# Distributed Primal-Dual Saddle Point Optimization Applied to Elevator Group Control Systems

Jeroen Buitendijk and Jannes Hühnerbein
{jeroenb,hjannes}@ethz.ch

***Abstract*—Elevator group control presents an opportunity for the application of distributed optimization algorithms. A distributed approach to elevator group control could make the system more robust to component failure, as the unaffected elevators can continue operation when one of the controllers fails to function. In this paper, we present a novel algorithm based on distributed primal-dual saddle point optimization. This algorithm is benchmarked against the ant colony optimization algorithm, which is well estalished in the literature, as well as two simple dispatching algorithms. All four algorithms were implemented and simulated in a self-developed elevator group control framework using models of people flows in buildings which are well established in the literature. The developed distributed controller outperforms the benchmark controllers for some building sizes and elevator systems.**

***Index Terms*—elevator group control systems, EGCS, primal-dual saddle point optimization, distributed control, distributed optimization, ant colony optimization**

## I. Introduction

For many people, the daily commute includes at least one elevator ride. In the United States alone, elevators complete 18 billion passenger trips a year [1]. This makes the elevator one of the most common modes of transport. Minimizing the time spent on elevators or waiting for one can therefore have a significant positive impact on the quality of life of millions of commuters.

It is possible to describe this problem, which is known as the elevator dispatch problem, as a mixed-integer problem. However, this problem formulation is NP-hard and therefore ill-suited to real-time control. Furthermore, it is difficult to implement in a distributed manner [2]. Therefore, this paper is focused on non-optimal elevator group control algorithms which nonetheless show acceptable performance and can be implemented in a distributed manner. Four non-optimal algorithms are presented. The first algorithm is a self-developed algorithm based on distributed primal-dual saddle point optimization (DPDSP) [3]. The second algorithm is a well established elevator group control algorithm based on ant colony optimization (ACO) [4]. In addition, two simple heuristic-based algorithms are introduced to serve as a benchmark: a "first in, first out" (FIFO) algorithm and a self-developed algorithm based on observations in small and medium sized buildings, which we named "classic elevator dispatching" (CED).

Both the two simple algorithms (FIFO and CED) and the two more advanced algorithms (DPDSP and ACO) are implemented in a simulation framework using MATLAB. Then, the performance of all four algorithms is examined and compared. This allows us to qualify the performance of our novel distributed control scheme through a quantitative comparison with a well-established control scheme (ACO) and simple heuristics-based control schemes.

## II. Elevator Group Dispatching Problem

An elevator group is a set of elevators which are controlled in a coordinated manner. Elevators are dispatched in response to a person registering a hall call. In principle, there are many different ways in which the elevator group can respond to hall calls and the chosen control scheme will have a significant effect on the speed and efficiency of transportation within the building and on user satisfaction. The question of what control scheme is best adopted is known as the elevator group dispatching problem [5].

The formulation of the elevator group dispatching problem depends on a number of factors. First, it is impacted by the architecture of the building and the installed elevator system as well as the occupants of the building. Second, the information structure of the elevator group dispatching problem may be different from one system to another. Conventionally, there are two hall call buttons on every level except for the first and the last level. When a user registers a hall call, he only provides the information of which direction he wishes to travel in. The number of people waiting on a particular floor and the destination of the people are not known to the control system. Alternatively, an information structure known as destination hall call registration (DHCR) can be applied. In this case, each user specifies their destination at the moment that they register a hall call. As this increases the available information, this DHCR allows for the formulation of a broader range of controllers, thus potentially improving performance. Finally, the elevator group dispatching problem may be subject to additional constraints arising from psychological or economic considerations. For example, while direction reversals are known to increase the efficiency of the elevator group control system, they usually cause users discomfort [2], [5].

Considering the problem structure, it is natural to formulate the elevator group dispatching problem as a mixed-integer problem where a cost function is chosen that maximizes utility to the user or the operator. This approach would make it possible to find a trajectory for the elevators whose global optimality is constrained only by the uncertainty of future events. However, the computational complexity of such a mixed-integer problem would be exponential, rendering this

solution impractical for real-time control of an elevator group [2], [5]. Instead, a less ideal but computationally cheaper control scheme needs to be implemented. A number of solutions to this problem have been published in literature, including controllers based on ant colony optimization [4], fuzzy logic controllers [6], controllers based on particle swarm optimization [7], controllers based on neural networks [8], controllers based on genetic network programming and others [9].

In this paper, we do not consider a particular information structure or a particular set of additional constraints a priori; rather, we introduce them with the particular elevator group control scheme (EGCS) we are implementing.

## III. Algorithms

### A. Distributed Primal-Dual Saddle Point Optimization (DPDSP)

In this paper, we present a novel algorithm for elevator group control. Our algorithm is adapted to a distributed system, where each elevator has a separate controller and the controllers are connected through a network.

At the core of our algorithm is a distributed optimization problem which allows us to define the destinations of all of the elevators in the EGCS. Let $y \in \mathbb{R}^n$ be variable that we optimize. The dimension $n$ of $y$ is given by the number of elevators. Let $y_i \in \mathbb{R}^n$ be controller $i$'s local estimate of $y$. The optimization problem is then given by

$$\min_{y \in \mathbb{R}^n} f(y) = \sum_{i=1}^{n} f_i(y), \tag{1}$$

where $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a private cost function known only to controller i and $f(y)$ is the cost function of the entire network of controllers (i.e. the cost function of the EGCS) [3]. However, equation (1) requires that every controller $i$ has knowledge of the variable that is optimized, $y$, which is not given in the case of a network of distributed controllers with private cost functions. Rather, each controller $i$ has its own estimate $y_i \in \mathbb{R}^n$ of the elevator destinations. To ensure that every controller converges to the same solution, a consensus constraint is added to the problem formulation [3].

The consensus constraint $\mathcal{C}(y)$ is constructed by exploiting the graph structure of the controller network. Each controller with its private cost function $f_i(\cdot)$ and its estimate $y_i$ is represented by a node on the graph. Any two controllers which can communicate are connected by an edge on the graph. The controllers communicate their local estimates $y_i$ to neighboring controllers. All communication links are assumed to be identical and bidirectional, which results in an unweighted undirected graph. This allows us to define the consensus constraint using the graph Laplacian [3]:

$$\mathcal{C}(y) = Ly = \mathbf{0}, \tag{2}$$

where $L$ is the graph Laplacian, $y \in \mathbb{R}^n$ is the vector of estimates $y_i$ and $\mathbf{0}$ is the zero vector. The consensus constraint ensures that all local estimates $y_i$ are equal when the algorithm converges and implicitly communicates the private cost functions $f_i(\cdot)$ of the other controllers. Equations (1) and (2) allow us to formulate the distributed optimization problem:

$$\begin{aligned} \min_{y_i \in \mathbb{R}^n} \quad & \tilde{f}(y) = \sum_{i=1}^{n} f_i(y_i), \\ s.t. \quad & Ly = \mathbf{0}. \end{aligned} \tag{3}$$

In order for this problem to be solvable using the distributed primal-dual saddle-flow algorithm, we require that each private cost function $f_i(\cdot)$ is strictly convex and twice continuously differentiable. The primal-dual saddle-flow algorithm can then be expressed as

$$\begin{aligned} \dot{y} &= -\frac{\partial}{\partial} \tilde{f}(y) - Ly - L\lambda, \\ \dot{\lambda} &= Ly, \end{aligned} \tag{4}$$

where $\lambda$ is the dual variable [3]. For individual controllers, equation (4) becomes

$$\begin{aligned} \dot{y}_i &= -\frac{\partial}{\partial y_i} f_i(y_i) - \sum_{j=1}^{n} a_{ij}(y_i - y_j) - \sum_{j=1}^{n} a_{ij}(\lambda_i - \lambda_j), \\ \dot{\lambda}_i &= \sum_{j=1}^{n} a_{ij}(y_i - y_j), \end{aligned} \tag{5}$$

where $a_{ij}$ is an element of the Laplacian matrix and $\lambda_i \in \mathbb{R}^n$ is the estimated dual variable of controller i. In our simulation, we implemented equations (5) using Euler discretization [3].

In our algorithm, each private cost function $f_i(\cdot)$ depends on three factors: the destinations of the people on board the elevator, the locations of people who have called an elevator, and the positions of the other elevators. The first two factors attract the elevator while the third factor pushes the elevator to avoid the other elevators. In order to develop a strictly convex local cost function $f_i(\cdot)$, we formulate the third factor as a point within the building that attracts elevator $i$ away from the other elevators and distributes them throughout the building. In order to find such a point, we first calculate the average position of the other elevators weighted by their free capacity. Next, we determine the point in the building which is the furthest away from this weighted average under the assumption that the top floor is considered adjacent to the ground floor (i.e. the levels of the building are treated as points on a circle). This is always the point that lies within the building and is half the building height away from the weighted average of the positions of the other elevators. We call this point $\bar{x}_i^{elev}$ and the weight given by the total free capacity of the other elevators $w_i^{elev}$.

We can now formulate a twice differentiable strictly convex cost function $f_i(\cdot)$ for each elevator. Let $\bar{x}_i^{dest} \in \mathbb{R}^d$ be the vector of the destinations of the people on board elevator $i$ and let $w_i^{dest}$ be the vector of the time that each passenger has already been waiting since submitting a hall call. Let $\bar{x}_i^{origin} \in \mathbb{R}^o$ be the vector of locations of people waiting for an elevator

ride and let $w_i^{origin}$ be the vector of the time that each person has been waiting multiplied by the free capacity of elevator $i$ divided by the total number of people waiting. Finally, let $x_i^m = y_i(i) \otimes \mathbf{1_m}$ be a vector of length $m$ where every entry is the destination of elevator $i$. The local cost function is then defined as

$$\begin{aligned} f_i(y_i) = & k^{elev}(x_i^1 - x_i^{elev})^T w_i^{elev} \mathbb{I}_1(x_i^1 - x_i^{elev}) \\ & + k^{dest}(x_i^d - x_i^{dest})^T w_i^{dest} \mathbb{I}_d(x_i^d - x_i^{dest}) \\ & + k^{origin}(x_i^o - x_i^{origin})^T w_i^{origin} \mathbb{I}_o(x_i^o - x_i^{origin}), \end{aligned} \tag{6}$$

where $k^{elev}$, $k^{dest}$, and $k^{origin}$ are weights on the relative contribution of each factor. They can be used to tune the algorithm for different buildings.

When the controller applies the distributed primal-dual saddle point algorithm (3) with cost function (6), the destination assigned to each elevator will usually not be an integer value. Therefore, rather than navigating to the exact location suggested by the algorithm, the elevator will navigate to the closest floor that either has people waiting to be picked up or is the destination of a passenger of the elevator.

The algorithm is executed in regular time intervals and the destinations are updated accordingly. This means that the destination can be updated before the elevator reaches its previous destination.

### B. Ant Colony Optimization (ACO)

*1) Description:* Ant colony optimization algorithms (ACO) are probabilistic, non-optimal algorithms that determine good paths through a graph [4]. In this case, the graph's nodes are given by passengers calling an elevator and by the elevators themselves. Once passengers have boarded an elevator, their desired destinations become nodes, too. The edges between nodes are weighted and describe the journey from an elevator to a certain call or from one call to another. This is roughly equivalent to the journey between two floors of the building [4].

The operating principle of the ACO algorithm can be described using the ant colony analogy. In this analogy the ants (elevators) each have to find the shortest path through a given graph (consisting of calls and elevators). They do so by following pheromones emitted by ants who have previously passed through the graph. Thus, at each node the ant chooses which edge to take next based on the pheromone intensity of the respective edges [4].

The pheromone intensity distribution used to navigate the ant is determined iteratively by simulating hundreds of ants passing through the graph. Each simulated ant takes the pheromones emitted by previous ants into account and emits pheromones itself depending on the cost of the path it took [4].

This process converges to a pheromone distribution among the edges. Based on this distribution, each call can be assigned to an elevator [4].

*2) Mathematical Formulation:* The mathematical formulation of the ACO algorithm was based on [10] and [11]. The algorithm consists of computing the probability for each elevator to serve a certain floor and updating the pheromone distribution.

*a) Pheromone Update:* The core part of the ant colony optimization algorithm is updating the pheromone distribution. $\tau_{ij}$ denotes the pheromone intensity that node $i$ has from the perspective of ant $j$. After each iteration the pheromones are updated according to the following rule.

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta\tau_{ij}^k \tag{7}$$

Here, $\rho \in (0, 1)$ is a constant coefficient. The incremental pheromone change $\Delta\tau_{ij}^k$ is computed as follows.

$$\Delta\tau_{ij}^k = \begin{cases} C/\Gamma^k & \text{if path } ij \text{ was passed} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$C$ is a constant coefficient and $\Gamma^k$ denotes the global cost of the $k^{\text{th}}$ cycle of the algorithm. Hence, the pheromone $\tau_{ij}$ receives a positive increment if an ant chose to take path $ij$ in this iteration.

*b) Random Proportion Rule:* The random proportion rule is given by [11] as

$$p_{ij}^k = \begin{cases} \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_j (\tau_{ij})^\alpha (\eta_{ij})^\beta} & j \in \text{allowed}_k \\ 0 & j \notin \text{allowed}_k \end{cases} \tag{9}$$

Here, $p_{ij}^k$ is the probability that elevator $i$ chooses node $j$ as its next destination. The index $k$ denotes the current iteration step of the algorithm. $\eta_{ij} = 1/\Gamma_{ij}^k$ where $\Gamma_{ij}^k$ is the *local* cost for elevator $i$ to serve node $j$.

The probability distribution in the final iteration $p_{ij}^K$ is used to dispatch the elevators.

*c) Cost Function:* In the previous paragraph two cost functions were used. The global cost and the local cost.

The local cost is defined as

$$\Gamma_{ij}^k = \Delta l_{ij} + \Delta d_{ij} \tag{10}$$

where $\Delta l_{ij}$ is the number of floors that need to be overcome between $i$ and $j$. $\Delta d_{ij}$ is zero if elevator $i$ does not need to turn around in order to serve $j$ and takes a constant positive value in case a change of heading is needed.

The global cost is merely the sum of all relevant local costs in the respective iteration $k$.

$$\Gamma_k = \sum_{\text{passed } ij} \Gamma_{ij}^k \tag{11}$$

Note, that only paths that were actually passed by an ant are included in the global cost.

*3) Implementation:* Figure 1 shows a flow chart of the implemented the ant colony optimization algorithm [11].

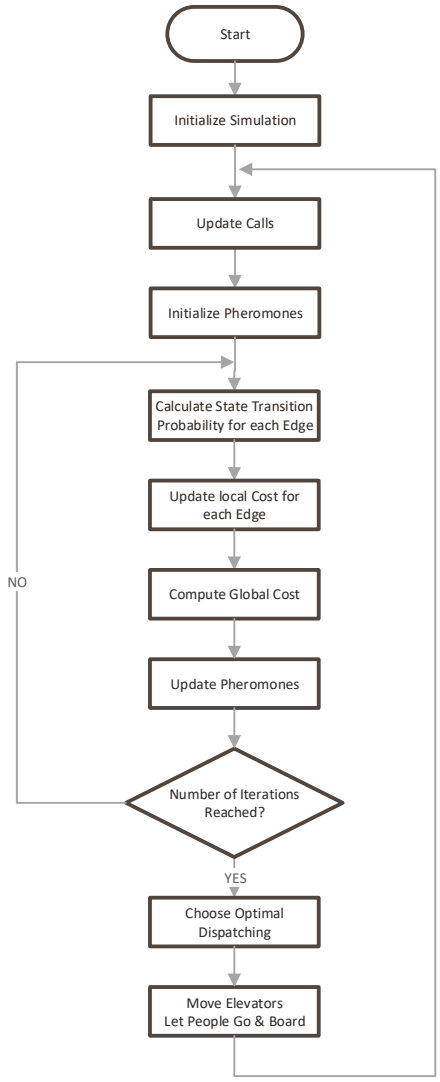


Fig. 1. Ant Colony Optimization Implementation

*4) Distributed Aspects:* While the algorithm is often implemented in a centralized manner, it is worth noting that parts of the ant colony optimization algorithm can be implemented in a distributed fashion.

To stay within the analogy, each ant can decide on its own which path to take relying on the pheromones emitted by previous ants. Since these "previous ants" are simulated this decision can be made by each elevator individually.

To implement this algorithm in a distributed manner, sufficient knowledge about the graph and the local cost function of each elevator need to be shared across all agents. Then, the random proportion rule (9) and the pheromone update (7) can be done by each elevator individually.

In very large building complexes with dozens of elevators this approach might have a practical advantage over a centralized controller. However, cost functions and exceptions have to be handled very carefully to balance the computational complexity with the benefits of a more complex controller.

### C. First In, First Out (FIFO)

*1) Description:* The FIFO algorithm is the most simple elevator dispatch algorithm. Its underlying mechanism is equivalent to the "first come, first serve" principle. When a passenger calls an elevator the call is assigned to the elevator whose current final destination is closest to the caller's floor. Once the passenger boards the elevator and registers his destination, that destination is also simply added to the respective elevator's queue of destinations.

There are obvious improvements that can be made to this algorithm. However, it was implemented in this very simple fashion to serve as a point of comparison for the more advanced algorithms.

*2) Implementation:* Implementing the FIFO algorithm is straightforward. Each elevator has a list of destinations called the "schedule". When a person calls or boards an elevator a new destination is added to the end of the schedule. Destinations are removed from the beginning of the schedule when the elevator has reached the respective floor.

**Algorithm 1** First In, First Out

**for** $i = 1, \cdots, n_{\text{passengers}}$ **do**
  **if** passenger$_i$ is waiting **then**
    $e \leftarrow$ closest elevator to passenger$_i$
    schedule$_e \leftarrow$ [schedule$_e$, origin of passenger$_i$]
  **end if**
  **if** passenger$_i$ has boarded **then**
    $e \leftarrow$ elevator passenger$_i$ is in
    schedule$_e \leftarrow$ [schedule$_e$, destination of passenger$_i$]
  **end if**
**end for**

### D. Classic Elevator Dispatching (CED)

*1) Description:* The "classic elevator dispatching" algorithm (CED) was self-developed based on everyday observations. At its core, this algorithm also applies a "first come, first serve" principle. However, some simple exceptions were added.

CED presents an extension to the FIFO algorithm in that each elevator can add stops between its current position and its current destination at any time. Additionally, the schedule is periodically re-sorted in increasing or decreasing order depending on the direction of travel. This way, people can be picked up and dropped off "on the way" even though they called or boarded the elevator later than someone else.

*2) Implementation:* The implementation of the CED algorithm was based on the FIFO algorithm. The extensions that were made mainly concern the way the "closest" elevator is determined. In contrast to FIFO, CED also considers elevators that pass by the respective passenger instead of only looking at the final destination.

**Algorithm 2** Classic Elevator Dispatching

**for** $i = 1, \cdots, n_{\text{passengers}}$ **do**
  **if** passenger$_i$ is waiting **then**
    $e \leftarrow$ elevator passing passenger$_i$
    **if** $e$ is empty **then**
      $e \leftarrow$ closest elevator to passenger$_i$
    **end if**
    schedule$_e \leftarrow$ [schedule$_e$, origin of passenger$_i$]
  **end if**
  **if** passenger$_i$ has boarded **then**
    $e \leftarrow$ elevator passenger$_i$ is in
    schedule$_e \leftarrow$ [schedule$_e$, destination of passenger$_i$]
  **end if**
**end for**
schedules $\leftarrow$ sort(schedules)

## IV. Simulation Environment

A custom-made simulation environment implemented in MATLAB serves as a testbed for the implemented controllers. Through its modular design, it can easily be adapted for different building architectures and extended to include more controllers. The environment includes a model of passenger flow within a building which was obtained from the literature. The model randomly generates passenger arrivals according to a batch Poisson process as described by Chaosangket et al. The batch size is a function of the time of day and therefore the operation mode of the building [12]. Each generated person is assigned an origin floor and a destination floor which are random variables sampled from a probability distribution as described by Qiu et al. and others [13]–[15]. The discrete probability density function of the origin floor is given by

$$origin(1) = X, \tag{12}$$

$$origin(i) = (Y + Z)\xi_i (i = 2, 3, ..., N), \tag{13}$$

$$\xi_i = \frac{pop(i)}{\sum_{i=2}^{N} pop(i)}, \tag{14}$$

where $pop(i)$ is the number of people on floor $i$, $X$ is the percentage of up-traffic, $Y$ is the percentage of down-traffic and $Z$ is the percentage of inter-floor traffic [13]. The probability density function of the destination floor is conditional on the origin floor. It can be represented as the origin-destination matrix

$$OD = \begin{bmatrix} od(1,1) & od(1,2) & \cdots & od(1,N) \\ od(2,1) & od(2,2) & \cdots & od(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ od(N,1) & od(N,2) & \cdots & od(N,N) \end{bmatrix}, \tag{15}$$

where $od(i, j)$ is the probability that the destination floor is floor $j$ given that the origin floor is floor $i$ [13]. The elements of the matrix depend on the target floor. For up-traffic, they are given by

$$od(1,j) = \begin{cases} 0 & j = 1 \\ \\ \xi_i & j = 2, 3, \cdots, N \end{cases}. \tag{16}$$

For down-traffic, they are given by

$$od(i,1) = \begin{cases} 0 & i = 1 \\ \\ \frac{Y}{Y+Z} & i = 2, 3, \cdots, N \end{cases}. \tag{17}$$

For inter-floor traffic, they are given by

$$od(i,j) = \begin{cases} 0 & i = j \\ \\ \frac{Z\eta_{ij}}{Y+Z} & i \neq j \end{cases}, \tag{18}$$

$$\eta_{ij} = \frac{pop(j)}{\sum_{i=2, i\neq j}^{N} pop(i)}. \tag{19}$$

The model of the passenger traffic can be adapted to different buildings and the different operating modes which occur at different times throughout the day by adjusting the parameters $X$, $Y$, and $Z$ as well as the Poisson intensity $\lambda$. Typically, traffic in large buildings follows a similar pattern: there is a pronounced up-traffic peak in the morning and a less pronounced down-traffic peak in the afternoon. Usually there are two more peaks around noon. Throughout the day, inter-floor traffic plays a larger role [12]–[16]. The batch size also changes throughout the day: in the morning, the batch size is close to one while it is larger around noon [12]. We were guided by the referenced literature in our choice of these tuning parameters.

## V. Simulation Results

To compare the algorithms four different building setups were used (see table I).

TABLE I
Simulation setups

| | **Setup #1** | **Setup #2** | **Setup #3** | **Setup #4** |
|---|---|---|---|---|
| # floors | 5 | 20 | 120 | 120 |
| # elevators | 1 | 10 | 30 | 10 |

These setups were chosen to cover a wide range of buildings varying in size and elevator to floor ratio. Note that the DPDSP algorithm is not compatible with only a single elevator. Hence the lack of results for setup #1. Also, due to the longer run time of the DPDSP algorithm, the averages were taken over a smaller number of runs.

The compared metrics are throughput (in people per hour), average waiting time (in seconds) and average journey time (in seconds).

Note that waiting time and journey time are not being normalized and can therefore only be compared within the

same experimental setup. Furthermore, the simulation environment uses a step size of 10s meaning that each transition from one floor to the next and each on-boarding/off-boarding process takes 10s. Since the main goal is to compare different algorithms this choice was made arbitrarily. With modern elevators reaching speeds of more than 20m/s [17], much smaller waiting and journey times can be expected in a real-world implementation, which also increases the overall throughput. This effect, however, applies to all algorithms. Therefore, a comprehensive comparison between algorithms is admissible even while neglecting the actual elevator velocity.

### A. Throughput

The average throughput of an elevator group is the most important metric. A high throughput minimizes the risk of congestion and generally correlates with the overall performance of the system.

Results for each setup and control algorithm can be found in table II.

TABLE II
THROUGHPUT

| | **Setup #1** | **Setup #2** | **Setup #3** | **Setup #4** |
|---|---|---|---|---|
| FIFO | 8 | 60 | 97 | 27 |
| CED | 44 | 660 | 168 | 55 |
| ACO | 181 | 448 | 375 | 131 |
| DPDSP | – | 653 | 383 | 172 |
| | in people / hour | | | |

The results show that for setup #1 (5 floors, 1 elevator) the ant colony optimization algorithm yields far better results than the simple algorithms. The same statement is true for setup #3 and #4. However, in setup # 2 (20 floors, 10 elevators), the classic elevator dispatching algorithm yields the best throughput among all algorithms. One possible explanation would be that the ACO and DPDSP algorithms are well suited for buildings with a high floor to elevator ratio. In such buildings, which are desirable from an economic perspective, a small number of elevators have to serve a large number of floors. This is the domain of complex elevator group control systems, and our simulation results show that the DPDSP algorithm performs well in such a scenario, outperforming even the ACO algorithm.

The performance gap is even greater in setup #4 than in setup #3. While in setup #3 the ACO algorithm has a 2.2 times larger throughput than CED, the ACO performance is 2.4 times larger than the CED performance in setup #4. Hence, the relative ACO throughput indeed slightly increases when increasing the floor to elevator ratio. The same statement holds for the DPDSP algorithm.

### B. Average Waiting Time

The waiting time is defined as the time measured from the moment a passenger calls an elevator to the instance when the passenger is picked up. Waiting time could be considered part of the overall journey time. However, when deciding which objectives to set, waiting time can play a separate role due to its psychological impact.

TABLE III
AVERAGE WAITING TIME

| | **Setup #1** | **Setup #2** | **Setup #3** | **Setup #4** |
|---|---|---|---|---|
| FIFO | 40 | 124 | 568 | 552 |
| CED | 121 | 112 | 390 | 461 |
| ACO | 122 | 190 | 362 | 374 |
| DPDSP | – | 286 | 137 | 142 |
| | in seconds | | | |

Table III shows the average waiting time across all passengers that have called and boarded an elevator within the simulated time horizon.

Despite its very poor throughput, the FIFO algorithm achieves a fairly low average waiting time, especially in the smaller buildings of setup #1 and #2. This is to be expected because no intermediate stops are made when a passenger is picked up.

Another noteworthy observation is that the ACO and CED algorithms have similar average waiting times in setups #1, #3 and #4. At the same time, the ACO algorithm manages a 2-4 times larger throughput performance.

The fact, that for setup #3 and #4, the DPDSP algorithm has the lowest waiting time while achieving the highest throughput suggests that it might be the most optimal algorithm for large buildings.

### C. Average Journey Time

The average journey time (table IV) is computed among all passengers that have boarded an elevator and successfully reached their destination.

TABLE IV
AVERAGE JOURNEY TIME

| | **Setup #1** | **Setup #2** | **Setup #3** | **Setup #4** |
|---|---|---|---|---|
| FIFO | 93 | 201 | 850 | 1054 |
| CED | 330 | 295 | 749 | 836 |
| ACO | 163 | 306 | 606 | 775 |
| DPDSP | – | 391 | 290 | 322 |
| | in seconds | | | |

Again, for smaller buildings FIFO achieves the lowest journey times since it always takes the direct way. In contrast to waiting times, the journey times of ACO and CED are not as similar. For setups #1, #3 and #4 ACO has significantly lower journey times while maintaining a 2-4 times higher throughput.

The DPDSP algorithm maintains a journey time roughly 50% lower than the ACO algorithm for setups #3 and #4. This solidifies the hypothesis that this algorithm is the best choice for high buildings.

## VI. CONCLUSION

The simulation results clearly show that as soon as elevator resources are sparse, complex elevator dispatch algorithms achieve far better results than simple heuristics.

The self-developed distributed primal-dual saddle point optimization algorithm (DPDSP) achieves a slightly higher throughput than the ant colony optimization algorithm. At the same time it also manages to keep waiting and journey times around 50% lower. This suggests that the new DPDSP algorithm can deliver good control performance and that it could be a candidate for a distributed elevator group control system. However, a broader choice of performance metrics such as maximum waiting and journey time should be examined before making such a definite statement. Nonetheless, the metrics taken in this paper prove functionality of the algorithm and show that its basic performance can keep up with well-established algorithms.

## REFERENCES


[1] National Elevator Industry Inc. Fact sheet. https://nationalelevatorindustry.org/wp-content/uploads/2019/02/Fact-Sheet.pdf, 2019. Accessed: 2021-06-25.

[2] Y. Wu and S. Tanaka. A mixed-integer programming approach to group control of elevator systems with destination hall call registration. In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 26–31, 2020.

[3] Wenjun Mei Florian Dö rfler, Mathias Hudoba de Badyn. Lecture notes in advanced topics in control, May 2021.

[4] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[5] Paul E. Utgoff and Margaret E. Connell. Real-time combinatorial optimization for elevator group dispatching. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 42(1):130–146, 2012.

[6] M.M. Rashid, Nahrul A. Rashid, Alias Farouq, and Md. Ataur Rahman. Design and implementation of fuzzy based controller for modern elevator group. In *2011 IEEE Symposium on Industrial Electronics and Applications*, pages 63–68, 2011.

[7] Luo Fei, Zhao Xiaocui, and Xu Yuge. A new hybrid elevator group control system scheduling strategy based on particle swarm simulated annealing optimization algorithm. In *2010 8th World Congress on Intelligent Control and Automation*, pages 5121–5124, 2010.

[8] Weipeng Liu, Ning Liu, Hexu Sun, Guansheng Xing, Yan Dong, and Haiyong Chen. Dispatching algorithm design for elevator group control system with q-learning based on a recurrent neural network. In *2013 25th Chinese Control and Decision Conference (CCDC)*, pages 3397–3402, 2013.

[9] Hoon Heo, Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. Elevator group supervisory control system with destination floor guidance system using genetic network programming. In *2006 SICE-ICASE International Joint Conference*, pages 5489–5493, 2006.

[10] Yu Le, Yang Shifeng, Li Huanhuan, Lv Zhicheng, and Hao Xiaobing. Research on elevator group optimal dispatch based on ant colony algorithm. In *2020 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pages 99–102, 2020.

[11] Jing-long Zhang, Jie Tang, Qun Zong, and Jun-fang Li. Energy-saving scheduling strategy for elevator group control system based on ant colony optimization. In *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, pages 37–40, 2010.

[12] Nathaporn Chaosangket, Pruk Sasithong, Sanika K. Wijayasekara, Widhyakorn Asdornwised, Lunchakorn Wuttisittikulkij, Pisit Vanichchanunt, and Muhammad Saadi. A simulation tool for vertical transportation systems using python. In *2018 5th International Conference on Business and Industrial Research (ICBIR)*, pages 270–275, 2018.

[13] JianDong Qiu and ZhaoYuan Jiang. The research and simulation on the elevator group control system scheduling algorithm. In *2011 International Conference on Electrical and Control Engineering*, pages 1346–1349, 2011.

[14] Lijun Fu and Tiegang Hao. Analysis and simulation of passenger flow model of elevator group control system. In *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 2353–2356, 2012.

[15] Pingli Wang, Gongxuan Zhang, and Ling Wang. Simulation of customers-flow model based-on elevators group control technique. In *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*, volume 1, pages 568–571, 2006.

[16] ChangBum Kim, K.A. Seong, Hyung Lee-Kwang, and J.O. Kim. Design and implementation of a fuzzy elevator group control system. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(3):277–287, 1998.

[17] Jane Sit Jenni Marsh. Shanghai tower picks up 3 guinness world records including fastest elevator. https://edition.cnn.com/style/article/worlds-fastest-tower/index.html, 2017. Accessed: 2021-07-02.