Journal of Electronic Research and Application, 2026, Volume 10, Issue 9
https://bbwpublisher.com/journal/JERA
Online ISSN: 2208-3510
Print ISSN: 2208-3502

Q-AWED: Adaptive-Weight Elevator Dispatching with Q-Learning Across Multiple Traffic Scenarios

Tianyun Liu*

Beijing Aidi School, Beijing, China

***Corresponding author:** Tianyun Liu, 2768305646@qq.com

Copyright: © 2026 Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0), permitting distribution and reproduction in any medium, provided the original work is cited.

Abstract: In different elevator transportation modes, there is a trade-off among waiting times, movements, and load balancing. This article uses adaptive-weight Q-learning, based on traffic conditions, to choose a weight scheme and use a weighted dispatch cost function, travel distance, stop proxy, load, and direction to dispatch a specific elevator. The method used 40 validation episodes and 40 test episodes to compare with seven main elevator dispatching approaches in six scenarios. Results show that Q-AWED improved by 3.88% compared with the untrained default policy, and the relevant differences were corrected using the Holm correction.

Keywords: Elevator dispatching; Q-learning; Adaptive weights; Multi-objective optimization; Traffic simulation

Online publication: October 30, 2026

1. Introduction

With the development of cities and buildings, there are more layers and traffic flow in the building. Therefore, the elevator dispatching problem has become a critical problem that needs to be optimized. A long waiting time not only reduces passengers' experience, but also expends more energy: elevators need to move constantly, judging requests more often. By contrast, a single rule is hard to apply to every scenario; every buildings have different number of layers, traffic flow, and the number of elevators. For instance, during the morning rush hour in the building, the system needs to increase the weight of requests from the first floor, making more elevators near the first floor in order to reduce the movement time and passengers' waiting time. Instead, in a shopping mall, the passengers' requests on each floor are random, and the number on each floor is approximately the same; therefore, the elevators need to identify the floor with the most requests and the floors that are close to each other, optimizing and reducing the overall average waiting time for users. Recent approaches such as First-Come, First-Served, Nearest Car, Collective Control, Static Zoning, and Destination Dispatch are limited by a single rule, hard to achieve universal applicability across all scenarios.

This article used an adaptive weights Q-learning, based on traffic conditions, to choose a weight scheme, finding an algorithm to learn traffic changes and trying to increase applicability across multiple scenarios; Thus, this algorithm will consider optimizing passengers' average waiting time, the long-wait proportion, elevators' average stop proxy, energy proxy, and load imbalance.

Ruokokoski et al. studied elevator dispatching with destination control, comparing an assignment formulation that only allocates requests under collective control with a routing formulation that explicitly plans elevator routes, using average waiting time and average journey time as separate objectives [1]. Under light and normal traffic, the formulations often produced identical solutions when minimizing average waiting time; explicit routing generally performed better for average journey time, showing that performance depends on the objective and traffic intensity. Kompauer developed mixed-integer linear programming, greedy, and proximal policy optimization approaches for elevator group control with destination registration, comparing them under up-peak, down-peak, and interfloor traffic at different system scales [2]. Dynamic mixed-integer linear programming generally produced good small-instance solutions at high computational cost; the greedy algorithm was fastest, whereas reinforcement learning was sensitive to hyperparameters and system size but could outperform the greedy method in small systems and when passengers waited in concentrated groups. Han et al. incorporated real-time car-occupancy awareness into a proximal policy optimization framework, defining a multi-objective reward from average waiting time, energy consumption, and crowding, and comparing it with shortest-seek-time-first, SCAN-EDF, and a genetic algorithm across 24 peak and interfloor datasets [3]. In peak traffic, their method reduced average waiting time by 19.58 seconds and energy consumption by 2.1 kWh relative to SCAN-EDF; in interfloor traffic, it reduced energy consumption by 2.3 kWh and crowding by 8% relative to the genetic algorithm, although objective weights remained fixed through the analytic hierarchy process and cross-building transfer was left for future work. Zhu studied four elevators in a 33-floor building, combining total operating time and total passenger waiting time in a weighted objective and comparing First-Come, First-Served, polling, and a genetic algorithm in simulation [4]. Relative to First-Come, First-Served, the genetic algorithm reduced operating time, waiting time, and energy consumption by approximately 25%, 38%, and 20%, respectively, but the model primarily addressed one building configuration, leaving complex traffic and online dynamic dispatching for further validation. Lin et al. developed simulation and multi-objective scheduling models for single- and dual-lift configurations in a double-deep, tier-captive automated storage system, using an improved NSGA-II to jointly optimize shuttle waiting, reshuffling, and total retrieval time [5]. Additional lifts provided limited gains at low workloads but became more beneficial as workload increased; because the study concerned warehouse lifts rather than passenger elevators, it provides methodological guidance on capacity configuration and multi-objective resource scheduling rather than direct passenger-elevator evidence. Yamauchi et al. used cameras to estimate waiting passengers, required passenger space, and remaining car space, incorporating estimated arrival time and spare capacity into dispatch priorities to improve waiting fairness for ordinary passengers and those using wheelchairs or strollers or carrying large luggage. Simulations supported both transport efficiency and waiting fairness, while the authors noted that waiting and spare-space weights should adapt to traffic patterns and arrival rates [6]. Gharbi represented elevator group control as a finite-state machine, comparing heuristics that maintain travel direction or reverse after the last call, then optimizing dispatching through genetic algorithms and simulated annealing [7]. Simulations with different floor and elevator counts favored the genetic algorithm for average waiting time, although iterative computation time and parameter

selection remained important constraints on real-time use. Cao et al. generated passengers from a typical three-peak traffic model, built an elevator simulator including capacity, acceleration, and a maximum-waiting threshold, and attempted to improve a hand-designed baseline controller with deep Q-learning [8]. The deep Q-network did not achieve consistent improvement, which the authors attributed to external random calls preventing the simplified state from satisfying their Markov decision process assumptions; this illustrates the dependence of reinforcement learning on state representation, environment modeling, and convergence verification rather than the algorithm's name alone. Al-Sharif et al. analyzed static sectoring during up-peak traffic using calculations and Monte Carlo simulation, dividing floors into service sectors to reduce stops per round trip and the highest reversal floor, thereby shortening round-trip and journey times [9]. More sectors increased the ability to handle high arrival rates but could also increase average waiting time, motivating adaptive selection of sector count according to traffic intensity. Batosalem et al. used access-control records and lobby CCTV data from a 20-floor university building to reproduce a Static Zoning system through agent-based modeling, calibrating round-trip and average waiting times over 30 simulation runs [10]. An illustrative experiment restricting downward elevator service reduced mean round-trip time from 2.30843 to 1.53990 and mean waiting time from 1.92940 to 0.75916, but the optimization stage had only five repetitions and required downward passengers to use stairs, limiting applicability to other buildings and user groups. Sorsa et al. formulated elevator dispatching as stochastic bilevel optimization, using a genetic algorithm to assign calls at the upper level and planning elevator routes under expected stochastic demand at the lower level, subject to a 500-millisecond online computation limit [11]. Simulations and commercial installations supported real-time feasibility, with average waiting time approximately 15% below the Enhanced Spacing Principle and 30% below another rule-based method, demonstrating the importance of jointly considering stochastic demand, call assignment, and route planning.

Existing studies show that First-Come, First-Served, direction-preserving rules, Collective Control, and Static Zoning are computationally fast and straightforward to implement, but their effectiveness often depends on traffic patterns. Destination control, mixed-integer linear programming, and genetic algorithms support more extensive assignment or route optimization, subject to information requirements and real-time computational costs. Reinforcement learning enables traffic-dependent decisions and has been used to optimize waiting, energy consumption, and crowding jointly, but its performance depends on state modeling, hyperparameters, system scale, and reward weights; some studies still use fixed weights or train on limited scenarios. A framework that adjusts multi-objective preferences according to traffic while retaining a transparent dispatch cost function therefore merits investigation. Q-AWED uses Q-learning to select state-dependent weight schemes and evaluates average waiting time, long-wait proportion, average stop proxy, energy proxy, and load imbalance under shared request streams and multiple traffic scenarios.

2. Methodology

Seven methods are compared under identical request streams: First-Come, First-Served (FCFS), Nearest Car (NC), approximations of Collective Control (SCAN), Static Zoning (SZ), and Destination Dispatch (DD), Traffic-Rule Heuristic (RH), and Q-AWED. FCFS assigns in arrival order to the earliest available elevator; NC minimizes pickup distance plus 0.65 times remaining busy time. SCAN favors matching directions, SZ penalizes assignments outside an origin zone, DD favors similar destination zones, and RH selects weight

schemes through fixed traffic rules. These are simplified implementations, not full reproductions of published controllers; RH does not train a random forest or reinforcement learning network. One Q-table is trained separately for each scenario. Raw metrics are primary, and composite scores are supplementary.

2.1. System model and experimental setting

A building has F floors, with lobby 0, and M elevators initially at the lobby. Request $r=(t_r,o_r,d_r)$ specifies arrival time, origin, and a different destination. Elevator e has queue-end floor f_e and availability time T_e. Tasks are served serially without shared rides or capacity constraints. Pickup distance is $h_{er}=|f_e-o_r|$, passenger travel distance is $d_r^{trip}=|d_r-o_r|$, and:

$$\begin{aligned} W_{er} &= max(0,T_e-t_r)+h_{er}, \quad s_{er}=2+1(T_e>t_r), \\ t_r^{pick} &= t_r+W_{er}, \quad T_e^{'}=t_r+W_{er}+d_r^{trip}+s_{er}, \quad f_e^{'}=d_r. \end{aligned} \tag{1}$$

Moving one floor or adding one stop-proxy unit costs one simulation time unit; time is not calibrated to physical seconds. The stop proxy is a rule, not a count of physical stops. All generated requests receive scheduled service, including completions after horizon T. Results therefore include every generated request's scheduled wait. Acceleration, regenerative braking, and physical door and capacity dynamics are not modeled.

Six synthetic scenarios are used: Up-Peak, Down-Peak, Short-Trip Random, Hotspot Floors, Three-Phase Commuting, and Residential Double-Peak. Seeded Bernoulli draws generate arrivals, with an additional request when $t\ mod\ [T/3]=0$ (at most three requests per step). Upward phases start at the lobby, downward phases end there, and mixed phases sample different origin and destination floors. Short-Trip Random favors offsets within three floors, with resampling when necessary; Hotspot Floors restricts requests to four designated floors. The phase sequences are up/up/mixed, mixed/down/down, all mixed, all mixed, up/mixed/down, and down/mixed/up, respectively. These are synthetic patterns, not measured building categories. **Table 1** gives the main settings; training and test requests use disjoint seeds, with identical test requests supplied to every algorithm.

Table 1. Building configurations, episode lengths, and repetitions

Scenario	F	M	T	**Training**	**Testing**	τ
Up-peak	14	4	135	210	34	12
Down-peak	14	4	135	210	34	12
Short-trip random	9	3	145	220	36	14
Hotspot floors	12	4	135	220	38	11
Three-phase commuting	16	4	135	210	35	13
Residential double-peak	13	3	125	190	34	14

2.2. Q-AWED algorithm design

2.2.1. State, actions, and dispatch cost

At each nonempty time step, state $s_t=(m_t,\delta_t,g_t,b_t)$ encodes the generator's known phase (upward, mixed, or downward), request count (at most one, two, or at least three), majority direction (upward, downward, or tied), and whether all elevators are busy. There are at most $3\times3\times3\times2=54$ states. An action selects one of six entire weight schemes for all requests in that step (**Table 2**). Q-values start at zero, but the hand-designed

weights do not; learning updates scheme values rather than individual weights.

Table 2. Six base weight schemes: waiting, travel, stops, assignment load, and direction

Scheme	w_W	w_E	w_S	w_L	w_D
Passenger-First	0.96	0.03	0.04	0.02	0.24
Energy-Saving	0.24	0.52	0.14	0.1	0.05
Stop-Reduction	0.3	0.12	0.46	0.12	0.12
Load-Balance	0.3	0.1	0.12	0.48	0.08
Balanced	0.46	0.2	0.2	0.14	0.12
Long-Wait Rescue	0.92	0.04	0.05	0.08	0.22

The selected base weights are adjusted by existing engineering rules. Let q be the busy-elevator fraction, τ the long-wait threshold, and $clip(x,l,u)=min(u,max(l,x))$. Define $x_W=clip(\lambda_W/1.35,0.70,1.55)$, $x_E=clip(\lambda_E/0.45,0.60,1.60)$, and $x_L=clip(\lambda_L/5,0.75,1.60)$. With $1(\cdot)$ denoting an indicator, the applied weights are:

$$\begin{gathered}\tilde{w}_W=w_W\left(0.72+0.28x_W\right)+0.081\left(\delta_t\geq 1\right)\\+0.081(q>0.65)+0.101(x_L>1.2 \text{ or } \tau\leq 12),\\ \tilde{w}_E=w_E\left(0.78+0.22x_E\right),\\ \tilde{w}_S=w_S\left[1+0.121\left(\text{Short-Trip Random}\right)\right],\\ \tilde{w}_L=w_L+0.031(q>0.80)+0.021\left(\text{Hotspot Floors}\right),\\ \tilde{w}_D=w_D+0.04+0.161\left(\text{non-mixed phase}\right)\\+0.061\left(\text{Three-Phase Commuting}\right).\end{gathered} \tag{2}$$

The full method therefore combines Q-based selection with manual corrections. For cumulative assigned-request count N_e and energy proxy $E_{er}=h_{er}+d_r^{trip}$, dispatch selects:

$$\begin{gathered}C\left(e,r\right)=\tilde{w}_W W_{er}+\tilde{w}_E E_{er}+\tilde{w}_S s_{er}+\tilde{w}_L N_e-\tilde{w}_D A\left(e,r\right),\\ e^*=\arg\min_e C\left(e,r\right).\end{gathered} \tag{3}$$

The terms represent waiting, travel, stops, assignment load, and direction preference. If the previous task direction is zero, $A=0$; if directions match, $A=1$ when the new origin is ahead of the queue-end position and $A=0.35$ otherwise; opposite directions give $A=-1$. Return travel is still counted in h_{er}, so the 0.35 discount is a preference rather than an on-route saving. Ties select the lower elevator index. Costs are ranking values, not electricity or time measurements.

2.2.2. Reward and Q-learning

For a batch of $n_t>0$ requests, let $\bar{W}_t,\bar{E}_t,\bar{S}_t$ be mean scheduled waiting, travel, and stop proxy. Define $L_t=n_t^{-1}\sum_r 1(W_r\geq\tau)$ and $I_t=[M^{-1}\sum_e(N_e(t)-\bar{N}(t))^2]^{1/2}$, where $\bar{N}(t)$ is the mean cumulative assignment count across elevators. The reward is:

$$R_t=-\left(\lambda_W\bar{W}_t+\lambda_E\bar{E}_t+0.35\bar{S}_t+\lambda_L L_t+0.18I_t\right). \tag{4}$$

Reward is computed immediately from scheduled outcomes, not after physical service. Coefficients express preferences; higher reward alone does not establish improvement in every metric. At training episode k, exploration probability is $\varepsilon_k=\varepsilon_0(0.985)^k$: choose a scheme uniformly with this probability,

otherwise choose the largest Q-value, resolving ties by lower action index. The selected entry is updated by:

$$Y_t = R_t + \gamma \max_{a'} Q(s', a'),$$
$$Q_{new}(s_t, a_t) = (1-\alpha) Q_{old}(s_t, a_t) + \alpha Y_t. \tag{5}$$

Here, α controls the response to new evidence and γ the weight of future outcomes. Only the chosen state-action entry changes. Training stops after the preset episode count; testing fixes $\varepsilon = 0$ and freezes Q-values. **Table 3** lists the learning and reward settings. They are preset configurations, not theoretically optimal ranges. The state omits full queues, and the current code updates only nonempty steps without explicitly zeroing terminal-state value; no convergence guarantee is claimed.

Table 3. Learning and reward parameters by scenario

Scenario	$\acute{a}$	$\tilde{a}$	$\mathring{a}_0$	$\ddot{e}_W$	$\ddot{e}_E$	$\ddot{e}_L$
Up-peak	0.20	0.92	0.33	1.77	0.25	6.15
Down-peak	0.20	0.91	0.33	1.74	0.25	6.07
Short-trip random	0.21	0.89	0.40	1.61	0.54	5.51
Hotspot floors	0.20	0.92	0.36	1.77	0.32	6.53
Three-phase commuting	0.20	0.90	0.36	1.66	0.34	5.87
Residential double-peak	0.18	0.88	0.33	1.54	0.50	5.55

2.3. Evaluation and statistical analysis

For episode j with N_j requests and elevator assignment counts $N_{e,j}$, the five metrics are:

$$\bar{W}_j = \frac{1}{N_j}\sum_i W_i, \quad L_j = \frac{1}{N_j}\sum_i 1(W_i \geq \tau),$$
$$\bar{E}_j = \frac{1}{N_j}\sum_i (h_i + d_i^{trip}), \quad \bar{S}_j = \frac{1}{N_j}\sum_i s_i, \tag{6}$$
$$I_j = \sqrt{\frac{1}{M}\sum_e (N_{e,j} - N_j / M)^2}.$$

These are average waiting time (simulation units), long-wait proportion, energy proxy (floors/request), average stop proxy (units/request), and load imbalance (standard deviation of request counts). Lower is better for each. Energy proxy is not measured electricity consumption, and long-wait thresholds are scenario-specific preferences. Episodes receive equal weights. A mean $\hat{x}$ over n episodes has interval $\tilde{x} \pm t_{0.975,n-1} s_x / \sqrt{n}$; algorithm contrasts use within-episode paired differences. Baseline intervals capture request variation under fixed trained policies, not retraining variation, and are not multiplicity-corrected.

For auxiliary scores, let x_{acm} be algorithm a 's mean on metric m in scenario c, and let $x_{cm}^{min}, x_{cm}^{max}$ be the extremes across algorithms:

$$z_{acm} = \frac{x_{cm}^{max} - x_{acm}}{x_{cm}^{max} - x_{cm}^{min}}, \quad S_{ac} = \frac{100}{5}\sum_{m=1}^{5} z_{acm}, \quad S_a^{cross} = \frac{1}{6}\sum_{c=1}^{6} S_{ac}. \tag{7}$$

Set $z_{acm} = 1$ when the range is zero. Equal metric and scenario weights are transparent conventions, not objective optima; scores depend on the comparator set and may amplify small differences. No preset

adaptability constant or additional waiting-rank term is included.

Supplementary experiments use 32 independently retrained policies per scenario, each with 40 validation and 40 test episodes. Comparators include Q-selection without manual corrections, a validation-selected fixed scheme, random scheme selection, the untrained Passenger-First default, NC, and RH. The fixed, random, and default variants retain manual corrections. Validation selects the fixed scheme with highest mean cumulative reward; testing never selects or trains a policy. The statistical unit is a repetition's 40-episode test mean, giving 32 paired observations, not 1,280 independently trained policies. Holm correction covers all $6\times6\times5=180$ two-sided metric comparisons at adjusted $p<0.05$. Training, validation, and test request seeds are disjoint, with base seed $2000000+2000000c+10000b$ for scenario c and repetition b, and offsets 0, 1000, and 2000 plus the episode index. Original dispatch and learning rules are unchanged; baseline and supplementary results are not pooled.

3. Results

FCFS, NC, SCAN, SZ, DD, RH, and Q denote the methods defined above, with Q denoting Q-AWED. The following baseline comparisons use independently generated test episodes after one training run per scenario; the supplementary retraining results are reported separately.

3.1. Waiting performance across six scenarios

Table 4. Average waiting time (episode means; simulation time units)

Method	Up-peak	Down-peak	Short-trip random	Hotspot floors	Three-phase commuting	Residential double-peak
FCFS	128.36	109.23	83.60	83.07	72.84	118.11
NC	127.25	102.45	64.63	58.47	66.71	115.17
SCAN	127.80	105.24	74.96	69.62	68.60	116.44
SZ	128.21	109.06	73.22	74.26	70.47	117.89
DD	127.71	104.09	69.17	69.36	67.98	116.11
RH	127.38	102.88	67.74	59.86	67.36	115.44
Q-AWED	127.32	103.12	66.25	61.24	67.90	115.31

Q-AWED's average waiting time ranks second or third in every scenario. Relative to FCFS, reductions are 0.81%, 5.59%, 20.75%, 26.27%, 6.78%, and 2.37% in scenario order. Improvements are largest for Short-Trip Random and Hotspot Floors, where distance and backlog information help avoid inefficient assignments. NC nevertheless has lower average waiting time in all six scenarios, so these benefits are not uniquely attributable to Q-learning. Paired intervals in **Figure 1** show why rankings alone are insufficient when mean differences are small.

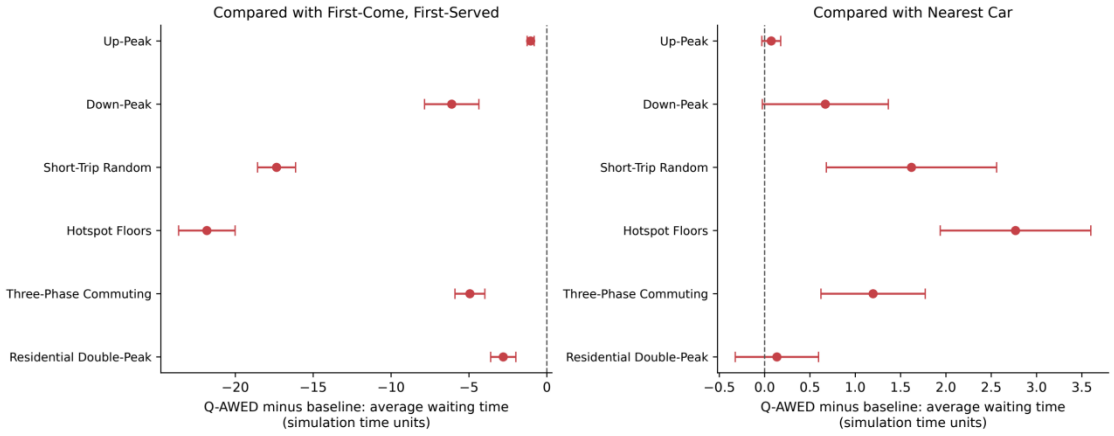


Figure 1. Paired average waiting-time differences with 95% confidence intervals. Negative values favor Q-AWED; intervals describe request variation under fixed trained policies.

Long-wait proportions remain high: Q-AWED records 92.80%, 84.85%, 88.56%, 92.12%, 89.12%, and 93.73% in scenario order, compared with NC's 92.86%, 84.52%, 88.50%, 91.66%, 89.19%, and 93.73%. This reflects substantial accumulation under serial service without shared rides, as well as arrival intensity and trip distance. Changing the threshold changes classification, not service speed; cross-scenario proportions must be interpreted with their thresholds.

3.2. Energy proxy, stops, and load balancing

Table 5. Energy proxy (mean floors traveled per request)

Method	Up-peak	Down-peak	Short-trip random	Hotspot floors	Three-phase commuting	Residential double-peak
FCFS	12.85	13.01	5.30	11.05	13.96	12.25
NC	12.43	12.58	4.29	8.67	13.23	11.98
SCAN	12.63	12.78	4.83	9.70	13.44	12.09
SZ	12.66	12.92	4.69	10.18	13.58	12.17
DD	12.55	12.71	4.51	9.68	13.36	12.08
RH	12.47	12.61	4.46	8.81	13.27	12.00
Q-AWED	12.46	12.64	4.40	8.95	13.35	12.01

The energy proxy combines pickup and passenger travel distances. Since the latter are fixed by shared requests, differences mainly reflect pickup assignments. Q-AWED has a lower proxy than FCFS in all six scenarios but a higher proxy than NC. Choosing more distant elevators to balance assignments may increase waiting and travel, so balancing must be judged jointly with these costs. See **Figure 2**.

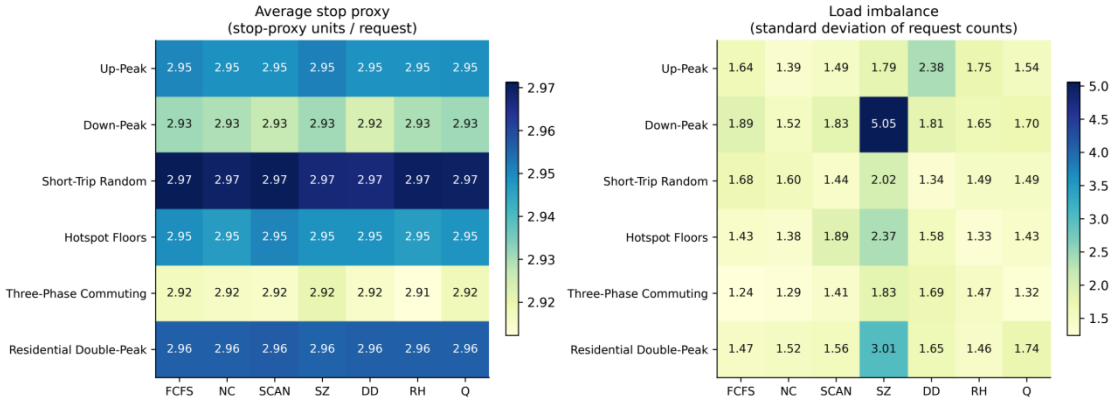


Figure 2. Average stop proxy and load imbalance. Colors are scaled separately within each panel; cell values report the original numerical means.

The stop proxy remains between 2 and 3 by construction; small differences mainly indicate whether elevators were busy at assignment, not physical stops saved through shared rides. Assignment-count balance likewise differs from actual passenger occupancy.

3.3. Cross-scenario performance and training benefit

Across the six equally weighted scenarios and five equally weighted metrics, the auxiliary scores are NC 84.09, RH 82.17, Q-AWED 76.78, DD 70.87, SCAN 59.59, FCFS 29.80, and SZ 25.91. Q-AWED ranks third overall, with a mean waiting-time rank of 2.50 and worst rank of third. These results describe separately trained scenario configurations, not direct transfer of one Q-table to untested buildings. Raw performance remains more informative than a preference-dependent score.

In the supplementary 32-repetition experiments, Q-AWED reduces average waiting time by 0.06–3.88% relative to the untrained default policy, with all six differences passing Holm correction. Short-Trip Random and Hotspot Floors improve by approximately 3.88% and 3.63%, respectively. No corrected waiting-time difference is found against random switching, while the validation-selected fixed scheme and NC have lower waiting and energy-proxy values in all six scenarios. Thus, the measured training benefit is improvement over the default policy, not superiority over every comparator.

4. Conclusion

4.1. Main findings

This article introduces a Q-AWED (Q-learning Adaptive Weighted Elevator Scheduling) approach to resolve the problem of which goals should be given priority at present, as well as specify which elevator should be assigned to requests. Q-learning, based on traffic conditions, chooses a weight scheme using a weighted dispatch cost function, travel distance, stop proxy, load, and direction to dispatch a specific elevator. Therefore, Q-learning sees the scheduling optimization as a decision process through reward feedback to modify object preferences instead of a fixed, single service sequence.

Q-AWED has a significant optimization on waiting time compared to simple algorithms such as First-Come, First-Served, their average waiting time decreased by 20.75% and 26.27% in Short-Trip Random and Hotspot Floors scenarios. Compared with allocating tasks based solely on the order of requests, Q-AWED

will consider elevators' position and pending tasks, reducing irrational allocation in both decentralized requests and concentrated hot-spots. Among seven main elevator dispatching algorithms, Q-AWED's average waiting time is at the second place or the third place, indicating it can be the former place under different requests after several training and configurations separately for each scenario.

Q-AWED's advantage is that it expresses the trade-offs explicitly among multiple objectives. Different from First-Come, First-Served and Static Zoning, this approach allows a single elevator to show its weighted objectives: priority for waiting, energy conservation priority, and load balancing, helping us trace back how the plan affects energy proxy and feedback.

Disclosure statement

The author declares no conflict of interest.

References

[1] Ruokokoski M, Sorsa J, Siikonen M, et al., 2016, Assignment Formulation for the Elevator Dispatching Problem with Destination Control and Its Performance Analysis. European Journal of Operational Research, 252: 397–406.

[2] Kompauer J, 2025, Optimizing Elevator Control with a Destination Registration System, thesis, TU Wien.

[3] Han Y, Wang S, Ye Y, et al., 2026, Simulation Study of Elevator Group Control Scheduling Based on Real-Time Occupancy Awareness. Journal of System Simulation, 38(7): 1993–2006.

[4] Zhu R, 2024, Research on Optimizing Elevator Operation Management Strategy Based on Mathematical Model Algorithm. SHS Web of Conferences, 200: 02010.

[5] Lin Y, Wang Y, Zhu J, et al., 2021, A Model and a Task Scheduling Method for Double-Deep Tier-Captive SBS/RS With Alternative Elevator-Patterns. IEEE Access, 9: 146378–146391.

[6] Yamauchi T, Ide R, Sugawara T, 2019, Fair and Effective Elevator Car Dispatching Method in Elevator Group Control System Using Cameras. Procedia Computer Science, 159: 455–464.

[7] Gharbi A, 2024, Exploring Heuristic and Optimization Approaches for Elevator Group Control Systems. Applied Sciences, 14: 995.

[8] Cao Z, Guo R, Tuguinay C, et al., 2022, Application of Deep Q Learning with Simulation Results for Elevator Optimization. arXiv.

[9] Al-Sharif L, Yang Z, Hakam A, et al., 2018, Comprehensive Analysis of Elevator Static Sectoring Control Systems Using Monte Carlo Simulation. Building Services Engineering Research and Technology, 39(5): 518–539.

[10] Batosalem A, Gaba J, Ngo J, et al., 2019, Static Zoning Division Elevator Traffic Simulation Using Agent-Based Modeling. Proceedings of the 2019 Winter Simulation Conference, 145–156.

[11] Sorsa J, Ehtamo H, Siikonen M, et al., 2009, The Elevator Dispatching Problem, manuscript submitted to Transportation Science.

Publisher's note

Bio-Byword Scientific Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.