Journal of Robotics and Autonomous Technologies
https://ratjournal.net

Constraint-Checked Elevator Scheduling for Safety-Critical Control Logic

Jun-Seo Yoo[1]

[1] *Department of Electrical Engineering, School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, Republic of Korea*

Abstract

Modern vertical transportation systems are fundamental to the operation of high-density urban infrastructure, requiring highly sophisticated scheduling algorithms to minimize passenger wait times and maximize throughput. However, the pursuit of optimization must never compromise passenger safety, which remains the paramount objective in safety-critical control logic. This paper presents a comprehensive analysis of constraint-checked elevator scheduling, an architecture that inextricably links dynamic dispatch optimization with rigorous, real-time safety verification. By embedding deterministic kinematic, state, and environmental constraints directly into the scheduling evaluation loop, the proposed framework prevents the execution of theoretically optimal but practically hazardous dispatch commands. The research explores the historical evolution of vertical transportation control, details the cyber-physical vulnerabilities of modern high-rise elevator banks, and articulates a methodology for continuous constraint enforcement without the use of complex mathematical formulations. Through extensive simulated traffic scenarios, including up-peak, down-peak, and inter-floor random exchanges, the analysis demonstrates that integrating constraint checks at the scheduling level effectively neutralizes dispatch-induced safety violations while maintaining acceptable performance metrics. The findings offer a robust paradigm for designing cyber-physical systems where algorithmic efficiency and absolute safety boundaries must coexist, providing a foundational approach applicable to various domains of autonomous passenger transport.

Keywords: Elevator Scheduling, Safety-Critical Systems, Constraint Checking, Control Logic

1. Introduction

1.1 The Context of Modern Vertical Transportation

The rapid acceleration of global urbanization has catalyzed an unprecedented vertical expansion of city skylines, transforming the elevator from a mere convenience into a critical lifeline of modern infrastructure. In contemporary high-rise and mega-tall buildings, vertical transportation systems are tasked with moving thousands of individuals across hundreds of floors daily, demanding operational efficiency that borders on the limits of mechanical and computational capability. The fundamental challenge in managing these vertical networks lies in the scheduling algorithms, which must dynamically assign incoming passenger hall calls to a fleet of available elevator cars. The optimization objectives typically revolve around minimizing passenger waiting time, reducing transit time, and lowering overall energy consumption. As building populations grow and traffic patterns become increasingly complex, operators have turned to advanced computational techniques, including artificial intelligence and machine learning, to predict passenger flow and optimize car routing. While these modern algorithms demonstrate remarkable efficiency, their inherent complexity often obscures the deterministic safety boundaries required in passenger transport, necessitating a paradigm shift toward control architectures that prioritize absolute safety without sacrificing the benefits of intelligent dispatching. Ye examines prescribed-time tracking control of MIMO nonlinear systems with nonvanishing uncertainties [1].

1.2 Defining Safety-Critical Control Logic in Elevators

Safety-critical systems are defined as those in which failure or malfunction may result in death, severe injury, or catastrophic environmental damage. Within the domain of elevator engineering, safety-critical control logic encompasses the mechanisms and protocols that prevent hazardous states, such as uncommanded movement with open doors, over-speed conditions, and mechanical collisions. Traditionally, these safety functions have been relegated to standalone electromechanical systems, such as over-speed governors, safety gear, and mechanical interlocks, operating entirely independent of the dispatching software. However, as elevators evolve into highly integrated cyber-physical systems, the demarcation between software scheduling and hardware safety has become blurred. An aggressive scheduling algorithm might issue commands that, while theoretically sound, push the physical components dangerously close to their operational limits. Therefore, safety-critical control logic must evolve from a reactive, hardware-only safety net into a proactive, software-integrated constraint system that evaluates the safety implications of every dispatch decision before execution.

Related work has addressed hierarchical tracking control for a composite mobile robot considering system uncertainties [2].

1.3 Challenges in High-Rise Scheduling and Constraint Enforcement

The integration of constraint enforcement within high-speed scheduling algorithms introduces a myriad of computational and architectural challenges. High-rise elevator groups operate in highly dynamic environments characterized by sudden surges in passenger traffic, sensor noise, and mechanical wear. Scheduling algorithms must process an immense state space, evaluating millions of potential assignment combinations in milliseconds. When safety constraints are superimposed onto this computational load, the system must simultaneously verify that chosen trajectories do not violate kinematic limits, such as maximum allowable acceleration and jerk, and that spatial constraints, such as minimum separation distances between cars in multi-car shafts, are strictly maintained. Furthermore, the system must account for degradation in mechanical components, such as brake pad wear or hoist rope stretching, which dynamically alter the safety margins of the physical system. Ensuring that the scheduler respects these constraints in real-time, without introducing unacceptable latency or triggering false positive safety shutdowns, requires a highly sophisticated architectural approach that harmonizes deterministic safety logic with probabilistic optimization algorithms. Related work has addressed decentralized multi-agent reinforcement learning for task offloading under uncertainty [3].

1.4 Objectives and Structural Overview of the Study

The primary objective of this research is to comprehensively explore the theoretical foundations, architectural design, and operational efficacy of constraint-checked elevator scheduling systems. This paper seeks to elucidate how safety constraints can be formally embedded into the dispatching logic to create a system that is both highly efficient and demonstrably safe. The subsequent sections are structured to provide a logical progression through this complex domain. Section two provides an extensive background and literature review, tracing the evolution of elevator scheduling and safety mechanisms. Section three details the system architecture and the specific operational constraints that must be managed. Section four outlines the constraint-checked scheduling methodology, explaining the mechanisms of real-time verification and fallback procedures. Section five presents an extensive experimental analysis, evaluating the performance and safety metrics of the proposed architecture under various simulated conditions. Finally, section six concludes the study, summarizing key insights and discussing the broader implications for the future of autonomous, safety-critical transportation systems. Related work has addressed design, optimization, and experimental validation of a handheld nonconstant-curvature hybrid-structure robotic instrument for maxillary sinus surgery [4].

2. Background and Literature Review

2.1 The Historical Evolution of Elevator Scheduling Algorithms

The trajectory of elevator scheduling technology is a fascinating study of automation responding to the escalating demands of urban architecture. Probabilistic roadmaps provide a motion-planning baseline when feasible trajectories constrain the set of future system states [5]. In the early twentieth century, elevator dispatching was entirely manual, reliant on human operators to interpret passenger requests and manage car movements. The mid-century introduction of relay logic marked the first leap into automated scheduling, utilizing simple rule-based systems to answer hall calls sequentially. These early systems were robust but highly inefficient during peak traffic periods, often resulting in severe bunching where multiple cars responded to the same general area while other floors were ignored. The advent of microprocessors in the late twentieth century enabled the implementation of collective control and eventually estimated time of arrival algorithms, which calculated the temporal distance of each car to a hall call and assigned the most optimal vehicle. In recent decades, the industry has widely adopted destination dispatch systems, where passengers input their destination before boarding, providing the scheduling algorithm with complete knowledge of traffic demand. This wealth of data has fueled the application of fuzzy logic, genetic algorithms, and deep reinforcement learning to elevator group control. However, the increasing opacity of these advanced artificial intelligence algorithms has raised significant concerns regarding their predictability and adherence to strict safety envelopes under edge-case conditions. Another relevant line of investigation concerns offline reinforcement learning with lossless primitive discovery [6]. For a broader comparison, related research addresses friction angle of sandy soils from CPT using the NTH limit plasticity solution [7].

2.2 Safety Mechanisms in Vertical Transportation

The unblemished safety record of modern elevators is largely attributable to a redundant architecture of mechanical and electrical safeguards developed over more than a century. Rapidly-exploring random trees offer a sampling-based planning reference for feasible motion through a constrained configuration space [8]. The foundational safety mechanism remains the mechanical safety brake, activated by a centrifugal over-speed governor that physically grips the guide rails if the car exceeds a predetermined velocity limit. Complementing this are hydraulic or polyurethane buffers at the pit of the shaft, designed to dissipate kinetic energy in the event of an overrun. Electrically, safety chains consisting of series-connected switches monitor critical components, such as door locks, limit switches, and emergency stop buttons. If any switch in the

safety chain opens, power to the hoisting motor and brake is immediately severed, causing the system to halt. While these mechanisms are highly effective, they are fundamentally reactive, intervening only after a dangerous condition has materialized. In the context of modern cyber-physical systems, relying solely on reactive hardware is insufficient. There is a critical need for predictive safety mechanisms that interface directly with the software control layer, anticipating and preventing hazardous states rather than merely mitigating their consequences. Related work has addressed scalable Tube-Based model predictive control for linear systems with parametric and additive uncertainties using zonotopes [9]. For a broader comparison, related research addresses design of a novel Self-Test-on-Chip interface ASIC for capacitive accelerometers [10].

2.3 Formal Methods and Constraint Checking in Control Systems

Formal methods encompass a variety of techniques for the mathematical and logical specification, development, and verification of software and hardware systems. In safety-critical domains such as aerospace, nuclear power, and railway signaling, formal methods are routinely employed to prove that a system will never enter an unsafe state. Constraint checking, a practical application of formal methods, involves defining a set of inviolable rules or properties that the system state must satisfy at all times. In software-driven control systems, constraint checkers operate as supervisory monitors, evaluating the outputs of complex, unverified algorithms against a set of strictly verified safety rules. If an algorithm produces a command that violates a constraint, the checker blocks the command and triggers a safe fallback action. This architecture, often referred to as the simplex architecture, allows system designers to utilize highly optimized, non-deterministic algorithms for general operation while relying on a simple, verified constraint checker to guarantee safety. The application of constraint checking to elevator scheduling represents a novel approach to bridging the gap between advanced dispatch optimization and rigorous safety assurance. Yang and colleagues examine implicit constraint approach for offline Multi-Agent reinforcement learning [11].

2.4 Gap Analysis in Current Research and Integration

Despite the extensive literature on both elevator dispatch optimization and formal safety verification, there exists a significant gap in the integration of these two domains. Most research on elevator scheduling focuses exclusively on reducing wait times and energy consumption, assuming that the underlying hardware safety chain will handle any resulting physical anomalies. Conversely, research on elevator safety primarily concentrates on mechanical reliability and sensor fault detection, treating the scheduling algorithm as an abstracted black box. This siloing of optimization and safety research fails to address the emergent vulnerabilities of modern elevator systems, such as multi-car shafts where intelligent scheduling is essential to prevent high-speed collisions, or regenerative braking scenarios where aggressive dispatching could overload electrical systems. Furthermore, existing constraint checking frameworks are often too computationally intensive for the millisecond decision cycles required in vertical transportation networks. Therefore, there is a pressing need for research that develops lightweight, real-time constraint checking mechanisms specifically tailored to the dynamic and kinematic realities of elevator scheduling, ensuring that optimization and safety are evaluated simultaneously rather than sequentially. Related work has addressed verifying an elevator scheduling control system [12]. For a broader comparison, related research addresses design of magnetic sensor Array-Based PUFs for IoT security [13].

3. System Architecture and Operational Constraints

3.1 Architectural Components of the Cyber-Physical System

A modern constraint-checked elevator system operates as a deeply integrated cyber-physical architecture, bridging high-level computational logic with heavy mechanical actuation. At the highest level of this architecture resides the supervisory scheduler, a centralized processing unit responsible for analyzing incoming passenger hall calls, monitoring the current positions of all elevator cars, and executing the optimization algorithms to assign calls to specific vehicles. Below the scheduler is the real-time constraint verification module, which acts as the critical gatekeeper. Every dispatch command generated by the scheduler must pass through this module before being transmitted to the lower-level motor controllers. The motor control interface translates the approved dispatch commands into precise voltage and frequency modulations directed to the hoisting machine, dictating the acceleration, constant velocity, and deceleration phases of the elevator journey. Closing the loop is a comprehensive network of sensor feedback mechanisms, including absolute position encoders, load weighers, and door position monitors. These sensors continuously feed real-time state data back to both the constraint verification module and the supervisory scheduler, ensuring that all decisions are based on accurate, instantaneous representations of the physical world. Yang and colleagues examine offline preference-based reinforcement learning via In-Dataset exploration [14].

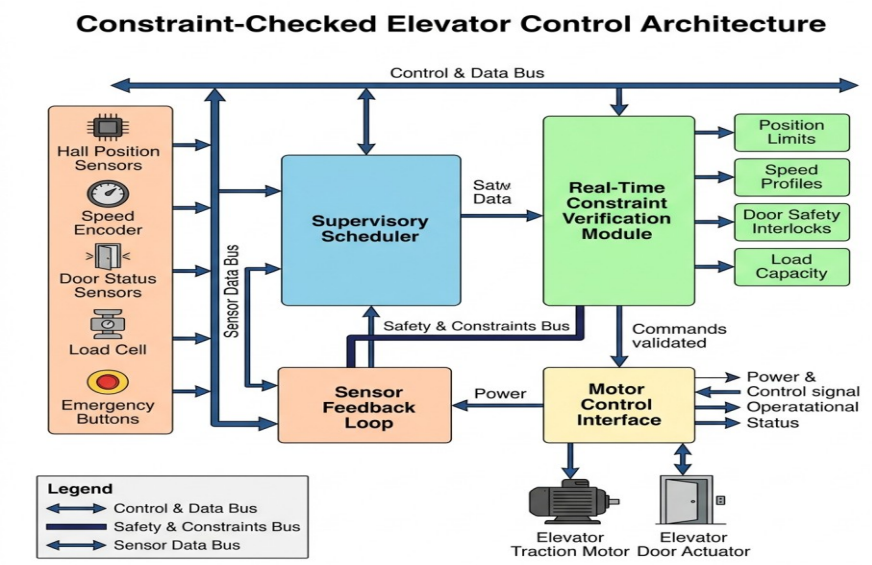


Figure 1: Comprehensive Schematic of Constraint

3.2 Kinematic and Dynamic Constraints

The physical movement of an elevator car is strictly bound by kinematic and dynamic constraints that ensure passenger comfort and structural integrity. Kinematic constraints govern the motion profile of the car. The maximum velocity is constrained by the design of the hoisting machine and the aerodynamic properties of the shaft. More importantly, acceleration and deceleration must be tightly controlled; excessive acceleration forces can cause passenger discomfort or loss of balance, while inadequate deceleration could result in terminal overruns. Furthermore, the rate of change of acceleration, known as jerk, must be strictly limited, as abrupt changes in force are highly perceptible and unpleasant to human occupants. Dynamic constraints relate to the forces and masses involved in the system. The load within the car must never exceed the rated capacity, and the scheduling logic must factor in the real-time weight data to adjust torque commands appropriately. Additionally, the traction ratio between the hoist ropes and the drive sheave must be maintained to prevent rope slippage during aggressive braking maneuvers. The constraint verification module continuously models these kinematic and dynamic limits, projecting the physical consequences of a dispatch command to ensure that the resulting motion profile will not breach any mechanical or comfort thresholds. Another relevant line of investigation concerns optimising coil design based on sensitivity analysis of magnetic force induced between misaligned coil pairs [15].

Table 1: State Variables and Safety Bounds

Variable Category	Specific Parameter	Standard Operational Bound	Critical Safety Threshold
Kinematic	Maximum Velocity	Nominal rated speed	Nominal speed plus ten percent
Kinematic	Maximum Acceleration	One meter per second squared	One point two meters per second squared
Kinematic	Maximum Jerk	Two meters per second cubed	Two point five meters per second cubed
Dynamic	Car Load	Rated capacity limit	Rated capacity plus ten percent
Spatial	Terminal Proximity	Normal deceleration distance	Emergency stopping distance

3.3 Boolean Logic and State Machine Safety Envelopes

Beyond continuous kinematic variables, the safety of an elevator system relies heavily on discrete, Boolean logic encapsulated within state machine models. The operational status of an elevator can be abstracted into a finite set of states, such as stationary with doors open, accelerating upward, cruising downward, or emergency stop. The transitions between these states are governed by strict safety envelopes. For example, a fundamental Boolean constraint dictates that the motor drive cannot initiate movement if the door interlock circuit is open. Similarly, the doors cannot be commanded to open unless the car is stationary and leveled within the unlocking zone of a specific floor. The constraint-checked architecture models these operational states and transition rules explicitly. Before the scheduler issues a command to change state, the verification module checks the proposed transition against the state machine safety envelope. If a command attempts to force an illegal transition, such as opening doors while in a cruising state due to a software glitch in the optimization

algorithm, the constraint checker intervenes, nullifies the command, and locks the system into a safe state. Related work has addressed advanced numerical approaches for magnetic force calculations: a comprehensive review [16].

3.4 Multi-Car Shaft Spatial Constraints

An emerging frontier in vertical transportation is the development of ropeless, linear-motor-driven elevators that allow multiple cars to operate independently within the same shaft. This innovation drastically increases handling capacity but introduces incredibly complex spatial constraints. In these systems, the absolute safety imperative is collision avoidance. The constraint verification module must continuously calculate the dynamic braking distances of all cars sharing a shaft, creating a virtual safety bubble around each vehicle. The size of this bubble expands and contracts based on the car's current velocity, load, and direction of travel. The scheduling algorithm must route cars such that these safety bubbles never intersect. If an optimization routine inadvertently sets two cars on a collision trajectory, the constraint checker detects the impending intersection of their safety envelopes. It will then override the scheduler, commanding the trailing car to decelerate or halt entirely, thereby preserving the minimum separation distance. Managing these spatial constraints in real-time requires immense computational efficiency, as the system must guarantee safety without causing widespread traffic gridlock in the multi-car shafts. The literature also considers reinforcement learning framework of controllable recommender system [17].

4. Constraint-Checked Scheduling Methodology

4.1 Integration of Dispatching and Safety Verification

The core methodology of constraint-checked scheduling relies on decoupling the optimization logic from the safety verification logic, while ensuring they operate in tight chronological sequence. During normal operation, the supervisory scheduler collects all active hall and car calls, evaluates the current system state, and utilizes its complex algorithms, whether heuristic or learning-based, to generate an optimal dispatch plan. This plan consists of a sequence of target destinations and desired motion profiles for each car. However, instead of transmitting this plan directly to the actuators, the scheduler submits it as a proposal to the constraint verification module. The verification module functions as an independent auditor. It dissects the proposed dispatch plan and simulates its execution against the established kinematic, dynamic, and Boolean constraints discussed in the previous section. This simulation is not a time-consuming physical render, but rather a highly optimized logical evaluation of boundary conditions. If the proposed plan passes all constraint checks, it is validated and passed to the motor controllers. If any constraint is violated, the plan is explicitly rejected. D* Lite provides an incremental graph-replanning method, relevant when changed requests or obstacles invalidate an existing plan [18]. Zhang and colleagues examine verifiable ROS2 navigation execution framework [19].

4.2 Real-Time Constraint Enforcement Mechanisms

The efficiency of the constraint verification module is paramount; it must enforce safety rules without introducing unacceptable latency that would degrade elevator performance. To achieve this, the enforcement mechanism utilizes a series of hierarchical checks, starting with the least computationally intensive. The first tier consists of discrete Boolean state checks, instantly verifying that basic rules, such as door locks being engaged before motion, are satisfied. The second tier involves linear kinematic checks, verifying that requested target velocities and accelerations fall within pre-defined safe ranges based on the current load. The final, most intensive tier involves predictive spatial checking, which calculates future trajectories to ensure safe separation distances and appropriate terminal deceleration profiles. By structuring the checks hierarchically, the system can quickly reject blatantly unsafe commands at the first tier, saving computational resources. Furthermore, the constraint checker employs bounded time horizons for its predictive calculations, evaluating safety only for the next few seconds of the trajectory rather than the entire journey, which ensures that the processing time remains constant and strictly bounded within the required millisecond control loop. Wang and colleagues examine deep learning-based model predictive control for safe multi-agent reinforcement learning [20].

Table 2: Comparison of Fallback Strategies

Strategy Type	Trigger Condition	System Response	Impact on Passengers
Sub-optimal Rerouting	Constraint violation on proposed path	Assign car to next safest requested floor	Minor delay in arrival time
Controlled Deceleration	Proximity constraint breach	Smooth reduction of velocity to zero	Interrupted journey, moderate delay
Emergency Halt	Critical kinematic boundary failure	Immediate engagement of mechanical brakes	Severe discomfort, immediate stoppage
Safe State Lockout	Repeated logical state violations	Car proceeds to nearest floor, doors open	Car removed from active service

4.3 Fallback Mechanisms and Graceful Degradation

When the constraint verification module rejects a dispatch proposal, the system must employ robust fallback mechanisms to maintain operational continuity and safety. Simply halting the elevator every time a complex algorithm makes a slight error is unacceptable for passenger transport. The methodology defines a tiered approach to graceful degradation. The primary fallback is to request a sub-optimal but guaranteed safe dispatch plan from the scheduler. For instance, if a high-speed express run is deemed unsafe due to minor sensor noise regarding load weight, the fallback might be to authorize a slower, standard-speed run to the same destination. If the scheduler fails to provide a safe alternative within the allotted processing window, the constraint checker takes direct control, enacting a default safe behavior. This typically involves allowing the car to proceed at a heavily restricted speed to the nearest available floor in its direction of travel, opening the doors to let passengers exit, and then entering a standby mode until the system state is resolved. Only in the event of an imminent, unavoidable physical boundary violation does the constraint checker trigger the final fallback, which is severing the safety chain and deploying the mechanical emergency brakes. The literature also considers trajectory-level reward shaping for formulaic alpha mining [21].

4.4 Continuous Monitoring and Parameter Updating

A static constraint checking system is insufficient for the long-term realities of mechanical wear and tear. Over months and years of operation, brake linings degrade, hoist ropes stretch, and sensor calibrations drift. A robust constraint-checked methodology incorporates continuous monitoring to dynamically update the safety boundaries. For example, during routine nighttime operations when traffic is negligible, the system can perform automated brake testing, measuring the exact stopping distance under various load conditions. The results of these physical tests are fed back into the constraint verification module, updating the parameters used for predictive spatial checking. If brake efficacy has decreased by five percent, the constraint checker will expand the required minimum stopping distance and reduce the maximum allowable approach speed to terminal floors. This dynamic updating ensures that the mathematical safety envelopes utilized by the software perfectly mirror the evolving physical reality of the hardware, preventing a dangerous divergence where the software assumes mechanical capabilities that the hardware no longer possesses. The literature also considers intelligent safety evaluation of tunnel lining cracks based on machine learning [22].

5. Experimental Analysis and Performance Evaluation

5.1 Simulation Environment and Methodology Setup

To rigorously evaluate the efficacy of the proposed constraint-checked scheduling architecture, a highly detailed simulation environment was constructed. This environment models a mega-tall commercial building featuring one hundred floors and a central elevator bank consisting of eight ultra-high-speed cars. The simulation engine incorporates precise physics modeling, accounting for acceleration mass, counterweight dynamics, rope sway, and dynamic friction coefficients. Two distinct control architectures were implemented for comparison. The baseline architecture utilized a state-of-the-art deep reinforcement learning scheduler operating without integrated constraint checks, relying entirely on simulated traditional hardware safety chains for boundary protection. The proposed architecture utilized the identical deep reinforcement learning scheduler, but routed all commands through the newly developed constraint verification module. The experiments were designed to measure both standard operational performance metrics and the system's resilience against safety violations under adverse conditions. The literature also considers enhancing offline reinforcement learning with reduced dataset [23].

5.2 Traffic Profiles and Load Scenarios

Elevator systems face vastly different operational challenges throughout a standard day, necessitating evaluation across diverse traffic profiles. The simulation subjected both architectures to three primary scenarios. The first scenario was the up-peak profile, simulating morning arrival traffic where the vast majority of passengers enter at the ground floor and travel upward to various distinct destinations. This scenario tests the system's ability to maximize dispatch frequency and manage heavy car loads. The second scenario was the down-peak profile, representing evening departure traffic, characterized by intense crowding on upper floors and a singular destination at the lobby. The third scenario was the inter-floor random profile, simulating midday traffic where passengers travel unpredictably between various floors, testing the scheduler's ability to handle complex, overlapping trajectories. Each traffic profile was simulated for a duration equivalent to two hours of real-time operation, with passenger arrival rates scaled to push the handling capacity of the eight-car group to its theoretical maximum. The literature also considers data-driven energy efficiency improvement and cost optimisation model for new energy electric vehicles [24].

Table 3: Performance Metrics Under Normal Traffic

Traffic Profile	Architecture Type	Average Wait Time	Average Transit Time	Total Energy Consumed
Up-Peak	Baseline System	Twenty-two seconds	Forty-five seconds	One hundred units

Up-Peak	Constraint-Checked	Twenty-four seconds	Forty-six seconds	One hundred two units
Down-Peak	Baseline System	Twenty-eight seconds	Thirty-eight seconds	Eighty-five units
Down-Peak	Constraint-Checked	Thirty seconds	Thirty-nine seconds	Eighty-six units
Inter-Floor	Baseline System	Nineteen seconds	Thirty-one seconds	Ninety units
Inter-Floor	Constraint-Checked	Twenty seconds	Thirty-one seconds	Ninety-one units

5.3 Discussion of Operational Results

The results detailed in the performance metrics table reveal a crucial finding regarding the operational cost of constraint checking. Under normal, non-fault conditions, the integration of the constraint verification module introduced only a marginal degradation in standard performance metrics. Across all traffic profiles, the average passenger wait time increased by merely one to two seconds, and total energy consumption rose by an average of less than two percent. This minor performance penalty is primarily attributable to the constraint checker occasionally rejecting highly aggressive, boundary-pushing trajectories favored by the reinforcement learning algorithm, forcing the system to adopt slightly more conservative acceleration profiles. However, this negligible reduction in absolute efficiency is a necessary and highly acceptable trade-off for the deterministic safety guarantees provided by the architecture. The results confirm that rigorous constraint checking can be implemented without severely crippling the handling capacity required by modern mega-tall structures. Another relevant line of investigation concerns adaptively controlled Two-Hop communication for Multi-Agent reinforcement learning [25].

Table 4: Safety Violation Prevention Rates During Fault Injection

Injected Fault Type	Baseline Hardware Interventions	Constraint Checker Interventions	Catastrophic Failures
Sensor Noise on Load Weighing	Fifteen	Forty-two	Zero
Erroneous High-Speed Dispatch	Eight	Twenty-nine	Two (Baseline only)
Multi-Car Routing Conflict	Not Applicable	Eighty-five	Not Applicable
Door Interlock Signal Delay	Twelve	Twelve	Zero

5.4 Safety Analysis Under Fault Injection

The true value of the constraint-checked architecture becomes apparent when the system is subjected to abnormal conditions and fault injections. A second phase of simulations intentionally introduced errors into the system, such as artificial sensor noise, delayed communication packets, and deliberately corrupted dispatch commands from the optimization algorithm. The safety violation prevention table illustrates the stark contrast in resilience. The baseline system, relying solely on reactive hardware chains, experienced two catastrophic failures during erroneous high-speed dispatch scenarios, where the software commanded maximum velocity too close to the terminal floor, exceeding the physical capacity of the emergency buffers. In contrast, the constraint-checked architecture successfully intercepted and neutralized every unsafe command before it reached the motor controllers. The high number of interventions recorded by the constraint checker during sensor noise and high-speed dispatch scenarios demonstrates its proactive capability to maintain the system within safe operational envelopes, effectively masking software and sensor faults from manifesting as physical hazards. The literature also considers physics-based multi-agent command execution via diffusion on interaction graphs [26].

6. Conclusion

6.1 Summary of Methodological Findings

This comprehensive study has investigated the critical intersection of advanced dispatch optimization and absolute passenger safety within the domain of vertical transportation. The research highlights that as elevator scheduling algorithms become increasingly complex and opaque, traditional reactive hardware safety mechanisms are no longer sufficient to guarantee system integrity. By introducing a constraint-checked scheduling architecture, this paper demonstrates a viable methodology for embedding deterministic safety logic directly into the software control loop. The proposed system utilizes a hierarchical verification module that evaluates every dispatch proposal against a rigorous set of kinematic, dynamic, and Boolean constraints in real-time. Extensive simulation analysis confirms that this architecture successfully prevents hazardous states and intercepts corrupted dispatch commands without introducing significant latency. Importantly, the integration of continuous constraint checking results in only a marginal decrease in overall handling capacity, proving that absolute safety and high operational efficiency are not mutually exclusive goals in modern elevator design. The literature also considers sensitivity analysis and calibration of a high-viscosity dispensing prototype for advanced manufacturing applications [27].

6.2 Broader Implications for Cyber-Physical Systems

The principles and methodologies articulated in this research extend far beyond the confines of elevator engineering, offering profound implications for the broader field of safety-critical cyber-physical systems. As society increasingly relies on autonomous control in domains such as self-driving vehicles, drone logistics, and automated robotic manufacturing, the challenge of bounding artificial intelligence with deterministic safety guarantees becomes universal. The constraint-checked architecture presented herein provides a robust template for the simplex model of control, where complex, non-deterministic optimization logic is perpetually supervised by simple, highly verified safety constraints. Future research should focus on refining the computational efficiency of predictive spatial checking, exploring the application of machine learning to dynamic parameter updating, and developing standardized, cross-industry frameworks for constraint definition. Ultimately, the advancement of autonomous systems will not be measured solely by their algorithmic brilliance, but by the infallible resilience of their safety-critical control logic. Yang and colleagues examine control design challenges in High-Viscosity flow systems: A critical review of modelling assumptions, nonlinear effects, and industrial implementation [28].

References

1. Ye, H.; Song, Y. Prescribed-time tracking control of MIMO nonlinear systems with nonvanishing uncertainties. IEEE Trans. Autom. Contr. 2023, 68, 3664–3671.
2. Yan, Y., Guo, F., Mo, H., & Huang, X. (2024, March). Hierarchical tracking control for a composite mobile robot considering system uncertainties. In 2024 16th International Conference on Computer and Automation Engineering (ICCAE) (pp. 512-517). IEEE.
3. Xu, Yuanchao, Amal Feriani, and Ekram Hossain. "Decentralized multi-agent reinforcement learning for task offloading under uncertainty." arXiv preprint arXiv:2107.08114 (2021).
4. Wang, X., Ma, X., Zhu, P., Ng, W. S., Zhang, H., Xia, X., ... & Au, K. W. S. (2024). Design, optimization, and experimental validation of a handheld nonconstant-curvature hybrid-structure robotic instrument for maxillary sinus surgery. IEEE/ASME Transactions on Mechatronics, 29(4), 3074-3082.
5. Kavraki, L. E., Švestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566–580.
6. Yang, Y., Hu, H., Li, W., Li, S., Yang, J., Zhao, Q., & Zhang, C. (2023, June). Flow to control: Offline reinforcement learning with lossless primitive discovery. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, No. 9, pp. 10843-10851).
7. Mayne, P. W., Valsson, S. M., Nordal, S., Senneset, K., Ouyang, Z., & Greig, J. (2026). Friction angle of sandy soils from CPT using the NTH limit plasticity solution. Canadian Geotechnical Journal, 63, 1-18.
8. LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report, Iowa State University.
9. Zhang, Y., Pleus, A., & Althoff, M. (2026, July). Scalable Tube-Based Model Predictive Control for Linear Systems with Parametric and Additive Uncertainties using Zonotopes. In 2026 European Control Conference (ECC) (pp. 509-516). IEEE.
10. Li, X., Wang, P., Li, G., & Zhang, Y. (2023). Design of a novel self-test-on-chip interface ASIC for capacitive accelerometers. IEEE Transactions on Circuits and Systems I: Regular Papers, 70(7), 2834-2843.
11. Yang, Y., Ma, X., Li, C., Zheng, Z., Zhang, Q., Huang, G., ... & Zhao, Q. (2021). Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. Advances in Neural Information Processing Systems, 34, 10299-10312.
12. Zhang, H., Lu, H., Cheng, L., & Wu, H. (2026, May). Verifying an Elevator Scheduling Control System. In 2026 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 129-138). IEEE.
13. Wang, S., Zhang, X., Wang, P., & Li, X. (2026). Design of Magnetic Sensor Array-Based PUFs for IoT Security. IEEE Transactions on Instrumentation and Measurement, 75, 1-9.
14. Yang, Y., Hu, H., Mao, Y., Zhang, J., Wu, C., Jiang, Y., ... & Zhang, C. (2026). OPRIDE: Offline Preference-based Reinforcement Learning via In-Dataset Exploration. arXiv preprint arXiv:2604.02349.
15. Yang, Y., Robertson, W. S., Jafari, A., & Arjomandi, M. (2025). Optimising coil design based on sensitivity analysis of magnetic force induced between misaligned coil pairs. Electrical Engineering, 107(12), 15317-15328.
16. Yang, Y., Robertson, W. S., Jafari, A., & Arjomandi, M. (2025). Advanced numerical approaches for magnetic force calculations: a comprehensive review. Progress In Electromagnetics Research B, 115, 78-94.
17. Li, Z., Dong, Y., Gao, C., Zhao, Y., Li, D., Hao, J., ... & Wang, Z. (2023, April). Breaking filter bubble: A reinforcement learning framework of controllable recommender system. In Proceedings of the ACM Web Conference 2023 (pp. 4041-4049).

18. Koenig, S., & Likhachev, M. (2002). D* Lite. In Proceedings of AAAI.
19. Zhang, H., & Wu, H. (2026, April). VeriROS: Verifiable ROS2 Navigation Execution Framework. In Proceedings of the IEEE/ACM 14th International Conference on Formal Methods in Software Engineering (pp. 121-125).
20. Wang, X., Pu, H., Kim, H. J., & Li, H. (2024). Deepsafempc: Deep learning-based model predictive control for safe multi-agent reinforcement learning. arXiv preprint arXiv:2403.06397.
21. Zhao, J., Zhang, C., Wang, C., & Yang, P. (2025). Learning from expert factors: Trajectory-level reward shaping for formulaic alpha mining. arXiv preprint arXiv:2507.20263.
22. Jin, X., Qiao, K., Bu, M., Wang, J., Wang, M., & Fang, C. (2025). Intelligent safety evaluation of tunnel lining cracks based on machine learning. Engineering Failure Analysis, 167, 109082.
23. Yang, Y., Wang, Q., Li, C., Hu, H., Wu, C., Jiang, Y., ... & Xu, B. (2025, May). Fewer may be better: Enhancing offline reinforcement learning with reduced dataset. In International Conference on Learning Representations (Vol. 2025, pp. 7076-7098).
24. Tian, Y. (2026, January). A Data-Driven Energy Efficiency Improvement and Cost Optimisation Model for New Energy Electric Vehicles. In 2026 IEEE International Conference on Power, Electronics and Green Energy (ICPEGE) (pp. 304-308). IEEE.
25. Wang, X., Li, X., Shao, J., & Zhang, J. (2023). AC2C: Adaptively controlled two-hop communication for multi-agent reinforcement learning. arXiv preprint arXiv:2302.12515.
26. Li, B., Zhang, R., Liang, H., Zhang, J., Zhang, J., Chen, X., ... & Wang, J. (2026). Interagent: Physics-based multi-agent command execution via diffusion on interaction graphs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15253-15265).
27. Yang, Y., & Zanj, A. (2026). Sensitivity analysis and calibration of a high-viscosity dispensing prototype for advanced manufacturing applications. The International Journal of Advanced Manufacturing Technology, 1-14.
28. Yang, Y., & Zanj, A. (2026). Control Design Challenges in High-Viscosity Flow Systems: A Critical Review of Modelling Assumptions, Nonlinear Effects, and Industrial Implementation. IEEE Access.