq V$.

A time window $[a_i, b_i]$ is also associated with every vertex $i \in V$, where a_i and b_i represent the earliest and latest time, respectively, at which service may start at vertex i. Given with each vertex $i \in P$ is an *elapsed time* $\gamma_i \geq 0$, the time from the arrival time of request i to the current moment.

Each arc $(i, j) \in A$ is associated with cost τ_{ij} which includes passengers transfer times at vertex i and elevator performance times between vertices i and j. In the next section we describe in detail how the costs are calculated.

Let $S_{eij} = \{k_g, k_{g+1}, \ldots, k_{h-1}, k_h\}$ be a subset of O_e (recall that O_e is the ordered set of vertices of elevator e) such that $k_{g-1} = i$ and $k_{h+1} = j$. Each S_{eij} is unambiguous. S_{eij} can be empty.

We define the following decision variables. For each vertex $i \in V$ and each elevator $e \in E$, let x_{ei} be a binary variable equal to 1 if elevator e visits at vertex i, and 0 otherwise. For each vertex $i \in V$, let a continuous time variable t_i be the arrival time and q_i be the elevator load upon departure at vertex i.

3.2. Formulation

With this notation, the mathematical formulation of the AF is:

$$\min \sum_{i \in P} \frac{\omega_i}{\Omega(P)} (t_i + \gamma_i) \tag{1}$$

subject to

$$\sum_{e \in E} x_{ei} = 1 \qquad \forall i \in P_1 \cup D_1 \tag{2}$$

$$x_{e,2n+e} = 1 \qquad \forall e \in E \tag{3}$$

$$x_{ei} = x_{e,n+i} \qquad \forall i \in P_1, e \in E \tag{4}$$

$$x_{e(i),i} = 1 \qquad \forall i \in P_2 \cup D_2 \cup D_3 \tag{5}$$

$$a_i \leq t_i \leq b_i \qquad \forall i \in P \cup D \tag{6}$$

$$t_i + \tau_{ij} - M_{ij}(2 - x_{ei} - x_{ej}) \leq t_j \qquad \forall e \in E, (i, j) \in A_e \tag{7}$$

$$t_i = 0 \qquad \forall i \in T \tag{8}$$

$$q_i + \omega_j - N_i(2 - x_{ei} - x_{ej} + \sum_{k \in S_{eij}} x_{ek}) \leq q_j \qquad \forall e \in E, (i, j) \in A_e \tag{9}$$

$$q_i = \omega_i \qquad \forall i \in T \tag{10}$$

$$\max\{0, \omega_i\} \leq q_i \leq N_i \qquad \forall i \in P \cup D \tag{11}$$

$$x_{ei} \in \{0, 1\} \qquad \forall i \in P \cup D, e \in E \tag{12}$$

The cost function (1) minimizes the average passenger waiting time. We use this objective function here as it is one of the most common objective functions used for the EDP. If the average journey time is minimized, the objective function becomes

$$\min \sum_{i \in D} \frac{|\omega_i|}{\Omega(D)} (t_i + \gamma_i). \tag{13}$$

Equalities (2) and (3) together with binary constraints (12) ensure that each unassigned pickup and delivery vertex is assigned to exactly one elevator and each elevator starts its route at its depot vertex, respectively. For each request, both the pickup and delivery vertices are visited by the same elevator (4). Precedence inequalities (origin vertex of request i must be visited before delivery vertex $n + i$) are automatically satisfied as for any vertex $n + i \in D_1 \cup D_2$ there is no path from it to i in any subgraph G_e, $e \in E$ that contains the vertices $n + i$ and i. Fixed requests cannot be reassigned (5).

Inequalities (6) impose the time window constraints. For $i \in P$, lower bound of the time window is the remaining walking time of request i. For $n+i \in D_1 \cup D_2$ we set $a_{n+i} = a_i + \tau_{i,n+i}$. For $n+i \in D_3$ we set the lower bound to the earliest time the elevator $e(n+i)$ can reach vertex $n+i$. Upper bounds of the time windows are used to restrict the long waiting and journey times of passengers. In particular, we set for $i \in P$, $b_i = W - \gamma_i$ and for $n+i \in D$, $b_{n+i} = J - \gamma_i$. Here W and J denote the maximum allowed waiting and journey times of passengers, respectively. In this way the waiting and journey times of a passenger never exceed the values of W and J, respectively.

Consistency of the time variables, t_i, is ensured by constraints (7). The constraint associated with arc (i, j) means that if elevator e visits at both vertices i and j then t_j must be at least $t_i + \tau_{ij}$. This relation must hold even if vertex j is not directly visited after i since the triangle inequality holds for arc costs and subgraph G_e is acyclic. In other cases the inequality must be redundant. This requirement is guaranteed by setting $M_{ij} = \max\{0, b_i + \tau_{ij} - a_j\}$. For example if $x_{ei} = 0$ and $x_{ej} = 1$, then the remaining constraint is $t_i - b_i + a_j \le t_j$ which is redundant since $t_i \le b_i$ and $t_j \ge a_j$ (assuming $b_i + \tau_{ij} - a_j > 0$).

Consistency of the load variables, q_i, is ensured by constraints (9), in a very similar way as for time variables. Since the triangle inequality does not hold for demands, we need an additional term, $\sum_{k \in S_{eij}} x_{ek}$, to make the constraint associated with arc (i, j) redundant in the case elevator e visits, in addition to vertices i and j, any other vertex k between i and j, i.e., $k \in S_{eij}$ and $x_{ek} = 1$. The validity of the constraints is ensured by setting $N_i = \min\{Q, Q + \omega_i\}$. For example if $x_{ei} = x_{ej} = 1$ and $\sum_{k \in S_{eij}} x_{ek} = 0$, that is, vertex j is visited directly after vertex i, then the remaining constraint is $q_i + \omega_j \le q_j$, whereas if $x_{ei} = x_{ej} = 1$ and $\sum_{k \in S_{eij}} x_{ek} = 1$, then the remaining constraint is $q_i + \omega_j - \min\{Q, Q + \omega_i\} \le q_j$, which is redundant as $q_i \le \min\{Q, Q + \omega_i\}$ and $\omega_j \le q_j$.

Each elevator starts with its initial load (10). Finally, inequalities (11) impose capacity constraints. For example after leaving a pickup vertex i the load must be at least ω_i whereas leaving a delivery vertex j the load must be less than or equal to $Q - |\omega_j|$.

3.3. Arc cost

Let us now consider how to calculate arc cost τ_{ij} from vertex i to j. Let κ, α, and υ denote the jerk, acceleration and nominal speed of the elevators, respectively. Define $f(i)$ to be the floor of vertex i. We assume that for each elevator the deceleration is the same as the acceleration. There are two cases to be considered: 1) start vertex of arc (i, j) is depot vertex, $i \in T$, and 2) start vertex is either pickup or delivery vertex, $i \in P \cup D$. In case $i \in P \cup D$, if $f(i) \neq f(j)$, cost τ_{ij} consists of passenger transfer time and constant door closing time at vertex i, a flight time θ_{ij} between vertices i and j, and a constant door opening time at vertex j; otherwise τ_{ij} just consists of the passenger transfer time at vertex i. The *passenger transfer time* at a vertex is proportional to the demand of the vertex. A formula for the *flight time* from vertex i to j is a piece-wise nonlinear function (Roschier & Kaakinen, 1979). If the distance d_{ij} between $f(i)$ and $f(j)$ is less than or equal to $\upsilon^2/\alpha + \alpha\upsilon/\kappa$ (nominal speed will not be reached), then

$$\theta_{ij} = \sqrt{\frac{\alpha^2}{\kappa^2} + \frac{4d_{ij}}{\alpha}} + \frac{\alpha}{\kappa},$$

otherwise (nominal speed will be reached)

$$\theta_{ij} = \frac{d_{ij}}{\upsilon} + \frac{\upsilon}{\alpha} + \frac{\alpha}{\kappa}.$$

If $i \in T$, the calculation of τ_{ij} is a bit complicated and case-dependent since the state of the elevator (recall closing doors, opening doors, accelerating, decelerating etc.) must be taken into account, and is therefore omitted here.

3.4. Elevators without directions

In the arc forming procedure we tacitly assumed that every elevator has a travel direction. If there is an elevator e without travel direction (i.e., it is standing idle without any requests on a floor), it is replaced by two elevators, e^u and e^d, the travel directions of which are upwards and downwards, respectively. The new travel direction of e is determined based on whether e^d or e^u gets requests: if e^u gets requests, then the direction is upwards, and if e^d gets requests, then the direction is downwards. In order to ensure that only one of these two elevators receives requests, we introduce two additional binary variables, y_d for e^d and y_u for e^u, and add the following constraints

$$\sum_{i \in P} x_{e^d,i} \le |P| y_d, \tag{14}$$

$$\sum_{i \in P} x_{e^u,i} \le |P| y_u, \tag{15}$$

$$y_d + y_u = 1. \tag{16}$$

Here constraint (16) requires that the travel direction of e is either upwards or downwards. Constraint (14) means that if direction is upwards, then e^d cannot get any requests, and correspondingly constraint (15) ensures that if direction is downwards, then e^u cannot get any requests.

4. Analysis of some traffic patterns

A traffic pattern can be characterized by its traffic components: incoming, outgoing and interfloor passengers (Siikonen, 1993). *Incoming* passengers depart from the entrance floor of the building and are destined to upper floors of the building. *Outgoing* passengers leave the upper floors and are destined to the entrance floor. *Interfloor* passengers travel only between the upper floors. Here we consider three well-known traffic patterns, up-peak, down-peak, and mixed. Table 1 gives the proportions of the traffic components for each of these patterns.

The next four theorems and corollaries show when the optimal solutions of the AF and RF are the same and when not. Observe that these results may not hold for other objective functions than those used here.

Theorem 1. *For any instance arising during the up-peak traffic pattern in buildings with one entrance floor and with unlimited capacity of the elevators, the optimal solution of the AF is optimal for the RF.*

Proof. Consider any instance that satisfies the conditions given in the theorem. Suppose that the entrance floor is at the lowest floor. In order to prove the theorem, we need to show that for any feasible assignment of requests to elevators there is a unique route. Let us therefore consider the route construction of an elevator for a given assignment. Without loss of generality, we start the consideration from the point where the elevator is empty and is returning back to the entrance floor.

Table 1
Traffic patterns.

Traffic pattern	Traffic components (percent)		
	Incoming	Outgoing	Interfloor
Up-peak	100	0	0
Down-peak	0	100	0
Mixed	40	40	20

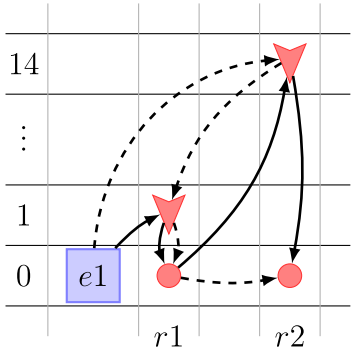

Fig. 4. Illustration of the optimal solutions of the RF (solid arcs) and AF (dashed arcs) for an instance arising during down-peak traffic pattern. (Square represents elevator $e1$, darts departure floors of down requests, and circles destination floors of the requests.).

Table 2
Waiting time (WT) and journey time (JT) of the requests in optimal solutions of both formulations (seconds) for the instance illustrated in Fig. 4.

	RF			AF			Difference (percent)
	r1	r2	Avg	r1	r2	Avg	
WT	10.41	47.95	29.18	50.15	25.13	37.64	22.48
JT	21.82	74.07	47.95	61.56	62.56	62.06	22.73

When the elevator arrives at the entrance floor to serve passengers assigned to it, the service order of them is unique due to Assumption 2. In addition, Assumption 3 implies that the elevator cannot leave the floor before all passengers are on-board. After the passengers are served there, the elevator starts to move upwards. The first stop must be at the nearest floor where some passengers want to leave the elevator, otherwise reversal constraint is violated. At that floor, the order of the leaving passengers is unique due to Assumption 4. The same procedure holds also for the remaining floors at which the elevator has to visit. Since at each point in the route construction the next action is unique, for any feasible assignment of requests there is only a single route. □

Corollary 2. *If the capacity is limited, the optimal solution of the AF is not necessarily optimal for the RF.*

Proof. Suppose that in the optimal solution for the RF at least one elevator needs to visit the entrance floor twice in order to transport all passengers assigned to it. That solution is not a feasible solution for the AF as two round trips are needed. This means that the optimal solution of the AF is not optimal for the RF. □

Theorem 3. *For an instance arising in the down-peak traffic pattern the optimal solution of the AF is not necessarily optimal for the RF.*

Proof. We prove this by an example where the optimality fails. Suppose that for a vacant elevator $e1$ that is located at floor 1 doors open is assigned two down destination requests, $r1$ and $r2$. The first request, $r1$, is from floor 1 to 0 and the second one, $r2$, is from 14 to 0. Fig. 4 illustrates this example. The floor heights and elevator parameters are the same as in Section 6. In this figure, the optimal solution of the AF is represented by dashed arcs. This solution differs from the optimal solution of the RF that is represented in the figure by the solid arcs. Table 2 reports the waiting time (WT) and journey time (JT) for the requests and for both formulations. Column Avg tells the average waiting and journey times. The last column reports how much in percentages the average waiting and average journey times have decreased when using the RF instead of the AF. □

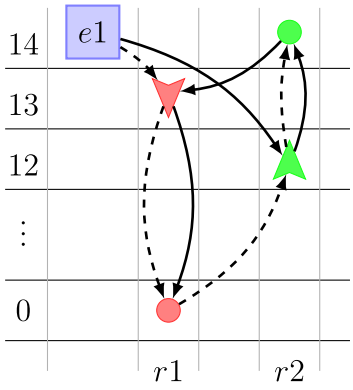

Fig. 5. Illustration of the optimal solutions of the RF (solid arcs) and AF (dashed arcs) for an instance arising during mixed traffic pattern.

Table 3
Waiting time (WT) and journey time (JT) of the requests in optimal solutions of both formulations (in seconds) for the instance illustrated in Fig. 5.

	RF			AF			Difference (percent)
	r1	r2	Aver.	r1	r2	Aver.	
WT	36.21	11.90	24.06	10.41	59.36	34.89	31.04
JT	61.24	24.80	43.02	35.44	72.26	53.85	20.11

Theorem 4. *For an instance arising in the mixed traffic pattern the optimal solution of the AF is not necessarily optimal for the RF.*

Proof. We prove this by an example, too. Suppose that for a vacant elevator $e1$ that is located at floor 14 doors open is assigned two destination requests, $r1$ and $r2$. The first request, $r1$, is from floor 13 to 0 and the second one, $r2$, is from 12 to 14. Fig. 5 illustrates this example. In a similar way as in the proof of the previous theorem, in this figure the optimal solution of the AF is represented by dashed arcs and the optimal solution of the RF is represented by solid arcs. Table 3 reports the waiting times (JT) and journey times (JT) for both formulations. Column Avg tells the average waiting and journey times. The last column reports how much in percentages the average waiting and average journey times are decreased when using the RF instead of the AF. □

5. Branch-and-bound algorithm

In this section, we describe briefly the branch-and-bound (BAB) algorithm used to solve the considered instances. First one more definition. Let X be a path that starts at some elevator origin vertex and is infeasible with respect to some of the four assumptions given in Section 2.

For the AF we use the basic BAB algorithm while in the case of the RF we add some constraints into the model if and only if they are violated in an LP-relaxation during the search process. The reason is that in the RF the cardinality of some constraint families are exponential with respect to the number of requests, and therefore cannot be added a priori. These constraints are the precedence, fixed vertex and infeasible path constraints. We call separation algorithms of them at each node to detect violated constraints.

The precedence constrains stipulate that for each request i, the pickup vertex i is visited before the delivery vertex $n+i$ and both vertices are visited by the same elevator. We identify violated precedence constraints by solving maximum flow problems. For those readers interested in the technical details we refer to a paper by Ropke, Cordeau, and Laporte (2007). Fixed vertex constraints ensure that if vertex i has been fixed to elevator e, then other elevators cannot serve it. In other words, in the solution there must be a path from $2n+e$ to i. We identify violated fixed vertex

constraints using the same idea as in the case of precedence constraints. If the flow from $2n+e$ to i is less than one, a violated fixed vertex constraint is identified.

Infeasible path X can be eliminated by adding the following constraint

$$\sum_{(i,j)\in A_X} x_{ij} \leq |A_X| - 1$$

where A_X is the set of arcs in path X. Recall that in the RF we associate with each arc (i, j) a decision variable x_{ij}. We separate violated infeasible paths by starting from an elevator origin vertex and then iteratively augment the path by adding an arc whose value in the solution is more than 0.9. If the augmented path becomes infeasible at some point, then the infeasible path constraint is added. This procedure is repeated for each elevator.

6. Computational experiments

The algorithm used in the experiments was implemented in C++ with IBM ILOG CPLEX Optimization Studio 12.5.1. The experiments were performed on a computer equipped with Intel dual core processor running at 3.0 gigahertz and having 8 gigabyte of memory. In the literature there are lots of data related to traffic profiles. Despite that, there are no data for single instances of the offline EDP. Therefore, we generate a set of random instances that is described next.

6.1. Description of test cases

Instead of generating pure random offline EDP instances, we first simulated a passenger traffic flow in a building by using a discrete-event simulator called Building Traffic Simulator (BTS) (Siikonen, Susi, & Hakonen, 2001) and then took some random instances solved during the simulation. BTS models the building, elevators, group control system and traffic generation. The traffic generator forms passengers according to the selected traffic pattern and Poisson distribution with arrival rate λ. Destination requests given by the passengers are assigned to elevators by solving the offline EDP instances using a genetic algorithm in the group control module.

Numerical values for the parameters used in the simulations are as follows. The floor height of each floor is 3.6 meters. The lowest floor, 0, is an entrance floor. The population of each floor above the entrance floor is 50. Each elevator has rated capacity 21 persons, nominal speed 3 meter per second, acceleration 1.0 meter per second square, and jerk 1.6 meter per second cube. Door opening time is 1.5 seconds and closing time 3.1 seconds, passenger transfer time in both directions, into and out from an elevator, is 1.0 second. Passenger arrival rate, λ, was set so that the generated traffic corresponds to light or normal traffic condition. Finally, the walking time from a destination control panel to the waiting area is 0 seconds.

We call the selected random instances basic instances. In practice each basic instance has only one non-assigned request since in a DC requests are assigned immediately and solving an instance takes usually a few milliseconds. In order to study how many calls our approach is able to assign we formed 5 instances from each basic instance by first removing the non-assigned request and 11 assigned requests and then adding 4, 6, 8, 10, or 12 of them back as non-assigned requests. This kind of instances can occur in an elevator system with a request prediction, in a DC with delayed assignment, or in a system where passengers have personal request giving devices, for example a smart phone, which makes possible to delay the announcement of the serving elevator. By varying the number of elevators and the traffic type in the BTS we selected 12 random basic instances and formed 60 instances in total. These instances represent instances arising in light and normal traffic conditions.

Table 4
Considered offline EDP instances.

			Traffic type					
			Up		Down		Mixed	
$\|E\|$	$\|F\|$	$\|C_1\|$	$\|C_{2,3}\|$	Name	$\|C_{2,3}\|$	Name	$\|C_{2,3}\|$	Name
1	9	4	3	U_1_4	4	D_1_4	2	M_1_4
		6		U_1_6		D_1_6		M_1_6
		8		U_1_8		D_1_8		M_1_8
		10		U_1_10		D_1_10		M_1_10
		12		U_1_12		D_1_12		M_1_12
2	9	4	4	U_2_4	10	D_2_4	3	M_2_4
		6		U_2_6		D_2_6		M_2_6
		8		U_2_8		D_2_8		M_2_8
		10		U_2_10		D_2_10		M_2_10
		12		U_2_12		D_2_12		M_2_12
4	15	4	7	U_4_4	7	D_4_4	25	M_4_4
		6		U_4_6		D_4_6		M_4_6
		8		U_4_8		D_4_8		M_4_8
		10		U_4_10		D_4_10		M_4_10
		12		U_4_12		D_4_12		M_4_12
6	22	4	32	U_6_4	29	D_6_4	39	M_6_4
		6		U_6_6		D_6_6		M_6_6
		8		U_6_8		D_6_8		M_6_8
		10		U_6_10		D_6_10		M_6_10
		12		U_6_12		D_6_12		M_6_12

Table 5
Number of decision variables, $|x|$, for each instance and both formulations.

Instance	RF	AF	Instance	RF	AF	Instance	RF	AF
U_1_4	43	4	D_1_4	74	4	M_1_4	37	4
U_1_6	87	6	D_1_6	130	6	M_1_6	149	6
U_1_8	150	8	D_1_8	199	8	M_1_8	262	8
U_1_10	231	10	D_1_10	284	10	M_1_10	227	10
U_1_12	327	12	D_1_12	385	12	M_1_12	259	12
U_2_4	69	8	D_2_4	106	8	M_2_4	56	8
U_2_6	123	12	D_2_6	178	12	M_2_6	100	12
U_2_8	197	16	D_2_8	256	16	M_2_8	159	16
U_2_10	289	20	D_2_10	350	20	M_2_10	230	20
U_2_12	407	24	D_2_12	453	24	M_2_12	308	24
U_4_4	60	16	D_4_4	131	16	M_4_4	148	16
U_4_6	114	24	D_4_6	211	24	M_4_6	225	24
U_4_8	181	32	D_4_8	303	32	M_4_8	315	32
U_4_10	266	40	D_4_10	411	40	M_4_10	399	40
U_4_12	373	48	D_4_12	539	48	M_4_12	506	48
U_6_4	112	24	D_6_4	342	24	M_6_4	225	24
U_6_6	173	36	D_6_6	494	36	M_6_6	307	36
U_6_8	254	48	D_6_8	666	48	M_6_8	393	48
U_6_10	350	60	D_6_10	830	60	M_6_10	519	60
U_6_12	464	72	D_6_12	1019	72	M_6_12	621	72

Table 4 provides the basic characteristics of the formed instances. The first column, $|E|$, gives the number of elevators, the second one, $|F|$, gives the number of floors whereas the third one, $|C_1|$, gives the number of non-assigned requests. It is assumed that each elevator can serve any floor. The last 6 columns report the instance names for each traffic type and the total number of assigned and on-board requests, $|C_{2,3}|$. The first character in an instance name is an abbreviation for the traffic type whereas the first and second number are the number of elevators and the number of non-assigned requests, respectively. The traffic types are up (U), down (D), and mixed (M), whose traffic components are given in Table 1.

The number of binary decision variables ($|x|$) for both formulations are given in Table 5. For the AF it is $|E| \times |C_1|$ whereas for the RF it equals to the number of arcs in the graph associated with the instance.

Table 6
Computational results for up-peak traffic instances.

	AWT				AJT			
	RF		AF		RF		AF	
Instance	CPU	z	CPU	z	CPU	z	CPU	z
U_1_4	0.07	79.14	0.09	**79.14**	0.23	91.52	0.02	**91.52**
U_1_6	0.17	78.58	0.05	**78.58**	0.53	103.35	0.02	**103.35**
U_1_8	0.22	77.06	0.04	**77.06**	11.53	107.34	0.02	**107.34**
U_1_10	0.80	75.70	0.04	**75.70**	4790.86	109.63	0.02	**109.63**
U_1_12	30.78	74.28	0.05	**74.28**	*7200.00	118.93	0.02	**118.93**
U_2_4	0.08	46.45	0.08	**46.45**	0.32	75.30	0.11	**75.30**
U_2_6	0.21	45.83	0.10	**45.83**	2.49	76.71	0.16	**76.71**
U_2_8	0.34	44.39	0.10	**44.39**	173.46	78.67	0.17	**78.67**
U_2_10	1.83	43.06	0.14	**43.06**	*7200.00	79.32	0.25	**79.32**
U_2_12	56.83	41.67	0.20	**41.67**	*7200.00	82.54	0.60	82.40
U_4_4	0.07	61.64	0.07	**61.64**	0.21	102.10	0.15	**102.10**
U_4_6	0.18	59.02	0.14	**59.02**	0.53	99.53	0.22	**99.53**
U_4_8	0.54	57.73	0.16	**57.73**	9.29	97.62	1.66	**97.62**
U_4_10	15.24	56.42	0.21	**56.42**	1052.82	95.02	7.13	**95.02**
U_4_12	3948.01	53.60	1.54	**53.60**	*7200.00	92.37	39.79	**92.37**
U_6_4	0.16	69.75	0.09	**69.75**	0.29	97.19	0.19	**97.19**
U_6_6	0.22	66.58	0.19	**66.58**	1.43	97.28	0.62	**97.28**
U_6_8	0.82	64.23	0.16	**64.23**	39.28	97.10	6.50	**97.10**
U_6_10	3.96	61.53	0.52	**61.53**	*7200.00	96.79	138.29	**96.79**
U_6_12	589.46	59.08	1.87	**59.08**	*7200.00	96.83	5761.89	**96.83**
Average	232.50		0.29		2464.16		297.89	

Table 7
Computational results for down-peak traffic instances.

	AWT				AJT			
	RF		AF		RF		AF	
Instance	CPU	z	CPU	z	CPU	z	CPU	z
D_1_4	0.13	57.19	0.06	**57.19**	0.21	67.76	0.02	78.20
D_1_6	0.50	55.81	0.04	**55.81**	1.30	74.70	0.02	86.80
D_1_8	1.26	57.44	0.06	**57.44**	68.05	81.74	0.02	89.00
D_1_10	109.24	60.17	0.04	**60.17**	4286.10	86.57	0.02	91.91
D_1_12	*7200.00	61.28	0.04	**61.28**	*7200.00	90.27	0.02	95.72
D_2_4	0.09	47.99	0.07	**47.99**	0.28	63.95	0.15	70.22
D_2_6	0.20	41.54	0.16	**41.54**	3.11	64.47	0.18	69.93
D_2_8	0.86	40.54	0.19	**40.54**	35.09	64.94	0.18	69.79
D_2_10	4.58	40.27	0.19	**40.27**	736.76	65.92	0.23	69.18
D_2_12	152.74	40.25	0.25	**40.25**	*7200.00	66.92	0.38	69.73
D_4_4	0.09	35.24	0.08	**35.24**	0.50	71.12	0.14	**71.12**
D_4_6	0.10	34.75	0.15	**34.75**	15.24	71.19	0.35	72.14
D_4_8	0.26	34.73	0.19	**34.73**	26.92	71.53	1.72	71.99
D_4_10	7.07	35.61	1.51	**35.61**	*7200.00	72.71	17.43	74.50
D_4_12	141.15	35.90	9.75	**35.90**	*7200.00	74.91	384.25	76.51
D_6_4	0.37	50.43	0.19	**50.43**	515.13	84.64	0.18	89.97
D_6_6	0.82	47.20	0.30	**47.20**	*7200.00	83.49	0.33	88.88
D_6_8	4.94	45.44	0.36	**45.44**	*7200.00	83.63	3.01	88.36
D_6_10	181.92	44.67	6.03	**44.67**	*7200.00	85.04	92.31	88.12
D_6_12	918.11	42.13	42.74	**42.13**	*7200.00	84.65	1237.20	88.31
Average	436.22		3.12		3164.43		86.91	

In all instances the maximum waiting and journey time is set to 400 and 800 seconds, respectively.

6.2. Comparison of the RF and AF

Each generated instance was solved by both formulations and with both objective functions, average waiting time (AWT) (1) and average journey time (AJT) (13). A limit of 7200 seconds for CPU time was used. Tables 6–8 provide the CPU time (CPU) and the objective value (z) for each instance and for both objective functions. When an instance could not be solved to optimality within the predefined CPU limit, a character $*$ precedes to indicate that. An objective value for the AF is given in bold if it equals to that of the RF.

From these tables one can observe that the BAB algorithm for the AF is faster than for the RF for almost all instances. To be precise, the AF with the AWT is faster for 58 instances out of 60 and with the AJT for all instances. For some instances the difference is quite huge. For example, the CPU time of the AF with the AWT on instance M_6_12 is about one second whereas for the RF it is more than one hour. On average, the AF with the AWT (AJT) is 62.46 percent (80.35 percent), 65.67 percent (93.41 percent), and 81.15 percent (95.40 percent) faster than the RF for up-peak, down-peak and mixed traffic instances, respectively. One major factor explaining the difference in CPU times is the number of binary decision variables which were given in Table 5. From the results one can also observe the change from the AWT to AJT makes instances much more difficult to solve for the BAB algorithm.

Table 8
Computational results for mixed traffic instances.

	AWT				AJT			
	RF		AF		RF		AF	
Instance	CPU	z	CPU	z	CPU	z	CPU	z
M_1_4	0.10	74.20	0.04	**74.20**	0.20	81.48	0.02	81.80
M_1_6	0.57	76.09	0.05	**76.09**	1.26	90.57	0.02	91.85
M_1_8	23.88	78.05	0.05	**78.05**	93.28	101.47	0.02	104.07
M_1_10	961.98	75.59	0.05	**75.59**	*7200.00	104.43	0.02	107.00
M_1_12	*7200.00	–	0.04	**71.69**	*7200.00	–	0.02	107.59
M_2_4	0.14	65.60	0.08	**65.60**	0.29	73.62	0.14	74.69
M_2_6	0.42	59.26	0.09	**59.26**	0.83	71.88	0.13	73.78
M_2_8	13.56	55.37	0.10	**55.37**	72.44	76.27	0.19	77.83
M_2_10	620.54	52.65	0.15	**52.65**	*7200.00	76.04	0.26	77.90
M_2_12	*7200.00	49.51	0.23	**49.51**	*7200.00	79.88	0.70	80.80
M_4_4	0.09	101.19	0.08	**101.19**	1.53	113.87	0.17	114.19
M_4_6	0.10	87.49	0.10	**87.49**	83.00	111.46	0.20	112.08
M_4_8	0.99	80.87	0.13	**80.87**	792.07	109.10	0.38	110.47
M_4_10	18.50	76.47	0.18	**76.47**	*7200.00	107.83	1.21	109.45
M_4_12	*7200.00	70.08	0.35	**70.08**	*7200.00	107.40	28.62	108.94
M_6_4	0.44	73.74	0.13	**73.74**	108.39	112.46	0.26	113.98
M_6_6	1.90	72.97	0.26	**72.97**	1731.15	112.05	0.55	113.81
M_6_8	13.42	65.83	0.30	**65.83**	*7200.00	111.88	3.22	113.23
M_6_10	307.63	60.79	0.60	**60.79**	*7200.00	110.88	7.74	111.59
M_6_12	4077.67	54.99	1.07	**54.99**	*7200.00	111.92	280.99	111.78
Average	1382.10		0.20		3384.22		16.24	

When considering the absolute CPU times one sees that both models can solve only small instances to optimality within half a second which is usually the upper limit for the CPU time in real installations. For instances arising in a DC this may be enough as there the number of requests to be assigned is usually just one or two. However, in a DC with predicted destination requests or in systems where some requests can be reassigned this may not be fast enough.

Let us next consider the objective values. First the focus is on the AWT objective values. The results show that for all instances solved to optimality, both formulations find the same solution. This observation indicates that during light and normal traffic such an instance is rare for which the optimal solution of both formulations are different. The major reason is the objective function itself. When the AWT is minimized, elevators try to get passengers on-board as fast as possible on average, and very often this is achieved by stopping pickup locations in a floor sequence. Also, when there is at least one passenger on-board their destination locations are visited in a floor sequence due to the reversal control principle.

Typically when the formulations do not provide the same solution, some elevator receive requests that are very different in terms of destination and/or origin floors. See, for instance, the examples given in Theorems 3 and 4. However, when a group contains multiple elevators, often such requests are given to different elevators and thus there is no need to deviate from the CC principle.

In case of the AJT objective function the situation is different. The results reveal that for almost all instances solved to optimality except up-peak instances, the objective value of the RF is better. The reason is that when the AJT is minimized, elevators try to visit delivery locations as fast as possible on average and often this is achieved by delaying the service of some passenger to subsequent round trips. On average, the difference in an objective value between RF with the AJT and the AF is 7.13 percent and 1.41 percent for down-peak and mixed traffic instances, respectively.

When we consider the results of Inamoto et al. (2002) we observe that in their experiments, which are performed on random instances formed according to the mixed traffic, the RF with the AJT produces quite different solutions compared to the CC. In particular, they report that the difference is 27.7 percent in objective value over 50 instances with 3 elevators and 10 non-assigned requests. This result differs from our results. The reason for the difference is that in their RF the information related to forthcoming passengers is complete, that is, the origin and destination floors as well as the arrival times of all passengers who will appear in the planning period are known. This information can be utilized to dispatch elevators to the arrival floors of passengers long before the requests are registered. In contrast to this, in the comparison model based on the CC they assume that the origin and destination floors of a passenger are revealed once the destination request is registered. Thus, their results rather show that the use of early information related to passengers improves the service level.

We finally compare the results given in Tanaka et al. (2005) to our results. Recall that they only study the single-elevator case. In their computational experiments they simulate up-peak and down-peak traffics with different arrival rates. Their results show that during light and normal traffic the both models, the CC and the RF with the AJT produce about the same service level of passengers whereas during heavy traffic the service level starts differing greatly. Our hypothesis is that this phenomenon also occurs in a multi-elevator case. However, we believe that this phenomenon is not so strong in the DC considered in this paper. The reason is that Tanaka et al. consider a DC with passenger guidance system that indicates the destination floors of passengers who should board the car when it stops there next. In this way they can more freely select which passenger should board the car. Unfortunately we cannot verify our hypothesis since our selected algorithm, BAB, is too slow for instances arising in heavy traffic.

7. Conclusion

In the EDP we are given an elevator group and a set of requests and the task is to design elevator routes serving all requests. The problem is an online problem since requests arrive gradually during the time. Therefore a rolling horizon scheme is used: an offline instance of the problem is solved repeatedly, for example every time a new request arrives, to update elevator routes.

In many of the approaches presented to the EDP, elevators are assumed to obey the collective control rule. The significance of this

rule is that it reduces the problem to one of designing the assignment of requests to elevators. However, an optimal solution of the EDPCC may not necessary be optimal for the EDP.

Quality considerations of the EDPCC have only been done in a single-elevator case and in a multi-elevator case with very simplified settings and with the assumption that an elevator can pickup any subset of passengers. This kind of system is rather unrealistic in practice

This paper compared the EDPCC with EDP in a multi-elevator case with a typical destination control. We began with a formulation of the EDPCC with decision variables specifying the assignment of requests to elevators. In the formulation the length of each elevator route was restricted to a round trip for modeling reasons. This limits the applicability of our model to light and normal traffics.

We then gave some theoretical properties of the problems. In particular, we proved that for any instance arising in the up-peak traffic in buildings with one entrance floor, the optimal solution of the EDPCC is also optimal for the EDP when the elevators have unlimited capacity. While for down-peak or up-peak traffic instances we showed by examples that the optimal solution of the EDPCC is not necessarily optimal for the EDP, and the difference in the average waiting time can be big, up to 31 percent.

We next carried out computational experiments on random instances arising in light or normal traffic conditions. The test instances were generated with the help of Building Traffic Simulator. When the average waiting time was optimized, both formulations provided the same solution for all test instances. This gives some indication that cases where the solutions differ are rare in practice in elevator groups, and we may conclude that the EDPCC with the AWT is a reasonable choice for light and normal traffic conditions. During heavy or intense traffic conditions it is expected that the solutions differ more often. It is still an open question how much the difference is.

When the average journey time was optimized, almost for each instance except up-peak instances the RF produced a different solution. The difference varied in the range of 0 percent to 16 percent. If the elevator group control system is equipped with a passenger guidance system, the solutions can differ even more since now group control has more freedom in designing the service order of the passengers.

Computation time comparison revealed that the BAB algorithm for the AF is more than 60 percent faster than for the RF. CPU times for single instances also showed that both models can solve only small instances to optimality within half a second which is usually the upper limit for the CPU time in real installations. For instances arising in DCs this may be enough as there the number of requests to be assigned is usually just one or two. However, in DCs with predicted destination requests or in systems where some requests can be reassigned this may not be fast enough.

It should be pointed out that despite the facts that the AF with the AWT objective function produce very often the same solution as the RF and the AF can be considered to be easier to solve, some new features presented for DCs regarding to individually tailored services such as VIP and access control may require the use of the RF, or are easier to implement in the RF. VIP service means that a passenger identified as a VIP must be served with the highest priority. There can be different levels of priorities. Access control can be used to restrict for safety and security reasons the floors to which a certain passenger can go. This is a useful feature in mixed used buildings consisting of separate office, commercial, residential, and hotel floors. Third example is the separation of passenger groups. For example in a multi-tenant offices it may be wanted that passengers from different tenants do not share a car.

Future studies include the development of the AF to cover multiple round trips and of a faster exact algorithm to be able to analyze larger instances. One potential algorithm is the column generation algorithm developed in Hiller et al. (2014). Their model used in the paper is though a heuristic model since all requests that have the same origin floor are replaced with a single request.

References

Barney, G. C. (2003). Elevator traffic handbook. Spon Pr.

Closs, G. D. (1970). *The computer control of passenger traffic in lift systems*. (Ph.D. thesis), The Victoria University of Manchester.

Fernandez, J., Cortes, P., Munuzuri, J., & Guadix, J. (2014). Dynamic fuzzy logic elevator group control system with relative waiting time consideration. *IEEE Transactions on Industrial Electronics, 61*(9), 4912–4919.

Hiller, B., Klug, T., & Tuchscherer, A. (2014). An exact reoptimization algorithm for the scheduling of elevator groups. *Flexible Services and Manufacturing Journal, 26*, 586–608.

Hirasawa, K., Kuzunuki, S., Iwasaki, T., & Kaneko, T. (1978). Optimal hall call assignment method of elevator supervisory control system. In *Proceedings of Joint Automatic Control Conference, USA* (pp. 305–313).

Inamoto, T., Tamaki, H., Murao, H., & Kitamura, S. (2002). An application of branch-and-bound method to deterministic optimization model of elevator operation problems. In *Proceedings of the 41st SICE annual conference: vol. 2* (pp. 987–992).

Levy, D., Yadin, M., & Alexandrovits, A. (1977). Optimal control of elevators. *International Journal of Systems Science, 8*(3), 310–320.

Ropke, S., Cordeau, J.-F., & Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks, 49*(4), 258–272.

Roschier, N.-R. Kaakinen, M. (1979). New formulae for elevator round trip calculation. Supplement to Elevator World fo ACIST Members, (pp. 189–197).

Ruokokoski, M., Ehtamo, H., & Pardalos, P. (2015). Elevator dispatching problem: a mixed integer linear programming formulation and polyhedral results. *Journal of Combinatorial Optimization, 29*, 750–780.

Schröder, J. (1990). Advanced dispatching. *Elevator World, 38*(3), 40–46.

Siikonen, M. L. (1993). Elevator traffic simulation. *Simulation, 61*, 257–267.

Siikonen, M.-L., Susi, T., & Hakonen, H. (2001). Passenger traffic flow simulation in tall buildings. *Elevator World, 49*(8), 117–123.

Smith, R., & Peters, R. (2002). ETD algorithm with destination dispatch and booster options. *Elevator World, 50*(7), 136–145.

Sorsa, J., Hakonen, H., & Siikonen, M.-L. (2006). Elevator selection with destination control system. *Elevator World, 54*(1), 148–155.

Tanaka, S., Uraguchi, Y., & Araki, M. (2005). Dynamic optimization of the operation of single-car elevator systems with destination hall call registration: Part I. formulation and simulations. *European Journal of Operational Research, 167*, 550–573.

Tyni, T., & Ylinen, J. (2001). Genetic algorithms in elevator car routing problem. In S. L. (Ed.), *Proceedings of the genetic and evolutionary conference (GECCO-2001)* (pp. 1413–1422). San Francisco, CA, USA: Morgan Kaufman Publishers.