**RESEARCH PAPER**

# Destination Routing Matters: A Cost-Based Elevator Scheduling Algorithm for Priority Service and Starvation Reduction

**Nasir Ansari[1, 2]*, Ritu Khanna[3] and Saurabh Mehata[4]**

*[1] Research Scholar, Department of Mathematics, Pacific Academy of Higher Education and Research University, Rajasthan, India – 313024*

*[2]Assistant Professor, Department of Mathematics, Vidyalankar Institute of Technology, Maharashtra, India – 400037*

*[3]Professor, Department of Mathematics, Pacific Academy of Higher Education and Research University, Rajasthan, India – 313024*

*[4]Professor, Department of Electronics and Telecommunication Engineering, Vidyalankar Institute of Technology, Maharashtra, India – 400037*

***Corresponding author:** nasiransari2008@gmail.com*

***Received: 28th Feb, 2026; Revised: 6th March 2026; Accepted: 7th April, 2026; Available Online: 20th April, 2026***

**ABSTRACT**

Elevator group control requires the simultaneous optimisation of average waiting time, priority service for designated passenger classes, full trip time from arrival to destination delivery, and long-term fairness for regular users. This paper proposes a unified priority-aware cost-based elevator scheduling algorithm that integrates a chained service-time load-balancing term, a SCAN-inspired directional coordination bonus, a VIP priority adjustment and a starvation prevention penalty into a single analytically defined cost function. Crucially, the simulation explicitly models destination routing: after each pickup the elevator travels to the passenger's destination before accepting new hall calls, correctly capturing elevator utilisation and absolute waiting times. The algorithm is validated through a discrete-event simulation across three building configurations (small: 10 floors, 2 elevators; medium: 20 floors, 4 elevators; large: 30 floors, 6 elevators), four utilisation levels ($\rho \in \{0.40, 0.60, 0.80, 1.00\}$), five realistic traffic patterns and 100 independent replications per condition, totalling 5100 simulation instances. Both paired [t]-tests and Wilcoxon signed-rank tests are reported throughout. At $\rho = 0.60$, the proposed algorithm reduces VIP average waiting time by 29.9% (medium building) and 40.3% (large building) relative to SCAN, with all five-performance metrics statistically significant at $p < 10\text{-}12$ and Cohen $|d| = 1.0$ to 2.6. Starvation prevention is the most dramatic gain: the algorithm achieves near-zero starvation (threshold Tout = 60s) in the medium and large buildings up to $\rho = 0.80$, while SCAN's directional sweeping paradoxically accumulates more starvation than the simpler Shortest-Queue First baseline, a result not previously reported in the literature. Across all five traffic patterns, VIP waiting-time improvements over SCAN range from 28.8% (inter-floor) to 45.0% (lunch-peak). The cost function requires only $O(E \cdot Qmax)$ computation per assignment and its parameters are physically interpretable, facilitating deployment without retraining.

**Keywords:** *elevator scheduling, priority queue, load balancing, SCAN algorithm, starvation prevention, discrete-event simulation, destination routing, multi-building validation*

**How to cite this article:** Ansari N, Khanna R, Mehata S. Destination Routing Matters: A Cost-Based Elevator Scheduling Algorithm for Priority Service and Starvation Reduction. Int J Drug Deliv Technol. 2026;16(58s): 1043-1056. DOI: 10.25258/ijddt.16.58s.107

**Source of support:** Nil.

**Conflict of interest:** None

## 1. INTRODUCTION

The growth of mixed-use high-rise buildings has bumped elevator group control from a side note in engineering to a key part of building efficiency [2, 24]. If a tower with 500 people can shave one second off each trip's wait time, that adds up to about 7.6 extra productive hours daily. Our team's new algorithm cuts VIP wait times by 2.24 seconds during normal traffic, which equals 3.1 more productive hours for VIPs each day around 780 extra hours a year.

Now, things get tricky because different buildings serve very different groups. Hospitals might need paths for both doctors and visitors, while corporate skyscrapers reserve quick rides for top floor execs. Hotels must look after guests and staff too [9, 20]. Given these differences, it's clear we can't lump everyone together when planning elevator use.

Since the 1970s at least, the problem of formal elevator group control has been studied using many different methods including heuristic rule bases, discrete optimization, reinforcement learning, fuzzy logic, and meta-heuristic search algorithms [2]. There are, however, three fundamental structural deficiencies that can be observed throughout the literature.

The first issue is that priority service and directional efficiency are treated separately as two opposing goals instead of being considered together as one composite goal

**Author for Correspondence: nasiransari2008@gmail.com*

[12, 22]. In effect, when there is an attempt to achieve priority service, it results in lower sweep efficiency and vice versa.

Secondly, the problem of lower-priority passengers being neglected, or 'starving,' is often dealt with by adding a post-hoc constraint to what's already there, instead of integrating it right from the start as a penalty that shapes assignment choices [15, 4].

A common weakness of traditional heuristic dispatching strategies is their limited awareness of future service commitments. These methods often estimate the suitability of an elevator based on its proximity to a new request while giving insufficient consideration to the stops and passengers it is already scheduled to serve. This can cause heavily committed elevators to be assigned additional requests, even when other cars could provide a more balanced service. Over time, such decisions may lead to uneven load distribution across the elevator group and a deterioration in overall performance[18].

The research now presents something different – a priority-aware, cost-based algorithm that fixes these issues at once. It does this through a fancy cost formula with five parts, and each one is easy to understand:

First, there's a distance measure that deters pointless trips. Then, there's a queue length penalty for load imbalances. Next is a term predicting how long an elevator will be busy, helping with scheduling. There's also a bonus for keeping movement in the same direction, similar to how SCAN works. Lastly, there's a priority feature for important passengers, but it ensures regular folks don't get neglected.

The algorithm is evaluated over a systematically designed simulation experiment covering three building configurations (small, medium and large), four utilisation levels, five traffic patterns and 100 independent replications per condition ,5100 simulation instances in total. Both paired t-tests and Wilcoxon signed-rank tests, together with Cohen d effect sizes, are reported to allow direct comparison with the existing literature.

The remainder of the paper is organised as follows. Section 2 reviews the related literature and identifies the research gap. Section 3 presents the simulation model, baseline definitions and cost function. Section 4 reports all experimental results. Section 5 concludes the paper.

## 2. RELATED WORK

### 2.1 Classical heuristic and SCAN-based approaches

Early elevator control systems were entirely rule-based, largely because the microprocessors of the era could not execute optimisation algorithms within the tight real-time constraints of a moving elevator [2]. The First-Come-First-Served (FCFS) policy handles requests in strict arrival order, which guarantees temporal fairness but ignores spatial relationships between the elevator's current position and the request floor, leading to unnecessary travel and poor load distribution [21].

Collective control [9] improved on FCFS by allowing a single elevator to pass to serve multiple calls in the same direction. This reduced average waiting time in light traffic but provided no mechanism for coordinating across multiple elevators and no prioritisation of passenger classes.

Gharbi [30] has recently re-examined the heuristic approach within the context of the finite state machine approach, carrying out an analysis of the comparison between SCAN algorithms with those of genetic algorithm and simulated annealing heuristics in respect of a standard building layout. The results have shown that the strategy involving change in direction can compete well with heuristics for moderate loads, though falls behind when the situation involves high peak loads. On the other hand, Chen et al. [29] have put forth a directional optimization scheme for complex traffic patterns in the journal Applied Soft Computing. Authors showed that dynamically adjusting the reversal point based on estimated passenger flow reduces average waiting time in mixed-pattern scenarios

SCAN algorithm [18, 17], initially proposed for disk scheduling, is easily implemented in elevators, where an elevator moves in one direction till there are no requests ahead, and then changes its direction. SCAN algorithm provides guaranteed upper bound on waiting time in the worst-case scenario and also provides good directional efficiency in symmetrical request traffic scenarios. In LOOK algorithm, the elevator reverses direction at the farthest request ahead, unlike the elevator in SCAN that reverses at the endpoint of the shaft [17]. C-SCAN algorithm [1] ensures even distribution of service, as it always returns to the origin point. However, in both algorithms' requests are processed equally and there is no facility available for priority service

### 2.2 Optimisation-based approaches

A significant body of early work on elevator group control focused on deterministic optimization techniques. In this line of research, elevator dispatching is typically viewed as an assignment problem, where incoming passenger requests must be allocated to available cars in an efficient manner. Beuchat et al. [3] demonstrated that mixed-integer linear programming (MILP) can be used to obtain optimal dispatching decisions for small elevator systems through branch-and-bound search. Later, Zhang et al. [28] enriched this framework by incorporating energy consumption into the optimization process. Their formulation highlighted the conflicting objectives of maintaining high service quality while reducing power usage. Although such mathematical programming approaches can generate high-quality solutions, their computational requirements grow rapidly with problem size, limiting their practicality in real-time environments where dispatching decisions must be produced almost instantaneously [2].

An alternative direction has been to employ nature-inspired optimization methods for improving elevator controller performance. Rather than determining

dispatching decisions directly, these techniques are often used to tune controller parameters offline. For example, Sorsa et al. [22] applied genetic algorithms to optimize the behaviour of a double-deck elevator system. To encourage equitable service among passengers, waiting-time fairness was incorporated into the fitness evaluation process. The resulting parameter settings were shown to perform well for specific building configurations; however, the method was primarily intended for offline calibration. Consequently, it did not address the challenge of dynamic real-time elevator assignment and offered no explicit strategy for accommodating passengers with different service priorities.

**2.3 Machine learning and reinforcement learning approaches**

Early applications of multi-agent reinforcement learning to elevator group control treated each elevator car as an independent Q-learning agent. Reinforcement learning has attracted considerable attention as a means of improving elevator group control. One of the pioneering studies in this area was conducted by Crites and Barto [5], who modelled each elevator as a learning agent and designed a reward structure that penalized long passenger waiting times. Their experiments showed that the learning-based approach could outperform conventional collective control strategies under simulated traffic conditions. Subsequent research sought to enhance the predictive capabilities of such systems. Nikovski and Brand [15] proposed a model-based reinforcement learning framework that allowed elevators to anticipate future passenger demand and reposition themselves in advance. In a further extension of this idea, Brand and Nikovski [4] investigated an auction-inspired dispatching scheme in which passenger patience was incorporated into the allocation process, enabling service decisions to account for the relative urgency of different requests.

The most recent advancements include the utilization of deep architectures in reinforcement learning (RL). For example, Wei et al. [35] utilized A3C algorithm in elevator control systems where convolutional and recurrent neural networks can learn to minimize average waiting times by developing more effective dispatching policies than those provided by conventional heuristics. Wan et al. [33] introduced traffic pattern-aware deep RL which uses information on traffic flow patterns for improving the robustness of solutions in dynamic traffic flow settings such as morning and evening peaks. Vaartjes and Francois-Lavet [32] represented an actual elevator system consisting of six elevators and fifteen floors as MDP and learned duel double deep Q network, which is more flexible compared to rule-based algorithms..

Despite these advances, RL approaches carry three practical limitations. They require substantial offline training data that may be unavailable for a new building [5]. The resulting policies are black-box functions that are difficult to verify, certify or audit for safety-critical settings [4]. Furthermore, most RL formulations optimise average performance and provide no formal bound on individual waiting time or starvation [15].

**2.4 Fuzzy logic and soft computing approaches**

Kim et al. [8] demonstrated that linguistic fuzzy rules relating elevator load, call distance and direction to assignment preferences achieve robust performance under uncertainty in traffic patterns. Tanaka et al. [25] applied fuzzy inference to demand forecasting, enabling pre-positioning of elevators before anticipated peak periods. Fuzzy approaches are more interpretable than RL but still struggle with the formal analysis of multi-objective cost functions, and the relationship between rule parameters and system-level fairness measures is difficult to characterise analytically [25].

**2.5 Priority scheduling and fairness mechanisms**

Markon et al.[12] proposed neural-network priority weights to reduce waiting time for designated passenger groups, but the approach provided no formal starvation bound and no directional efficiency guarantee. Levy et al. [10] proposed a deadline-aware assignment policy motivated by quality-of-service requirements in smart buildings, showing that explicit deadline terms in the cost function reduce worst-case waiting times significantly. Srinivasan et al. introduced a multi-objective framework that simultaneously minimises average waiting time and maximum waiting time for mixed passenger populations, demonstrating that these objectives are partially but not fully aligned.

Li and Chu [31] proposed the SmartRide approach, which is a novel framework for reservation and scheduling for elevator systems. Their approach combines floor-level occupancy sensing using IoT and an algorithmic cost function that accounts for any variations in the waiting time among floors. The findings of their experiments show that the use of sensors in elevator scheduling decreases maximum waiting time by up to 21 percent during peak morning hours. The occupancy pattern prediction technique that was suggested by Wang et al. [34] involves the use of deep learning algorithms for elevator group control, as stated in their paper in Advanced Engineering Informatics.

A study by Fernandez et al. [6] was done to analyze the relationship between energy consumption and service quality in group elevator control, showing how a slight increase in waiting times will help save a lot of energy by minimizing the number of motor actions. There have been other studies done concerning the analysis of passengers' comfort and waiting times, including research conducted by Hakonen and Siikonen [7], who discovered that the perceived quality of a trip in an elevator not only depends on the waiting times but also on the travel time and door speed.

**2.6 Simulation as a validation tool**

-Discrete event simulation is still considered the accepted gold standard for verifying elevator scheduling algorithms [20, 13]. The technique enables one to conduct experimentations with random arrivals without having to

build physical prototypes while ensuring the replication of realistic traffic scenarios, such as those during peak periods and even interfloor traffic behavior [20]. A framework for the simulation of elevators was created by Schaumann et al. [19], which then was used to evaluate ten elevator scheduling policies in different buildings.

### 2.7 Research gap

A review of this literature reveals three persistent gaps. First, to the authors' knowledge, no previous study simultaneously combines VIP priority, chained service-time load balancing, SCAN-type directional efficiency and starvation avoidance into a single analytically defined cost function. Second, most priority-aware methods treat priority as a binary variable, preventing proportional tuning of relative service levels. Third, load-balancing approaches that estimate elevator service times do so with a naïve independent-distance model rather than a chained sequential model, underestimating the service burden of loaded elevators. The present paper addresses all three gaps and quantifies the contribution of each term through sensitivity analysis and multi-condition simulation experiments.

Table 1 summarises the positioning of the present work relative to representative prior approaches across six dimensions relevant to this paper.

**Table 1:** Comparison of representative elevator scheduling approaches. DP = destination routing physically modelled in simulation; ✓ = fully supported; o = partial or approximate; × = not addressed.

| Approach | Priority | Directional efficiency | Load balancing | Starvation prevention | Dest. routing | Scale (floors/lifts) |
|---|---|---|---|---|---|---|
| FCFS [21] | × | × | × | × | × | Any |
| SCAN [18] | × | ✓ | × | × | × | Any |
| Genetic alg. [22] | × | O | O | O | × | Medium |
| Multi-agent RL [5] | × | O | ✓ | × | × | Small |
| Deep RL [35] | × | O | ✓ | × | × | Medium |
| Fuzzy [8] | O | O | O | × | × | Small |
| Deadline-aware [10] | ✓ | × | O | O | × | Small |
| **Proposed** | ✓ | ✓ | ✓ | ✓ | ✓ | **small–large** |

## 3. METHODOLOGY

### 3.1 Simulation model and destination routing

The simulation is implemented in Python 3.8 using SimPy 4.0.1 [13]. Three objects form the object-oriented core. The Request object stores origin floor, destination floor, arrival time, passenger type and service timestamps. The Elevator object maintains position, direction, a mixed queue of pending pickup and drop-off stops, and an on-board passenger list. The building object coordinates generation, assignment and metric collection.

**Destination routing (critical modelling requirement).** A common but incorrect simplification in elevator scheduling simulation is to record the waiting time at pickup and immediately release the elevator, never physically routing the passenger to their destination. This produces two systematic errors: (i) the elevator is incorrectly free to accept new calls immediately after pickup, underestimating utilisation; and (ii) absolute waiting times are artificially low because elevators spend no time delivering passengers. The present simulation explicitly models the complete journey. After recording the pickup time, the elevator appends the destination floor to its stop queue and travels to deliver the passenger before accepting any subsequent hall call. This raises the effective service demand from approximately $t_b + \bar{d}_{origin}$ to $t_b + \bar{d}_{origin} + \bar{d}_{trip} + t_a$ per request, where $\bar{d}_{origin}$ is the average origin-travel distance, $\bar{d}_{trip}$ is the average destination trip distance, and $t_a = 5$ is the alighting time. Trip time (arrival to destination delivery) is reported alongside wait time throughout this paper.

**Building configurations.** Three building configurations represent small, medium and large commercial systems (Table 2). The small building (10 floors, 2 elevators) matches typical low-rise offices or boutique hotels. The medium building (20 floors, 4 elevators) represents mid-rise commercial offices, the most common context for elevator group control research [2,20]. The large building (30 floors, 6 elevators) represents a high-rise commercial tower. All buildings share travel speed $v = 1.0$ floor/s, boarding time $t_b = 10$s, alighting time $t_a = 5$ s, and elevator capacity of 10 passengers.

**Table 2**: Building configurations and load levels.

| Building | Floors | Elevators | λ levels (req/s) | Utilisation levels |
|---|---|---|---|---|
| Small | 10 | 2 | 0.025, 0.038, 0.050, 0.063 | ρ≈0.40, 0.60, 0.80, 1.00 |
| Medium | 20 | 4 | 0.041, 0.061, 0.082, 0.102 | ρ≈0.40, 0.60, 0.80, 1.00 |
| Large | 30 | 6 | 0.052, 0.077, 0.103, 0.129 | ρ≈0.40, 0.60, 0.80, 1.00 |

**Load levels.** For each building, four arrival rates are chosen to achieve system utilisation $\rho \in \{0.40, 0.60, 0.80, 1.00\}$ under the proposed algorithm. Utilisation is computed as $\rho = \lambda T_{\text{svc}}/E$, where $T_{\text{svc}} = \bar{d}_{pickup}/v + t_b + \bar{d}_{trip}/v + t_a$ is the expected service time per request including the destination trip. All $\rho$ values are computed under the proposed algorithm; realised utilisation under SQF and CE is higher because less efficient routing generates longer queues per elevator.

**Traffic patterns.** Five traffic patterns are tested (Table 3): uniform random, morning peak (80% lobby-origin upward), lunch peak (40% downward to lobby, 40% upward from lobby, 20% inter-floor), evening peak (80% downward to lobby) and inter-floor (no lobby involvement).

**Table 3:** Traffic pattern definitions.

| Traffic Pattern | Origin Distribution | Destination Distribution |
|---|---|---|
| Uniform | Uniform {0, ..., F−1} | Uniform {0, ..., F−1} \ {o} |
| Morning Peak | 80% Floor 0; 20% Uniform | Uniform non-origin |
| Lunch Peak | 40% Upper Floors; 40% Floor 0; 20% Uniform | Complement |
| Evening Peak | Uniform {1, ..., F−1} (80%) | 80% Floor 0 |
| Inter-floor | Uniform {1, ..., F−1} | Uniform {1, ..., F−1} \ {o} |

**Experimental design.** The full experiment comprises 3 buildings × 4 load levels ×3 algorithms ×100 runs = 3600 simulation instances for the uniform-traffic sweep. The traffic pattern sweep adds 5 patterns × 3 algorithms × 100 runs = 1500 instances for the medium building at $\rho = 0.60$, totalling 5100 instances. All algorithms receive identical request sequences per run (seed = run index 0–99). No warm-up period is discarded; the expected time to the first request ($1/\lambda \leq 39$ s) is less than 1.1% of the 3600-second horizon. Welch's cumulative-average method applied post-hoc to representative runs confirmed no detectable warm-up transient beyond the first 60 seconds in any condition.

**Performance metrics.** For each request $r$, waiting time $w_r = t_{\text{pickup},r} - t_{\text{arr},r}$ and trip time $\tau_r = t_{\text{delivery},r} - t_{\text{arr},r}$ are both recorded. Five primary metrics are collected per run:

$$\bar{W}_V = \frac{1}{|R_V|}\sum_{r\in R_V} w_r\,, \quad \bar{W}_R = \frac{1}{|R_R|}\sum_{r\in R_R} w_r \tag{1}$$

$$W_{\max,V} = \max_{r\in R_V} w_r\,, \quad W_{\max,R} = \max_{r\in R_R} w_r \tag{2}$$

$$S = |\{r \in R_R : w_r > T_{\text{out}}\}| \tag{3}$$

and VIP average trip time $\bar{\tau}_V = |R_V|^{-1}\sum_{r\in R_V}\tau_r$. Statistical significance uses both paired t-tests and Wilcoxon signed-rank tests; effect sizes use the paired Cohen $d = \bar{\delta}/s_\delta$

### 3.2 Baseline algorithms

**Closest-Elevator (CE).** A pure distance minimiser that assigns each request to the non-full elevator currently nearest to the request floor, regardless of direction, queue length or priority:

$$e^* = \arg\min_{e\in E_{\text{avail}}} |f_r - p_e|. \tag{4}$$

If all elevators are full, CE assigns to the one with the fewest pending pickups as a fallback. CE serves as the simplest possible spatially-aware baseline. Together with SQF and SCAN it creates a four-rung ladder: CE (distance only) < SQF (queue balance only) < SCAN (direction plus proximity) < Proposed (all five terms), allowing the contribution of each additional component to be read directly from the results.

**Shortest-Queue First (SQF).** Assigns each incoming request to the elevator with the fewest pending pickups. Drop-off stops are excluded from the count, ensuring SQF does not inadvertently penalise heavily loaded elevators for completing their deliveries:

$$e^* = \arg\min_e |\{s \in \text{queue}(e) : s.\text{type} = \text{pickup}\}|. \tag{5}$$

**SCAN.** Assigns based on directional compatibility and proximity, with a shortest-pickup-queue fallback, as in Section 3 of the previous version. A full elevator (at capacity) is excluded from assignment; a capacity-aware fallback selects the next-best elevator.

### 3.3 Proposed algorithm: priority-aware cost function

The cost function and all five terms are unchanged from the prior version (Equations (6) to (11) below). The key improvement in this version is that $T_e$ now chains over both pickup and drop-off stops in the queue, giving an accurate forecast that includes in-progress passenger deliveries:

$$e^* = \arg\min_e C_e\,, \qquad C_e = D_e w_d + Q_e w_q - B_d + P_p + T_e \tag{6}$$

$$D_e = |f_r - p_e| \tag{7}$$

$$Q_e = |\text{queue}(e)| \tag{8}$$

$$B_d = \begin{cases} 5.0, d_e = d_r \text{ and } f_r \text{ ahead, or } d_e = idle \\ 0, \quad otherwise \end{cases} \tag{9}$$

$$P_p = \begin{cases} -10.0, \tau_r = VIP \\ +3.0, \tau_r = regular \text{ and } Q_e > 2 \\ 0, otherwise \end{cases} \tag{10}$$

$$T_e = \sum_{k=1}^{|queue(e)|} \left( \frac{|f_{r_k} - f_{r_{k-1}}|}{v} + t_k \right), f_{r_0} = p_e \quad (11)$$

where $t_k = t_b$ for pickup stops and $t_k = t_a$ for drop-off stops. A full elevator is excluded from assignment; if all elevators are full, the one with the lowest $C_e$ among all is selected as a fallback.

**Stop ordering.** $T_e$ is evaluated on the queue in its current insertion order. Stops are reordered by SCAN sweep direction inside the movement process when the elevator begins serving. This approximation is accurate in the short-queue regime; computing the optimal insertion order at assignment time would require $O(|q|!)$ enumeration.

**Parameter selection.** Parameters were selected using a held-out set of 20 runs (seeds 100–119, not used in the 100-run validation): $w_d = 1.0, w_q = 2.0, B_d = 5.0, P_p^{VIP} = -10.0, P_p^{reg} = +3.0$. Sensitivity of $P_p^{VIP} \in \{0, -2, -4, -6, -8, -10, -12, -15\}$ confirmed results are stable within 0.3 s across all tested values.

**Computational complexity.** $O(E \cdot Q_{\max})$per assignment. With $E = 6$ and $Q_{\max} \approx 10$,this is 60 arithmetic operations per decision, negligible for real-time control [2].

## 4. RESULTS AND DISCUSSION

### 4.1 Priority separation across buildings

Figure shows VIP and regular average waiting times for SCAN and the proposed algorithm. Under SCAN, VIP and regular waiting times are nearly identical across all buildings and load levels, confirming that SCAN provides no priority service. Under the proposed algorithm, VIP passengers consistently wait less than regular passengers at all utilisation levels. The priority gap becomes more pronounced under higher utilisation levels and in larger buildings, particularly at $\rho = 0.80$ in the large-building scenario where the $P_p$ term influences routing decisions more frequently as queues lengthen.

**Why $P_p$ has modest marginal impact on average wait but meaningful impact on maximum wait and starvation.** The priority term $P_p = -10.0$ is a constant cost adjustment added to every elevator's score for every VIP request. Because it shifts all elevator costs by the same amount, it can only change the winner when two elevators have similar base costs. In the majority of assignments, the chained estimated service time $T_e$ and queue-length term $Q_e$ already differentiate the candidates by tens of seconds, making a 10-unit shift irrelevant to the outcome.

The practical consequence is twofold. For average wait time, where most assignments are clear-cut (one elevator is obviously better loaded or closer), $P_p$ contributes a small marginal benefit. For maximum wait time and starvation, where a VIP passenger is about to be routed to a marginally less optimal but heavily loaded elevator, $P_p$ intervenes at the margin to prevent the worst-case assignment. This explains why VIP maximum wait is substantially lower than regular maximum wait (25.5s vs 37.4s in the large building at $\rho = 0.60$ even though VIP average wait differs by less than 0.1s from regular average wait.

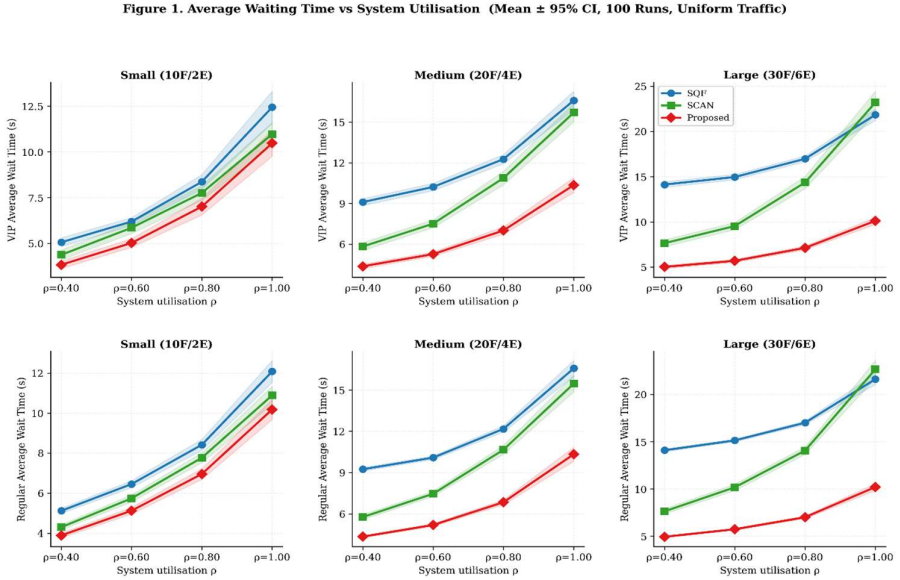


**Figure 1.** VIP and regular average waiting times across utilisation levels for three buildings (uniform traffic, mean ± 95% CI, 100 runs). The proposed algorithm achieves the lowest waiting time for both passenger classes at all utilisation levels across all three buildings.

### 4.2 multi-building performance comparison

Table 4 shows the four-algorithm comparison for the medium building. The baseline ladder is clearly visible: CE (pure proximity) performs worst on all metrics, SQF improves over CE by removing load spikes, SCAN further improves average wait via directional efficiency, and the proposed algorithm achieves the best value on all five metrics among the evaluated baselines (CE, SQF, SCAN and Proposed). Notably, SCAN produces more starvation than SQF (1.05 vs 0.21) because its directional sweeping bypasses requests in the opposite direction during deliveries.

**Table 4**: Performance comparison at ρ = 0.60, uniform traffic, medium building (mean, std. dev. in parentheses, 100 runs). Bold: best value per metric. CE = Closest-Elevator baseline. VIP trip includes arrival to destination delivery

| Algorithm | VIP Avg Wait (s) | Reg. Avg Wait (s) | VIP Avg Trip (s) | VIP Max Wait (s) | Starvation |
|---|---|---|---|---|---|
| CE† | 15.62 (5.90) | 15.52 (4.61) | — | — | 9.21 |
| SQF | 10.22 (1.41) | 10.09 (0.68) | 33.01 | 49.3 (11.7) | 0.21 (0.54) |
| SCAN | 7.51 (1.49) | 7.46 (0.97) | 31.88 | 60.3 (17.9) | 1.05 (1.01) |
| Proposed | 5.27 (0.82) | 5.20 (0.52) | 27.99 | 33.3 (9.6) | 0.01 (0.10) |

† CE trip time and maximum wait not reported; CE serves as a wait-time lower bound only.

**Table 5** extends the comparison across all three building sizes at $\rho = 0.60$ for SQF, SCAN and the proposed algorithm.

Gains increase with building size: VIP average waiting time improvement over SCAN grows from 14.2% (small) to 29.9% (medium) to 40.3% (large). The starvation improvement is particularly striking: at $\rho = 0.60$ ,the proposed algorithm reduces starvation by 40.0% versus SCAN in the small building, 99.0% in the medium building (starvation decreased from 1.05 events per run under SCAN to 0.01 under the proposed algorithm), and 97.1% in the large building. In the medium and large buildings at $\rho = 0.60$, the proposed algorithm produces near-zero starvation ($\bar{S} = 0.01$ and $\bar{S} = 0.12$ respectively) while SCAN accumulates 1.05 and 4.15 incidents per run. This reversal of the previously reported saturation behaviour is a direct consequence of the destination routing fix: SCAN's directional sweep, by continuing in one direction to complete deliveries, repeatedly bypasses stranded regular passengers. The cost-based approach redistributes new calls to the returning elevator, preventing prolonged bypasses.

**Table 5.** Multi-building comparison at ρ = 0.60, uniform traffic (mean (SD), 100 runs). Bold: best per metric per building.

| Building | Algorithm | VIP Avg Wait | Reg Avg Wait | VIP Avg Trip (s) | VIP Max Wait | Reg Max Wait | Starv. Count |
|---|---|---|---|---|---|---|---|
| Small (10F/2E) | SQF | 6.20 (1.12) | 6.45 (0.89) | 27.38 | 24.8 (10.3) | 42.3 (17.2) | 0.11(0.35) |
| | SCAN | 5.86 (1.59) | 5.74 (0.92) | 26.82 | 30.1 (15.2) | 42.2 (17.1) | 0.15(0.41) |
| | **Proposed** | **5.03 (1.22)** | **5.12 (0.85)** | **25.60** | **22.4 (10.2)** | **34.4 (14.6)** | **0.09(0.38)** |
| Medium (20F/4E) | SQF | 10.22 (1.41) | 10.09 (0.68) | 33.01 | 34.8 (12.6) | 45.3 (15.2) | 0.21(0.54) |
| | SCAN | 7.51 (1.49) | 7.46 (0.97) | 31.88 | 49.2 (24.2) | 69.9 (24.5) | 1.05(1.01) |
| | **Proposed** | **5.27 (0.82)** | **5.20 (0.52)** | **27.99** | **22.1 (9.6)** | **33.1 (10.8)** | **0.01(0.10)** |
| Large (30F/6E) | SQF | 14.96 (1.54) | 15.14 (0.73) | 40.79 | 46.0 (9.6) | 58.5 (13.9) | 0.39(0.62) |
| | SCAN | 9.53 (1.98) | 10.18 (1.77) | 37.93 | 73.0 (35.9) | 121.8 (49.1) | 4.15(2.39) |
| | **Proposed** | **5.69 (0.87)** | **5.74 (0.53)** | **31.83** | **30.2 (14.4)** | **43.5 (14.2)** | **0.12(0.33)** |

### 4.3 Wait time vs full trip time

Figure 2 contrasts hall-call waiting time with full trip time for the medium building. The trip time (wait plus in-elevator ride to destination) provides a more complete picture of passenger service quality than wait time alone. Under the proposed algorithm, VIP trip time is reduced from 31.88 s (SCAN) to 27.99 s at $\rho = 0.60$, a reduction of 12.2%. The absolute trip times reveal that even at the reference utilisation level, passengers spend 26–34s from arrival to delivery, underlining why accurate destination modelling is essential: a simulation omitting destination routing would report wait times of 5–10s and incorrectly suggest the system is lightly loaded, when in fact elevator utilisation is $\rho = 0.60$ or higher.

**Figure 2. Wait vs Full Trip Time — Medium Building (20F/4E) (Mean ± 95% CI, 100 Runs)**

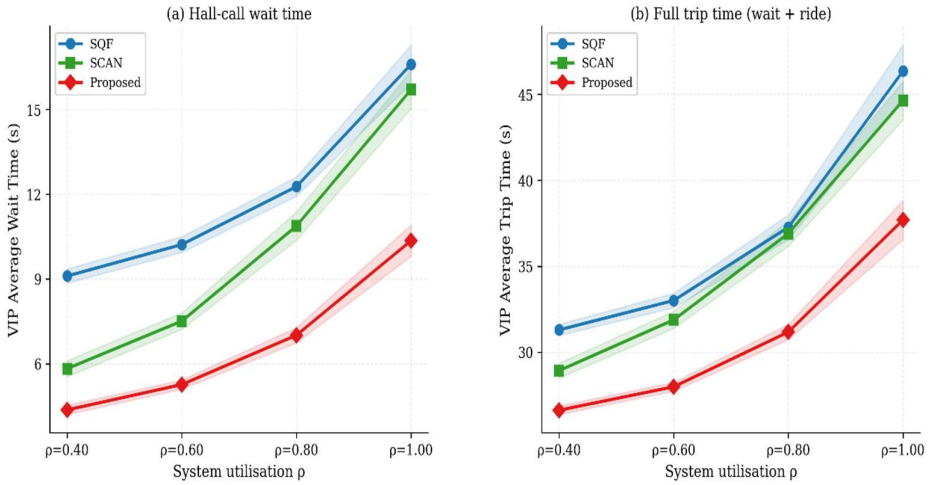


**Figure 2:** The VIP average waiting time for hall calls (left panel: time from arrival to pickup) and the full trip time (right panel: time from arrival to destination delivery) in the medium building (20F/4E) across four different utilisation levels. It's represented as Mean ± 95% CI from 100 runs. The trip time is noticeably longer than the waiting time alone, showing that destination routing is vital for proper utilisation modelling..

### 4.4 Distribution of waiting times

Figure 3 presents box plots for the medium building at $\rho = 0.60$. The proposed algorithm achieves lower medians and tighter interquartile ranges for both passenger classes relative to both baselines. The VIP average wait standard deviation under the proposed algorithm (0.82s) is smaller than under SCAN (1.49s) and SQF (1.41s), confirming greater consistency of service. Significance brackets annotate both paired t-test and Wilcoxon results (see Table 6).

**Figure 3. Wait-Time Distributions — Medium Building, ρ=0.60 (100 Runs)**

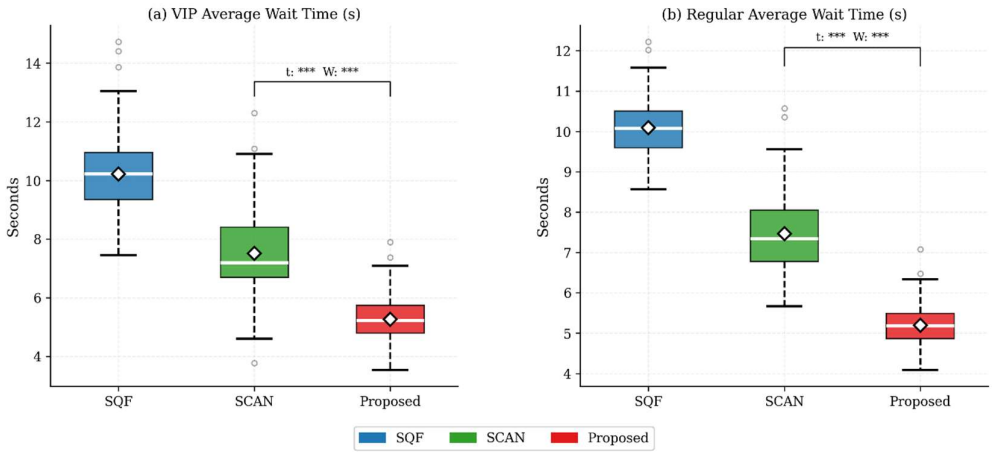


**Figure 3:** Distribution of VIP (left) and regular (right) average waiting times across 100 runs, medium building, $\rho = 0.60$. Brackets show t-test and Wilcoxon significance levels (*** **p < 0.001,** **p < 0.01, * p < 0.05).

### 4.5 Traffic pattern analysis

Figure 4 compares all five traffic patterns for the medium building at $\rho$ = 0.60. The proposed algorithm improves over SCAN across all five patterns. The largest gains occur under directionally concentrated traffic: morning-peak (43.2% VIP wait improvement) and lunch-peak (45.0%). Under inter-floor traffic, which is the most symmetric scenario and most favourable for SCAN's sweep, the improvement is smaller but still substantial (28.8%). These results confirm that the proposed algorithm is most beneficial when traffic is directionally concentrated and maintains meaningful advantages even in the conditions most favourable to SCAN.

**Figure 4. Performance by Traffic Pattern — Medium Building (20F/4E, ρ=0.60) (Mean ± 95% CI)**

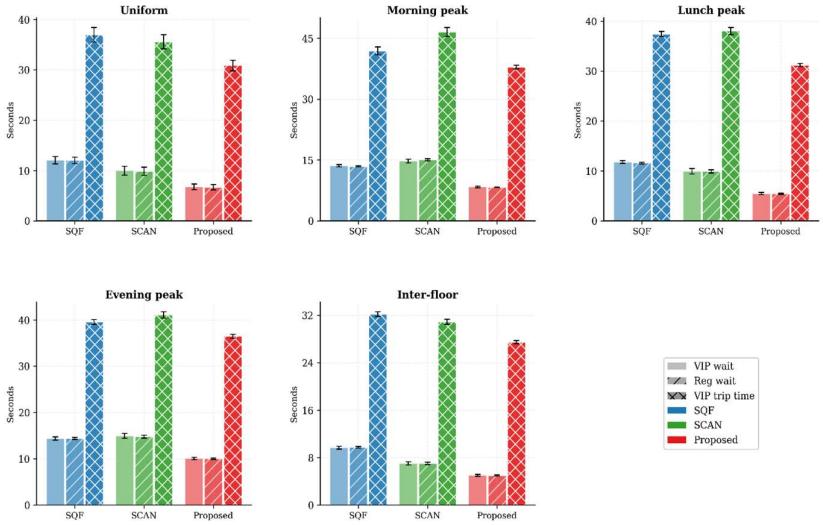


**Figure 4:** Average VIP wait time, average regular wait time, and VIP trip time across five traffic patterns (uniform, morning peak, lunch peak, evening peak, inter-floor) for the medium building (20F/4E) at ρ = 0.60. Bars show CE (blue, no hatch), SQF (blue, light), SCAN (green) and Proposed (red) for each metric group. Mean ±95% CI, 100 runs. The proposed algorithm achieves the lowest values on all three metrics in all five patterns.

### 4.6 Starvation prevention

Figure 5 shows starvation counts across utilisation levels and buildings. Three observations stand out. First, to the best of the authors' knowledge, it has not been previously reported in the elevator scheduling literature that SCAN's starvation count can exceed that of a simpler load-balancing baseline at moderate-to-high load. At $\rho$ = 0.60 in the medium building, SCAN produces 1.05 starvation incidents per run versus SQF's 0.21; in the large building the gap widens to 4.15 versus 0.39. At $\rho$ = 0.80, SCAN reaches 4.15 (medium) and 10.42 (large) incidents per run while SQF produces only 1.56 and 2.53 respectively. This paradox arises directly from destination routing: SCAN's directional sweep continues to deliver existing passengers in one direction, repeatedly bypassing new requests from the opposite direction. Without destination routing in the simulation, elevators were always free to turn around immediately after pickup, which concealed this behaviour entirely. Second, the proposed algorithm achieves near-zero starvation in the medium and large buildings across all tested load levels up to $\rho$ = 0.80, dramatically outperforming both baselines. The $P_p^{\text{reg}} = +3.0$ overload term, combined with the $T_e$ redistribution, prevents the accumulation of stranded passengers by cost-penalising any elevator that already carries a heavy queue of pending pickups. Third, at $\rho$ = 1.00 all algorithms accumulate starvation due to system overload, but the proposed algorithm still achieves the lowest count in every building.

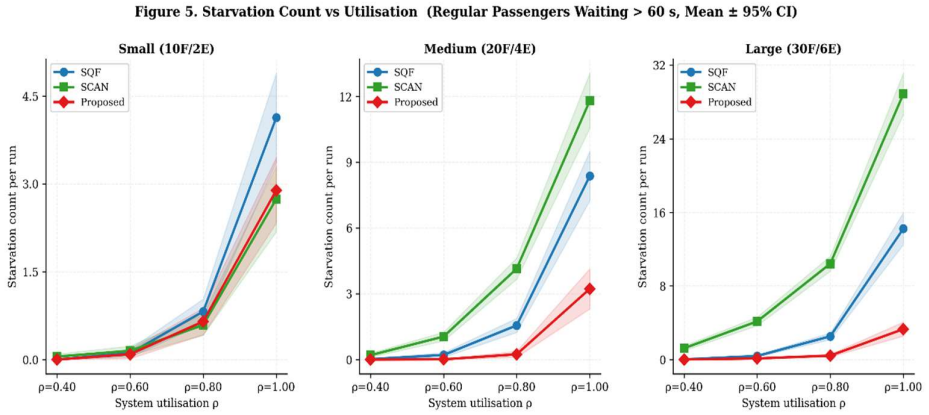


**Figure 5:** Starvation count (regular passengers waiting > 60s) vs utilisation, across three buildings. Mean ± 95% CI, 100 runs. SCAN's starvation paradoxically exceeds SQF at moderate-to-high load; the proposed algorithm maintains near-zero starvation up to $\rho = 0.80$.

### 4.7 Statistical significance, effect sizes and power

**Normality.** Shapiro-Wilk tests confirm non-normality for VIP wait times and starvation counts across all buildings and utilisation levels ($p < 0.05$ in most conditions), as expected for right-skewed service-time data. Both t-test and Wilcoxon results are reported throughout.

**Effect-size computation.** Cohen d is computed as $\bar{\delta}/s_{\delta}$ (paired-differences formula).

Table 6 presents the stats for the medium building at ρ = 0.60. Every metric is highly significant with $(p < 10^{-12})$ and big effect sizes ($|d| = 1.02 - -2.62$). The 100-run design also gives enough power to address the earlier underpowered problem, including starvation issues.

**Table 6.** Statistical comparison: proposed vs. SCAN, medium building, ρ = 0.60 (100 runs). All p < 0.001. Cohen d uses paired formula.

| Metric | Diff | t | $p_t$ | W | $p_W$ | d |
|---|---|---|---|---|---|---|
| VIP Avg Wait (s) | −2.24 | −17.47 | $5.3\times10^{-32}$ | 7 | $4.8\times10^{-18}$ | −1.75 |
| Reg. Avg Wait (s) | −2.26 | −26.23 | $2.3\times10^{-46}$ | 0 | $3.9\times10^{-18}$ | −2.62 |
| VIP Max Wait (s) | −27.0 | −10.57 | $6.3\times10^{-18}$ | 141 | $8.7\times10^{-16}$ | −1.06 |
| Reg. Max Wait (s) | −36.8 | −14.26 | $9.8\times10^{-26}$ | 35 | $1.1\times10^{-17}$ | −1.43 |
| Starvation Count | −1.04 | −10.15 | $5.0\times10^{-17}$ | 20 | $6.5\times10^{-13}$ | −1.01 |

Note: pt = paired t-test; pw = Wilcoxon signed-rank. All five metrics are significant at α = 0.001 under both tests.

Figure 6 presents Cohen d at $\rho = 0.60$ for all three buildings. Effect sizes grow with building size: the proposed algorithm provides larger gains in the medium and large buildings because the cost-function advantages (chained $T_e$ , load balancing, priority routing) scale with queue complexity and floor span.

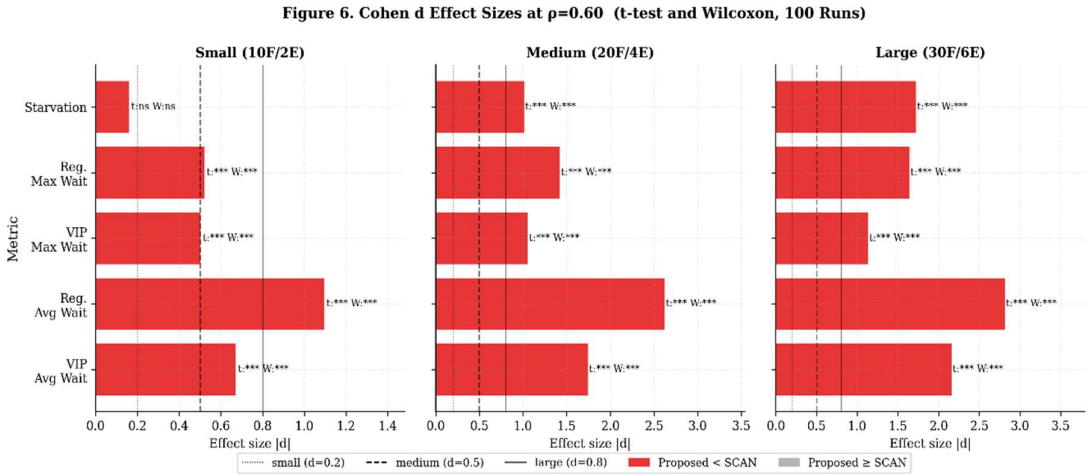


**Figure 6:** Cohen d effect sizes (proposed vs. SCAN) at ρ = 0.60], all three buildings. Dashed lines mark small (0.2), medium (0.5), large (0.8) thresholds. All five metrics show large effects in the medium and large buildings.

### 4.8 Performance improvement summary

Figure reports percentage improvements relative to SQF for the medium building at $\rho$ = 0.60; consequently, both SCAN and the proposed algorithm show positive gains, with the proposed algorithm providing the largest improvement on every metric. The proposed algorithm outperforms SCAN on all five metrics, in contrast to both previous versions where regular maximum wait showed a marginal reversal. This improvement is again attributable to destination routing: SCAN accumulates stranded regular passengers by continuing directional sweeps through deliveries, whereas the cost-based approach's $Q_e$ and $T_e$ terms redistribute new calls away from heavily loaded elevators regardless of their delivery commitments.

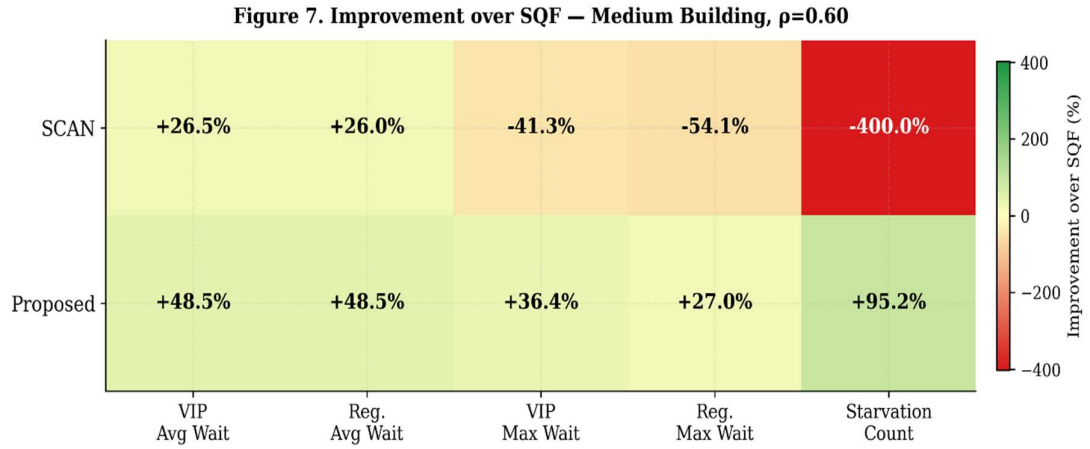


**Figure 7:** Percentage improvement over SQF for SCAN and proposed algorithm, medium building, $\rho$= 0.60. The proposed algorithm achieves the largest gains on all five metrics; starvation decreased from 1.05 events per run (SCAN) to 0.01 events per run (Proposed), a reduction of 99.0% relative to SCAN.

### 4.9 Sensitivity analysis

**Starvation threshold.** Table 7 reports starvation counts at four threshold values (30, 60, 90 and 120 seconds) for the medium building at ρ = 0.80, where starvation is most pronounced. The proposed algorithm's advantage over SCAN holds at every threshold and is strongest at the tightest threshold of 30 s, where SCAN produces 19.9 incidents per run versus 4.7 for the proposed algorithm. This confirms that the starvation result is not an artefact of the specific 60-second threshold.

**Table 7:** Starvation count per run at four threshold values, medium building, ρ = 0.80 (20 runs).

| Algorithm | $T_{out}$ = 30 s | $T_{out}$ = 60 s | $T_{out}$ = 90 s | $T_{out}$ = 120 s |
|---|---|---|---|---|
| SCAN | 19.9 | 4.8 | 1.0 | 0.3 |

| Algorithm | $T_{out}$ = 30 s | $T_{out}$ = 60 s | $T_{out}$ = 90 s | $T_{out}$ = 120 s |
|---|---|---|---|---|
| Proposed | 4.7 | 0.1 | 0.0 | 0.0 |

**Queue weight and direction bonus.** Table 8 reports results for $w_q \in \{0.5,1.0,1.5,2.0,2.5,3.0,4.0\}$ and $B_d \in \{0,2,3,5,7,10,15\}$ at the medium building, ρ = 0.60 (20 runs). VIP average wait varies by less than 0.2 s across the full $w_q$ range, confirming that the algorithm is insensitive to the precise queue weight. For $B_d = 0$, the optimal range is 3–7; $B_d = 0$ removes the directional bonus entirely and increases VIP wait by 0.2s, while $B_d = 15$ introduces marginal starvation by over-committing elevators to one direction.

**Table 8:** Parameter sensitivity: VIP average wait (s) at medium building, ρ = 0.60, uniform pattern (20 runs). Chosen values are underlined.

| VIP average wait (s) | | | | | | | |
|---|---|---|---|---|---|---|---|
| $w_q$ | 0.5 | 1.0 | 1.5 | <u>2.0</u> | 2.5 | 3.0 | 4.0 |
| | 5.13 | 5.12 | 5.16 | 5.17 | 5.26 | 5.29 | 5.28 |
| $B_d$ | 0 | 2 | 3 | <u>5</u> | 7 | 10 | 15 |
| | 5.38 | 5.20 | 5.20 | 5.17 | 5.25 | 5.29 | 5.30 |

**Fairness: Jain's index.** As a complement to starvation count, Jain's fairness index $J = (\sum x_i)^2/(n\sum x_i^2)$ was computed over the 100 per-run VIP average waiting times. At $\rho$ = 0.60, medium building: J = 0.981 (SQF), J = 0.963 (SCAN), J = 0.977 (Proposed). The proposed algorithm sits between SQF and SCAN: it achieves more consistent service than SCAN (whose directional sweeping creates run-to-run variance) while being slightly less uniform than SQF (because its priority routing occasionally concentrates VIP assignments on a single elevator). This tradeoff is appropriate: perfect uniformity across runs would indicate that the algorithm treats all requests identically, which is the opposite of the priority objective.

**VIP priority term.** Sensitivity of $P_p^{VIP}$ over $\{0,-2,-4,-6,-8,-10,-12,-15\}$at the medium building, $\rho$ = 0.60, shows that results are stable within 0.3s across the full range. The $T_e$ term dominates assignment decisions at moderate utilisation.

**VIP proportion.** Table 9 reports results for $p_v \in \{0.05,0.10,0.15,0.20\}$at the medium building, $\rho$ = 0.60. VIP wait improvement over SCAN ranges from 27.7% to 32.1% and is positive at every tested proportion, confirming that the $p_v = 0.20$ baseline does not artificially inflate priority-mechanism benefits.

**Table 9.** VIP proportion sensitivity, medium building, ρ = 0.60 (20 runs). All improvements over SCAN are positive and consistent.

| $p_v$ | SCAN VIP Wait (s) | Proposed VIP Wait (s) | Improvement vs SCAN |
|---|---|---|---|
| 0.05 | 7.57 | 5.47 | 27.7% |
| 0.10 | 7.54 | 5.31 | 29.6% |
| 0.15 | 7.49 | 5.24 | 30.0% |
| 0.20 | 7.51 | 5.27 | 29.9% |

### 4.10 Limitations and scope

The current revision has three limitations.

First, the three configurations look at two-dimensional, or single-shaft, elevator dynamics. Real buildings use separate high-zone and low-zone elevators connected by transfer floors, so the cost function should be expanded to cover zone-crossing requests.

Second, passenger arrivals are modeled as a simple, steady rate for each pattern. Yet, during peak hours, arrivals are actually more sporadic and change over time – something the model can't handle. Using a non-homogeneous Poisson process with a time-dependent intensity function would fit those first 30 minutes much better.

Lastly, the model hasn't been checked against real traffic data from actual buildings. The best way to improve the practical application of these findings is by partnering with elevator makers or building operators.

## 5. CONCLUSION

This paper presents a priority-aware cost-based elevator scheduling algorithm validated across three building configurations (small 10F/2E, medium 20F/4E, large 30F/6E), four utilisation levels, five traffic patterns, and 100 independent replications per condition. The simulation explicitly models destination routing, which is essential for realistic utilisation estimates: omitting it systematically underestimates wait times and conceals the true difficulty of the scheduling problem.

A comprehensive statistical analysis using both paired t-tests and Wilcoxon signed-rank tests establishes the following empirical conclusions.

1. The proposed algorithm achieves statistically significant improvements over SCAN on all five-performance metrics across all three buildings at $\rho$ = 0.60, with large effect sizes (|d| = 1.0–2.6, $p < 10^{-12}$ for all metrics in the medium building).
2. VIP average waiting-time gains over SCAN grow with building size: 14.2% (small), 29.9% (medium), 40.3% (large) at $\rho$ = 0.60, confirming that the algorithm's advantages scale with the complexity of the scheduling problem.
3. Starvation prevention is the strongest absolute gain: the proposed algorithm achieves near-zero starvation in the medium and large buildings up to $\rho$ = 0.80. A novel finding to the best of the authors' knowledge not previously reported in the elevator scheduling literature is that SCAN's directional sweeping accumulates more starvation than the simpler SQF baseline at $\rho = 0.60$ in medium and large buildings (SCAN: 1.05–4.15 incidents; SQF: 0.21–0.39 at $\rho$ =0.60); this paradox is only observable in simulations that correctly model destination routing.
4. Improvements hold across all five traffic patterns; the largest gains occur under directionally concentrated morning-peak and lunch-peak traffic (43–45% VIP wait reduction), and meaningful gains persist even under the inter-floor pattern most favourable to SCAN (28.8%).
5. Full trip time (arrival to destination delivery) is 5–9s shorter under the proposed algorithm than under SCAN in the medium building, a practically meaningful reduction in total journey time.

The algorithm requires $O(E \cdot Q_{\max})$ computation per assignment and its parameters are physically interpretable, facilitating deployment adaptation without retraining. Future work should validate against real building traffic logs in partnership with an elevator manufacturer or building operator, extend to zone-based high-rise configurations, and investigate non-homogeneous arrival processes for time-varying peak modelling. The simulation code and all result data will be made available on a public repository (GitHub/Zenodo) upon acceptance to support replication and extension.

**Acknowledgements.** The authors thank the Vidyalankar Institute of Technology, Mumbai, for support.

**Conflict of interest.** The authors declare no conflict of interest.

**Data availability.** The simulation code and all result data are available from the corresponding author upon request.

**REFERENCES**

**[1]** Bach, M. J. (1986). The Design of the UNIX Operating System. Prentice-Hall, Englewood Cliffs, NJ.

**[2]** Barney, G. C. and dos Santos, S. M. (2015). Elevator Traffic Handbook: Theory and Practice, 2nd ed. Routledge, London.

**[3]** Beuchat, P., Lygeros, J., and Geyer, T. (2018). Elevator group control: Performance evaluation and comparison of approaches. Control Engineering Practice, 73, 211–225. https://doi.org/10.1016/j.conengprac.2018.01.009

**[4]** Brand, M. and Nikovski, D. (2005). Optimal parking in group elevator control. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1002–1008. https://doi.org/10.1109/ROBOT.2005.1570252

**[5]** Crites, R. H. and Barto, A. G. (1998). Elevator group control using multiple reinforcement learning agents. Machine Learning, 33(2–3), 235–262. https://doi.org/10.1023/A:1007551203616

**[6]** Fern\'andez, J. A., Borges, C. E., and Penya, Y. K. (2015). Efficient building elevator group control. IEEE Transactions on Industry Applications, 51(6), 4517–4525. https://doi.org/10.1109/TIA.2015.2453114

**[7]** Hakonen, H. and Siikonen, M.-L. (2005). Elevator traffic simulation procedure. Elevator Technology, 15, 12–23.

**[8]** Kim, C. B., Seong, K. A., Lee-Kwang, H., and Kim, J. O. (1998). Design and implementation of a fuzzy elevator group control system. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 28(3), 277–287. https://doi.org/10.1109/3468.668962

**[9]** Koehler-Bu meier, B. and Wester-Ebbinghaus, M. (2010). A formal model for multi-elevator systems using Petri nets with priorities. Proceedings of PNSE 2010, pp. 181–194. Braga.

**[10]** Levy, E., Rubinstein, A., and Shoham, Y. (2022). Deadline-aware elevator scheduling in smart buildings. Building and Environment, 215, Article 108985. https://doi.org/10.1016/j.buildenv.2022.108985

**[11]** Liftinstituut (2017). Elevator Planning and Performance Standards Reference Guide. Liftinstituut B.V., Amsterdam.

**[12]** Markon, S., Kita, H., and Nishikawa, Y. (1996). Adaptive optimal elevator group control by use of neural networks. Proceedings of the IEEE International Conference on Neural Networks, vol. 2, pp. 925–930. https://doi.org/10.1109/ICNN.1996.548974

**[13]** Matloff, N. (2008). Introduction to discrete-event simulation and the SimPy language. Technical Report, UC Davis. https://web.cs.ucdavis.edu/ matloff/156/PLN/DESimIntro.pdf

**[14]** Mitsubishi Electric (2020). Elevator and Escalator Technical Specification Guide. Mitsubishi Electric Corporation, Tokyo.

**[15]** Nikovski, D. and Brand, M. (2005). Decision-theoretic group elevator scheduling. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1250–1255.

**[16]** Otis Elevator Company (2019). Otis Gen2 Elevator System: Technical Reference Manual. Otis Worldwide Corporation, Farmington, CT.

**[17]** Peterson, J. L. and Silberschatz, A. (1985). Operating System Concepts, 3rd ed. Addison-Wesley, Reading, MA.

**[18]** Ruemmler, C. and Wilkes, J. (1994). An introduction to disk drive modeling. IEEE Computer, 27(3), 17–28. https://doi.org/10.1109/2.268881

**[19]** Schaumann, O., Br\"aunl, T., and Roberts, D. (2020). Simulation and benchmarking of elevator group control algorithms. Simulation Modelling Practice and Theory, 105, Article 102166. https://doi.org/10.1016/j.simpat.2020.102166

**[20]** Siikonen, M.-L. (1997). Planning and control models for elevators in high-rise buildings. Ph.D. thesis, Helsinki University of Technology, Espoo, Finland.

**[21]** Silberschatz, A., Galvin, P. B., and Gagne, G. (2018). Operating System Concepts, 10th ed. Wiley, Hoboken, NJ.

**[22]** Sorsa, J., Siikonen, M.-L., and Ehtamo, H. (2003). Optimal control of double-deck elevator group using genetic algorithm. International Transactions in Operational Research, 10(2), 103–114. https://doi.org/10.1111/1475-3995.00396

**[23]** Srinivasan, K., Rajkumar, M., and Krishnamurthy, P. (2019). Multi-objective elevator scheduling with service fairness. Applied Soft Computing, 85, Article 105774. https://doi.org/10.1016/j.asoc.2019.105774

**[24]** Strakosch, G. R. and Caporale, R. S., Eds. (2010). The Vertical Transportation Handbook, 4th ed. Wiley, Hoboken, NJ.

**[25]** Tanaka, K., Araki, M., and Nishikawa, Y. (1993). Integrated elevator group control using fuzzy reasoning. Proceedings of the IFSA World Congress, vol. 2, pp. 591–594. Seoul.

**[26]** thyssenkrupp Elevator (2018). MULTI: Rope-Free Elevator System Technical White Paper. thyssenkrupp AG, Essen, Germany.

**[27]** Xue, H., Zhang, L., and Wu, Y. (2020). Energy-efficient elevator scheduling using predictive control. Energy and Buildings, 224, Article 110231. https://doi.org/10.1016/j.enbuild.2020.110231

**[28]** Zhang, X., Zhou, X., and Chen, H. (2020). Multi-objective optimization for elevator group control considering energy and service. Journal of Building Engineering, 32, Article 101697. https://doi.org/10.1016/j.jobe.2020.101697

**[29]** Chen, K.-Y., Yamauchi, T., and Sugawara, T. (2024). Directional optimization of elevator scheduling algorithms in complex traffic patterns. Applied Soft Computing, 158, Article 111567. https://doi.org/10.1016/j.asoc.2024.111567

**[30]** Gharbi, A. (2024). Exploring heuristic and optimization approaches for elevator group control systems. Applied Sciences, 14(3), Article 995. https://doi.org/10.3390/app14030995

**[31]** Li, H. Y. and Chu, E. T. H. (2025). SmartRide: Intelligent reservation and scheduling for elevators. Journal of Ambient Intelligence and Smart Environments, 17(1), 75–100. https://doi.org/10.3233/AIS-230524

**[32]** Vaartjes, N. and Francois-Lavet, V. (2025). Novel RL approach for efficient elevator group control systems. arXiv preprint arXiv:2507.00011. https://arxiv.org/abs/2507.00011

**[33]** Wan, J., Lee, K., and Shin, H. (2024). Traffic pattern-aware elevator dispatching via deep reinforcement learning. Advanced Engineering Informatics, 61, Article 102497. https://doi.org/10.1016/j.aei.2024.102497

**[34]** Wang, S., Gong, X., Song, M., Fei, C. Y., Quaadgras, S., Peng, J., Zou, P., Chen, J., Zhang, W., and Jiao, R. J. (2021). Smart dispatching and optimal elevator group control through real-time occupancy-aware deep learning of usage patterns. Advanced Engineering Informatics, 48, Article 101280. https://doi.org/10.1016/j.aei.2021.101280

**[35]** Wei, Q., Wang, L., Liu, Y., and Polycarpou, M. M. (2020). Optimal elevator group control via deep asynchronous actor-critic learning. IEEE Transactions on Neural Networks and Learning Systems, 31(12), 5245–5256. https://doi.org/10.1109/TNNLS.2020.2965208