Journal of Industrial and Management Optimization

https://www.aimspress.com/journal/jimo

JIMO, 22(8): 4088–4137.
DOI:10.3934/jimo.2026144
Received: 16 May 2026
Revised: 16 June 2026
Accepted: 21 July 2026
Published: 28 July 2026

Research article

Adaptive switching optimization for heterogeneous decision modules under operational uncertainty

Heon Kyun Shin[1,5], Chaehwa Lee[2], Jinho Cha[3], Minyoung Kil[4] and Jaehyuk Choi[6,*]

[1] C4ISR Systems Technology Planning Team, Korea Research Institute for Defense Technology Planning and Advancement (KRIT), Jinju, Republic of Korea

[2] Center for Army Analysis and Simulation, Republic of Korea Army, Gyeryong, Republic of Korea

[3] Department of Computer Science, Gwinnett Technical College, Lawrenceville, GA, USA

[4] C6 (Command, Control, Communications, and Cyber), ROK–U.S. Combined Forces Command, Pyeongtaek, Republic of Korea

[5] Department of AI Convergence Engineering, Gyeongsang National University, Jinju, Republic of Korea

[6] Department of Defense AI and Robotics, Korea National Defense University, Nonsan, Republic of Korea

* **Correspondence:** Email: checks@korea.kr.

Abstract: We developed a stochastic dynamic optimization framework for orchestrating heterogeneous decision modules under operational uncertainty. The modules may differ in quality, latency, reliability, and switching friction. We formulated their orchestration as a finite discounted Markov decision process in which the active module is part of the state and module selection is the control. The reward permits general nondecreasing latency disutility, including linear, threshold-like, power, and convex penalties. We established stationary optimality, switching and continuation regions, pairwise hysteresis, latency-driven preference shifts, an exact adaptive–greedy Bellman decomposition, zero-friction equivalence under action-independent exogenous dynamics, and near-optimality bounds for simple policies. A faster module is preferred when its delay advantage, together with switching and continuation effects, exceeds the quality advantage of a slower module. The framework identifies when dynamic orchestration is valuable and when simpler myopic or threshold policies are adequate.

Keywords: adaptive switching; heterogeneous decision modules; stochastic dynamic programming; latency–quality tradeoff; hysteresis
Mathematics Subject Classification: 90B70, 90C15, 68T05, 91B32

1. Introduction

Modern operational systems increasingly rely on portfolios of heterogeneous decision modules rather than on a single analytical engine [1–3]. These modules may include rule-based procedures, legacy optimization routines, statistical forecasting pipelines, machine-learning predictors, and advanced artificial intelligence (AI) inference architectures. Each module has a distinct operational profile: Some provide rapid but coarse responses, whereas others generate higher-quality decisions at the cost of greater latency, computational burden, interpretability limitations, or implementation complexity [4–6]. As operational environments become increasingly uncertain, time-sensitive, and data-intensive, the key managerial question is no longer which analytical model is best in isolation. Rather, the question is which decision module should be activated, retained, or replaced as operational conditions evolve over time [7].

This question arises in a wide range of time-sensitive operational systems [8–10]. Emergency response platforms must decide whether lightweight heuristic procedures or computationally intensive optimization routines should guide resource dispatch under severe time pressure. Intelligent transportation systems must balance fast reactive control against more sophisticated predictive coordination [11–13]. Cybersecurity operations may alternate among rule-based filtering, anomaly detection, and deep-learning modules depending on traffic volatility and infrastructure stress. Similar tradeoffs arise in healthcare operations, digital infrastructure management, industrial automation, and large-scale service systems, where multiple analytical engines coexist and must be orchestrated under changing workload, latency, and reliability conditions [14].

A central feature of these settings is that module performance is state dependent [15–17]. Under stable operating conditions, a computationally sophisticated AI or optimization module may generate superior decision quality. Under stressed regimes characterized by congestion, urgency, communication degradation, or high workload intensity, however, a lower-latency module may become operationally preferable despite lower analytical precision. Thus, the most accurate module need not be the most valuable module from an operational perspective. Static assignment policies are therefore limited because they ignore regime transitions, latency pressure, and the dynamic consequences of module deployment [13, 18, 19].

A second difficulty is that switching itself is costly [20–22]. Transitioning from one module to another may require recalibration, data synchronization, interface reconfiguration, supervisory approval, model warm-up, or temporary operational instability. Consequently, an organization should not switch modules whenever another module offers a marginally higher one-period reward [23–25]. Excessive reassignment may reduce reliability, increase instability, and undermine operational continuity. The resulting problem is a dynamic tradeoff among decision quality, response latency, switching cost, and stability. This tradeoff is especially important in mission-critical and time-sensitive environments, where both slow decision-making and excessive module oscillation can degrade system performance [26, 27].

Despite the practical importance of this problem, existing research does not provide a complete optimization framework for stability-aware orchestration of heterogeneous decision modules [28–30]. The decision support systems literature has examined AI-enabled decision augmentation, explainability, human–AI collaboration, and trust, but has paid less attention to the sequential control problem of when to switch among coexisting analytical engines [12, 18, 31]. The stochastic dynamic optimization literature provides powerful tools for sequential decision-making under uncertainty, yet typically treats the decision technology itself as fixed [19, 32, 33]. Research on operational flexibility and

switching decisions highlights the importance of transition costs and reconfiguration frictions, but the object of switching is usually a physical resource, production mode, service channel, or infrastructure configuration rather than an analytical decision module [20–22]. These streams leave open a basic operations question: How should a planner dynamically orchestrate heterogeneous analytical modules when latency, quality, and switching friction jointly determine long-run value [25, 26, 34]?

In this paper, we address this gap by developing an adaptive switching optimization framework for heterogeneous decision modules under operational uncertainty [35, 36]. We formulate module orchestration as a finite discounted Markov decision process (MDP) in which the currently active module is included in the system state and module selection is the control decision [2, 8, 37]. At each decision epoch, the planner observes the operational regime, latency-pressure state, and currently active module, and then decides whether to retain the current module or switch to another one [32, 33, 38]. The reward function jointly captures state-dependent analytical quality, response latency, switching cost, and instability penalties. This formulation converts heterogeneous AI deployment from a static model-selection problem into a dynamic operations optimization problem [7, 16].

We make four contributions. First, we introduce a stochastic dynamic optimization model for adaptive orchestration of heterogeneous decision modules. Unlike static model-selection and request-level routing approaches, the active analytical module is treated as an endogenous state component, and module deployment is optimized sequentially over time. Second, we develop a unified quality–latency–friction formulation with a general nondecreasing latency-disutility transformation. This formulation accommodates the linear baseline as well as convex and threshold-like delay penalties without changing the finite-state dynamic-programming foundation. Third, we establish structural properties of optimal switching policies, including stationary optimality, failure of universal module dominance, continuation and switching regions, pairwise hysteresis, switching-cost comparative statics, and latency-driven preference shifts. We further derive an exact Bellman decomposition of every adaptive–greedy disagreement, identify sufficient conditions under which zero switching friction makes the adaptive and greedy policies coincide, and provide value-loss bounds that characterize when greedy or threshold policies are nearly optimal. Fourth, we develop a regime-driven computational validation design that separates structural mechanism tests from statistical robustness checks. The design compares adaptive switching with interpretable benchmarks under multiple operating regimes, independent simulation replications, switching-cost perturbations, and alternative latency-disutility forms.

The main implication is that deployment of AI-enabled decision technologies should be viewed as a dynamic orchestration problem rather than as a one-time model-selection problem [7, 10, 16]. A high-quality analytical module may be valuable in stable regimes, but operational stress can make latency and switching stability more important than nominal accuracy. Conversely, a fast module may be useful during periods of urgency but inefficient if used permanently [12, 15, 39]. The proposed model provides a formal basis for understanding when adaptation is valuable, when continuation is preferable, and how switching friction creates stability-aware deployment policies [17].

The remainder is organized as follows. In Section 2, we review the related literature and position the present contribution. In Section 3, we develop the adaptive switching optimization framework. In Section 4, we establish structural properties of optimal switching policies. In Section 5, we present the solution method and policy characterization. In Section 6, we present the computational experiment design for validating the structural predictions of the model. In Section 7, we discuss managerial and operational implications. In Section 8, we conclude and outline future research directions.

2. Literature review and research positioning

In this section, we position the proposed adaptive switching framework relative to five streams of literature: decision support systems and AI-enabled operations, large language model (LLM) routing and adaptive inference, Markov decision processes and stochastic dynamic programming, switching and reconfiguration models, and latency–quality tradeoffs in time-sensitive operations. These streams provide important foundations for the present study, but none directly addresses the dynamic orchestration of heterogeneous analytical decision modules when latency, quality, switching friction, and operational instability jointly determine long-run value.

2.1. Decision support systems and AI-enabled operations

Decision support systems have evolved from stand-alone analytical tools into AI-enabled operational platforms that increasingly support prediction, recommendation, monitoring, and decision execution [28–30]. Recent research has examined intelligent decision support architectures, human–AI collaboration, explainability, trust, transparency, and the integration of machine-learning models into organizational workflows [7, 12, 37]. This literature establishes that AI-enabled systems can improve analytical quality, responsiveness, and decision support capability in complex operational environments [18, 19, 31].

However, much of this literature evaluates AI-enabled decision systems as individual analytical artifacts [15, 16, 39]. The dominant questions concern whether a given model is accurate, interpretable, trustworthy, usable, or adoptable. These are important questions, but they do not fully address the operational problem that arises when several heterogeneous analytical engines coexist within the same decision environment. In such settings, the decision-maker must determine not only whether a module performs well in isolation, but also when it should be activated, retained, suspended, or replaced as operating conditions change.

This distinction is important because AI deployment in operational systems is rarely a one-time model-selection decision [7, 16, 18]. A module that is highly accurate under stable conditions may be too slow under high workload or urgent response requirements, while a fast rule-based module may be valuable during stressed regimes but inefficient if used permanently. Therefore, the relevant operations problem is not merely AI adoption or model evaluation, but sequential module orchestration. In this paper, we address this gap by formulating AI-enabled decision support as a dynamic switching optimization problem [19].

2.2. LLM routing and adaptive inference

Recent LLM routing and adaptive-inference systems select among models with different capability, latency, and monetary cost profiles. Representative approaches such as FrugalGPT and RouteLLM demonstrate that request-level routing can reduce inference cost while preserving response quality. This literature is closely related to the present study because both settings recognize that no single analytical model is uniformly best across all requests or operating conditions.

The decision structure considered here is nevertheless different in three respects. First, request-level routing is typically formulated as a one-shot or myopic selection problem, whereas our planner optimizes a sequence of module choices. Second, the currently active module is included in the state, so a selection changes the configuration inherited by the next decision epoch. Third, switching may require

synchronization, warm-up, validation, or interface reconfiguration; these transition frictions generate continuation regions and hysteresis that are absent from cost–quality routing rules that evaluate each request independently. The proposed model therefore complements adaptive inference by providing an intertemporal orchestration layer for settings in which routing decisions are operationally coupled across time.

2.3. Markov decision processes and stochastic dynamic programming

The methodological foundation of this study is related to Markov decision processes, stochastic control, and dynamic programming [32–34]. These approaches provide a rigorous framework for sequential decision-making under uncertainty, where current decisions affect immediate rewards, future states, and long-run performance [35, 36, 40]. They have been widely applied to inventory control, admission control, scheduling, routing, service operations, maintenance, resource allocation, and other operational decision problems [41, 42].

The adaptive switching problem studied in this paper naturally belongs to this tradition because the planner repeatedly observes the operating regime and chooses an action that affects both current operational value and the future system configuration. However, the decision object differs from standard dynamic optimization models. In many existing models, the decision technology is treated as fixed: The planner optimizes inventory levels, service admissions, routing decisions, or resource allocations within a given decision model. For example, threshold-type control structures have been studied in queueing and service systems [34], and nonstationary online resource allocation models optimize allocation decisions under changing demand environments [35, 43]. In contrast to existing online resource-allocation frameworks [44], we treat the active analytical module itself as part of the system state and make module selection the control decision.

This modeling shift is essential for AI-enabled operations. Modern organizations do not only optimize decisions using a single analytical engine; they also manage portfolios of decision technologies with different latency, quality, reliability, and implementation profiles. The proposed formulation therefore extends the dynamic programming perspective from optimizing actions within a fixed model to optimizing the deployment of models themselves over time.

2.4. Switching, reconfiguration, and transition frictions

A related stream of operations research studies switching, reconfiguration, and transition decisions in production, service, computing, and infrastructure systems [20–22]. These models emphasize that flexibility can improve performance under uncertainty, but that switching is rarely costless. Transition may generate setup costs, downtime, migration delays, coordination losses, retraining requirements, temporary instability, or implementation risk [23, 25, 26]. As a result, optimal policies often balance the value of adaptation against the cost of transition [24, 45].

Recent work on adaptive scheduling, resilient restoration, and infrastructure resource planning illustrates the value of dynamic reconfiguration under uncertain operating conditions [13, 14, 36]. In such settings, reconfiguration is typically associated with physical resources, project schedules, infrastructure capacities, routing decisions, or service restoration actions. These studies show that adaptive reconfiguration can improve resilience and operational performance, but they do not focus on switching among analytical decision technologies [27, 46].

The present study shares this flexibility–friction logic but applies it to a different object: analytical decision modules [1–3]. Switching from one module to another may require recalibration, data synchronization, interface reconfiguration, supervisory approval, model warm-up, validation checks, or temporary disruption in operational routines. These transition frictions are not secondary implementation details; they directly affect the optimality of switching decisions. A planner should not switch simply because another module offers a slightly higher immediate reward [4, 6].

This observation motivates one of the central structural features of the model: pairwise switching hysteresis [10, 11, 17]. Positive transition friction can create continuation regions in which retaining the current module is optimal in a pairwise comparison even when an alternative module has a higher one-period value. This stability-aware behavior is especially important in mission-critical and time-sensitive operational systems, where excessive module oscillation can undermine reliability and operational continuity. We therefore extend switching and reconfiguration analysis to the orchestration of heterogeneous decision technologies [47].

2.5. Latency–quality tradeoffs in time-sensitive operations

Time-sensitive operational systems often face a fundamental tradeoff between response latency and decision quality [10, 13, 14]. Fast rule-based or heuristic modules may be preferable under urgent, congested, or degraded conditions, whereas slower optimization or AI-based modules may generate higher-quality decisions when sufficient time and computational resources are available [11, 38, 48]. Similar tradeoffs arise in emergency response, cybersecurity monitoring, intelligent transportation, healthcare operations, industrial automation, digital infrastructure management, and mission-oriented command-and-control systems [49–52].

Existing studies [53, 54] often treat latency, accuracy, and decision quality as separate design objectives [12, 15, 17]. For example, one line of work focuses on improving predictive accuracy, another on reducing response delay, and another on enhancing robustness, interpretability, or trustworthiness. In operational deployment, however, these attributes interact [39, 55, 56]. A module with high analytical quality may lose value when latency pressure is severe, while a fast module may become suboptimal when the environment is stable and decision quality dominates response speed [57].

In this paper, we explicitly integrate latency and quality within a stochastic switching framework. The reward function captures the operational value of analytical quality, the penalty of response latency, the cost of module transition, and the instability associated with switching. This integrated formulation enables analysis of regime-dependent module preference, latency-driven pairwise preference shifts, and stability-aware continuation policies.

2.6. Research positioning

Table 1 summarizes the positioning of the present contribution relative to the four streams reviewed above. The main distinction is that heterogeneous analytical modules are treated as operational assets whose deployment must be dynamically optimized under uncertainty, rather than as fixed tools evaluated independently.

Table 1. Research positioning relative to existing literature.

Research stream	Primary emphasis	Gap addressed in this paper
Decision support systems and AI-enabled operations	AI-assisted recommendation, explainability, trust, adoption, and human–AI interaction	Limited treatment of sequential orchestration among multiple heterogeneous analytical modules operating within the same decision environment
LLM routing and adaptive inference	Request-level selection among models with different quality, latency, and inference-cost profiles	Typically treats routing decisions independently and does not model the active module as a state variable, path-dependent switching friction, or long-run continuation value
Markov decision processes and stochastic dynamic programming	Sequential decision-making under uncertainty with state-dependent rewards and future consequences	Decision technology is usually fixed; in this paper, we treat the active analytical module as an endogenous state component and module selection as the control decision
Switching and reconfiguration models	Flexibility, transition costs, setup effects, reconfiguration, and operational frictions	Switching is typically applied to physical resources, production modes, or service channels rather than analytical decision modules
Latency–quality tradeoffs in time-sensitive operations	Balancing speed, accuracy, responsiveness, and robustness in operational systems	Limited integration of latency, analytical quality, switching cost, and instability within a single stochastic dynamic optimization model

Accordingly, we contribute to the operations research and decision support literature by developing a stochastic dynamic optimization framework for adaptive switching among heterogeneous decision modules [20, 32, 33]. The framework connects AI-enabled operational systems with dynamic programming, switching-cost analysis, and latency-sensitive decision optimization. This positioning clarifies that the contribution is not primarily about improving a single AI model; rather, we study how heterogeneous analytical modules should be dynamically governed when operational regimes evolve and switching itself is costly [12, 13, 28].

3. Adaptive switching optimization model

In this section, we develop an adaptive switching optimization model for coordinating heterogeneous decision modules under operational uncertainty [20, 32, 33]. The model treats the currently active decision module as part of the system state and module selection as the control decision. The formulation captures the operational tradeoff among analytical quality, response latency, switching friction, and system stability. A key modeling distinction is that the decision-maker does not merely optimize actions

within a fixed analytical model. Rather, the planner dynamically determines which analytical module should remain active or be replaced as operating conditions evolve [2].

The theoretical model is formulated as a finite-state, infinite-horizon discounted Markov decision process. The active module m_t is part of the state, whereas the selected module a_t is the control, with $m_{t+1} = a_t$. The general formulation permits controlled non-module-state transitions $P(y' \mid y, m, a)$, while the computational specialization may impose action-independent exogenous dynamics $P(y' \mid y)$. Rather than collecting all symbols in a separate notation table, we define each quantity when it first appears in the relevant subsection. The notation is introduced in four groups: operational state and decision variables, module-performance and reward components, controlled-transition and dynamic-programming quantities, and policy-comparison quantities. Computational notation used only for numerical evaluation, including the finite simulation horizon, Monte Carlo episodes, independent replications, and policy-comparison metrics, is introduced separately in Section 6.

Table 2 summarizes the notation used in the adaptive switching optimization framework. The currently active module m_t is included in the state, whereas the selected module a_t is the control decision. A switch occurs when $a_t \neq m_t$, and the selected module becomes active in the next period, so that $m_{t+1} = a_t$. The general model permits controlled non-module-state transitions $P(y' \mid y, m, a)$, whereas the computational specialization may assume action-independent exogenous dynamics, $P(y' \mid y, m, a) = P(y' \mid y)$. The theoretical formulation is infinite-horizon and discounted; the finite horizon T, simulation replications, and reported performance metrics are used only in the computational study.

Table 2. Notation used in the adaptive switching optimization framework.

Symbol	Type	Description
Panel A: Operational state and decision structure		
t	Index	Decision epoch in the dynamic model.
$\mathcal{S}$	Set	Finite operational state space.
s_t	State	Full operational state at decision epoch t.
r_t	State component	Operating regime, such as stable, volatile, stressed, or recovery.
d_t	State component	Workload, congestion, urgency, or latency-pressure intensity.
z_t	State component	Auxiliary operational variables, such as communication quality, demand intensity, queue length, disruption level, or system stress.
$y_t = (r_t, d_t, z_t)$	State component	Non-module operational state.
m_t	State component	Decision module active immediately before the decision at epoch t.
$s_t = (y_t, m_t)$	State	Full state representation separating operational conditions from the active module.
$\mathcal{R}$	Set	Finite set of operating regimes.
$\mathcal{Y}$	Set	Finite set of non-module operational states.
$\mathcal{M} = \{1, \ldots, M\}$	Set	Finite set of available decision modules.
M	Scalar	Number of available decision modules.
a_t	Control	Decision module selected at epoch t.
$m_{t+1} = a_t$	State update	Module-state transition induced by the selected action.
π	Policy	State-dependent module-selection policy.
$\pi^\star$	Policy	Optimal stationary adaptive switching policy.

Continued on next page

Symbol	Type	Description
Panel A: Operational state and decision structure		
π^G	Policy	Greedy policy that maximizes the one-period reward in each state.
$\widehat{\pi}$	Policy	Approximate, myopic, or threshold policy evaluated relative to $\pi^\star$.
Panel B: Module performance and reward components		
$Q_j(y)$	Function	Expected operational decision quality of module j under non-module state y.
$L_j(y)$	Function	Response latency of module j under non-module state y.
$U_j(y)$	Function	Operational instability, unreliability, or stress penalty associated with module j under state y.
$g(\ell)$	Function	Nonnegative and nondecreasing latency-disutility transformation applied to latency ℓ; $g(\ell) = \ell$ gives the linear baseline.
$\beta(d)$	Function	Nonnegative workload-dependent latency-disutility weight, assumed to be nondecreasing in workload or urgency d.
$\beta(d)g(L_j(y))$	Function	Transformed latency-disutility term associated with selecting module j under state y.
$C(i, j)$	Function	Cost of switching from currently active module i to selected module j.
c_{ij}	Scalar	Pairwise switching-cost coefficient, with $c_{ij} > 0$ for $i \neq j$ and $c_{ii} = 0$.
α	Parameter	Weight assigned to analytical decision quality.
γ	Parameter	Weight assigned to switching friction.
η	Parameter	Weight assigned to operational instability.
$R(y, m, a)$	Function	One-period reward incorporating quality, transformed latency disutility, switching friction, and instability.
$\Delta_{ij}(y)$	Function	Immediate reward differential between switching from module i to module j and retaining module i.
$D_{ij}(y)$	Function	Full dynamic switching gain from module i to module j, including immediate and continuation-value effects.
$B_{ij}(y)$	Function	Continuation-adjusted pairwise benefit of selecting module j instead of module i, before direct switching friction.
Panel C: Dynamic-programming and structural quantities		
$P(y' \mid y, m, a)$	Kernel	Transition probability for the next non-module state y', conditional on current state components y, m and selected module a.
$P(s' \mid s, a)$	Kernel	Full transition probability from state s to state s' under selected module a.
$\delta \in (0, 1)$	Parameter	Discount factor in the infinite-horizon objective.
$V^\pi(y, m)$	Function	Expected discounted value under policy π, starting from state (y, m).
$V^\star(y, m)$	Function	Optimal value function.
$Q^\star(s, a)$	Function	Optimal state–action value from selecting module a in state s and following an optimal policy thereafter.
$\mathcal{T}$	Operator	Bellman optimality operator.
$V_k(y, m)$	Function	Approximate value function at value-iteration step k.
$\varepsilon_{\mathrm{VI}}$	Scalar	Numerical tolerance used to terminate value iteration.
arg max	Operator	Maximizer used to construct a module-selection policy.
$a^A(s)$	Action	Module selected by the optimal full adaptive policy in state s.
$a^G(s)$	Action	Module selected by the greedy policy in state s.

Continued on next page

Symbol	Type	Description
Panel C: Dynamic-programming and structural quantities		
$\Delta_R(s)$	Function	Immediate-reward difference between the adaptive and greedy actions in state s.
$\Delta_F(s)$	Function	Discounted continuation-value difference between the adaptive and greedy actions in state s.
$\Delta_Q(s)$	Function	Bellman advantage of the adaptive action over the greedy action, satisfying $\Delta_Q(s) = \Delta_R(s) + \Delta_F(s)$.
$\omega(s)$	Function	Spread of discounted continuation values across feasible actions in state s.
ϵ_{pol}	Scalar	Uniform statewise Bellman loss used to bound the value loss of an approximate, greedy, or threshold policy.
C_i	Set	Continuation region in which retaining currently active module i is optimal.
$\mathcal{W}_{ij}$	Set	Switching region in which moving from module i to module j is optimal.
$\mathcal{A}_{ij}(\gamma)$	Set	Pairwise set of non-module states in which switching from i to j is attractive at switching-friction weight γ.
Panel D: Computational evaluation quantities		
T	Scalar	Finite simulation horizon used in the computational experiments.
N_{MC}	Scalar	Number of Monte Carlo episodes or trajectories within a replication.
N_{rep}	Scalar	Number of independent simulation replications or random seeds.
Reward	Metric	Average or cumulative operational reward over the evaluation horizon.
Latency	Metric	Average raw response latency.
LatencyPenalty	Metric	Average transformed latency-disutility term.
Quality	Metric	Average operational decision quality.
SwitchFreq	Metric	Frequency of module switching over the evaluation horizon.
SwitchCost	Metric	Average or cumulative switching cost.
Instability	Metric	Average or cumulative operational instability penalty.
PolicyAgreement	Metric	Proportion of states in which the adaptive and greedy policies select the same module.
DisagreementStates	Metric	Number of states in which the adaptive and greedy policies select different modules.
RelativeImprovement	Metric	Percentage improvement in mean reward of the adaptive policy relative to the greedy policy.
CRN	Design principle	Common random numbers used to compare policies under identical exogenous sample paths.

3.1. Operational state and decision structure

Consider a discrete-time operational system observed at decision epochs $t = 0, 1, 2, \ldots$. At each epoch, the organization operates under uncertain conditions such as workload fluctuations, congestion escalation, communication degradation, urgency shocks, and operational disturbances. The planner must decide whether to retain the currently active analytical module or switch to another available decision module.

The non-module operational state is denoted by

$$y_t = (r_t, d_t, z_t) \in \mathcal{Y}, \tag{3.1}$$

where r_t denotes the operating regime, d_t denotes workload or urgency intensity, and z_t collects auxiliary operational variables such as congestion, queue length, communication quality, demand intensity, or disruption signals. The full operational state is

$$s_t = (y_t, m_t) \in \mathcal{S}, \tag{3.2}$$

where m_t denotes the currently active decision module. Equivalently, $s_t = (r_t, d_t, z_t, m_t)$. The state space $\mathcal{S}$ is finite.

Let

$$\mathcal{M} = \{1, \dots, M\}$$

denote the finite set of available decision modules. These modules may include lightweight heuristic engines, rule-based systems, legacy optimization routines, machine-learning pipelines, and advanced AI inference architectures [7, 12, 16]. At each decision epoch, the planner selects

$$a_t \in \mathcal{M}. \tag{3.3}$$

If $a_t = m_t$, the current module is retained. If $a_t \neq m_t$, a switching action occurs. The selected module becomes the active module in the next period:

$$m_{t+1} = a_t. \tag{3.4}$$

Thus, module selection affects both immediate operational performance and future system configuration. This modeling choice distinguishes the present model from static model-selection approaches.

3.2. Module performance and controlled transition dynamics

Each module has state-dependent operational characteristics. For module $j \in \mathcal{M}$ and non-module operational state $y = (r, d, z)$, let

$$Q_j(y), \qquad L_j(y), \qquad U_j(y)$$

denote, respectively, its expected decision quality, response latency, and operational instability or reliability penalty. These quantities depend on the operating regime, workload intensity, urgency, communication quality, and other operational variables. They do not depend on the identity of the currently active module except through switching costs and future state transitions.

The non-module components of the operational environment evolve according to a controlled Markov transition law [32, 33]. Conditional on current non-module state y_t, currently active module m_t, and selected module a_t, let

$$P(y_{t+1} \mid y_t, m_t, a_t) \tag{3.5}$$

denote the probability of the next non-module operational state. Since the selected module becomes active in the next period, the full transition probability satisfies

$$P(s_{t+1} \mid s_t, a_t) = P(y_{t+1} \mid y_t, m_t, a_t)\mathbf{1}\{m_{t+1} = a_t\}, \tag{3.6}$$

where $s_t = (y_t, m_t)$ and $s_{t+1} = (y_{t+1}, m_{t+1})$. This representation makes explicit that the action updates the module component of the state, while the operational environment evolves stochastically. It also allows

the selected module to influence future conditions, for example by reducing congestion, increasing reliability, or accelerating recovery.

For theoretical generality, the kernel in (3.6) permits the selected module to affect the evolution of the non-module state. The computational specialization used later is deliberately narrower:

$$P(y' \mid y, m, a) = P(y' \mid y), \qquad m' = a. \tag{3.7}$$

Thus, actions affect future value through the inherited active-module configuration, while regime, workload, and stress evolve exogenously. We state results that depend on this specialization explicitly rather than presenting them as properties of the general controlled-transition model.

3.3. Reward function and switching frictions

The one-period operational reward from selecting module a_t in state $s_t = (y_t, m_t)$ is defined as a quality–latency–friction tradeoff [13, 14, 58]:

$$R_g(s_t, a_t) = R_g(y_t, m_t, a_t) = \alpha Q_{a_t}(y_t) - \beta(d_t) g(L_{a_t}(y_t)) - \gamma C(m_t, a_t) - \eta U_{a_t}(y_t), \tag{3.8}$$

where $\alpha, \gamma, \eta \geq 0, \beta(d_t) \geq 0$ is the workload-dependent latency-disutility weight, and $g : \mathbb{R}_+ \to \mathbb{R}_+$ is a nondecreasing latency-disutility transformation with $g(0) = 0$. The baseline model uses $g(L) = L$. Convex power, normalized exponential, and piecewise deadline penalties are admissible alternatives. The function $\beta(\cdot)$ is assumed to be nondecreasing in d_t, reflecting the fact that the same response delay becomes more costly as workload, congestion, urgency, or mission pressure increases. This distinction between the weight $\beta(d)$ and the transform $g(L)$ avoids conflating state-dependent latency pressure with the functional shape of delay disutility. Throughout the remainder of the theoretical development, we write $R(s, a)$ as shorthand for $R_g(s, a)$ when the transformation g is fixed.

The switching cost is specified as

$$C(i, j) = \begin{cases} 0, & i = j, \\ c_{ij}, & i \neq j, \end{cases} \qquad c_{ij} > 0 \text{ [20–22]}. \tag{3.9}$$

The term $C(i, j)$ captures transition frictions such as recalibration delay, data synchronization, interface reconfiguration, supervisory approval, model warm-up, validation checks, or temporary disruption [23, 24, 26]. Positive switching friction is essential because it prevents excessive oscillation among modules and creates stability regions in which continuation remains optimal [11].

For later analysis, consider a state $s = (y, i)$ in which module i is currently active. The immediate pairwise switching differential from retaining module i to selecting module j is

$$\Delta_{ij}(y) = R(y, i, j) - R(y, i, i). \tag{3.10}$$

For $i \neq j$, this differential can be expanded as

$$\Delta_{ij}(y) = \alpha\{Q_j(y) - Q_i(y)\} - \beta(d)\{g(L_j(y)) - g(L_i(y))\} - \eta\{U_j(y) - U_i(y)\} - \gamma c_{ij}. \tag{3.11}$$

Thus, switching is immediately attractive only when the quality, latency, or stability advantage of module j is large enough to overcome switching friction. However, the optimal switching decision is

not determined by immediate rewards alone. The selected module also determines the future active module and may change future operating conditions.

To capture this dynamic effect, define the full dynamic switching gain from module i to module j at state $s = (y, i)$ as

$$D_{ij}(y) = \left[R(y,i,j) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,i,j) V^\star(y',j)\right] - \left[R(y,i,i) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,i,i) V^\star(y',i)\right]. \tag{3.12}$$

Here $V^\star(y', j)$ denotes the optimal continuation value when the next non-module state is y' and the active module is j. Thus, the continuation value explicitly reflects the fact that the selected module becomes active in the next decision epoch.

Switching from i to j is optimal only if $D_{ij}(y)$ is sufficiently large relative to all other feasible module choices. This distinction between $\Delta_{ij}(y)$ and $D_{ij}(y)$ is important: A module may look attractive myopically but still be dynamically suboptimal because it increases future instability, induces additional transitions, or leads to unfavorable operating regimes.

3.4. Dynamic optimization problem

A stationary policy $\pi : \mathcal{S} \to \mathcal{M}$ maps each operational state to a selected decision module. Equivalently, since $s = (y, m)$, a policy can be written as $\pi(y, m)$. For an initial state (y, m), the expected discounted value of policy π is

$$V^\pi(y,m) = \mathbb{E}^\pi_{(y,m)}\left[\sum_{t=0}^{\infty} \delta^t R(y_t, m_t, a_t)\right], \qquad 0 < \delta < 1. \tag{3.13}$$

The optimal value function is

$$V^\star(y,m) = \sup_\pi V^\pi(y,m). \tag{3.14}$$

The adaptive switching problem can therefore be written as

$$\max_\pi \ \mathbb{E}^\pi_{(y,m)}\left[\sum_{t=0}^{\infty} \delta^t \left\{\alpha Q_{a_t}(y_t) - \beta(d_t) g(L_{a_t}(y_t)) - \gamma C(m_t, a_t) - \eta U_{a_t}(y_t)\right\}\right], \tag{3.15}$$

subject to the controlled transition law in (3.6). This formulation emphasizes that the planner optimizes long-run operational value rather than one-period module performance.

The optimal value function satisfies the Bellman optimality equation [32, 33]

$$V^\star(y,m) = \max_{a \in \mathcal{M}} \left\{R(y,m,a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,m,a) V^\star(y',a)\right\}. \tag{3.16}$$

The corresponding optimal policy is

$$\pi^\star(y,m) \in \arg\max_{a \in \mathcal{M}} \left\{R(y,m,a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,m,a) V^\star(y',a)\right\}. \tag{3.17}$$

Define the Bellman operator by

$$(\mathcal{T}V)(y,m) = \max_{a \in \mathcal{M}} \left\{R(y,m,a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,m,a) V(y',a)\right\}. \tag{3.18}$$

3.5. Standing assumptions and foundational properties

We now state the standing assumptions used throughout the structural analysis. These assumptions are standard for finite discounted dynamic programming [32, 33], but are stated explicitly to clarify the mathematical basis of the adaptive switching model.

Assumption 3.1 (Finite discounted switching model)**.** The non-module state space $\mathcal{Y}$ and module set $\mathcal{M}$ are finite. The full state is $s = (y, m) \in \mathcal{Y} \times \mathcal{M}$. The transition law $P(y' \mid y, m, a)$ is Markovian, the reward $R(y, m, a)$ is bounded, and the discount factor satisfies $0 < \delta < 1$.

Assumption 3.2 (Positive transition friction)**.** The switching-cost function satisfies

$$C(i, i) = 0, \qquad C(i, j) = c_{ij} > 0 \quad \text{for all } i \neq j.$$

The coefficients c_{ij} need not be symmetric.

Assumption 3.3 (Monotone latency pressure)**.** The latency-disutility weight $\beta(d)$ is nonnegative and nondecreasing in the workload or urgency intensity d. That is, for any $d_1 \leq d_2$,

$$0 \leq \beta(d_1) \leq \beta(d_2).$$

Assumption 3.4 (Nondegenerate module heterogeneity)**.** There exist two modules $i, j \in \mathcal{M}$ and at least one non-module state $y \in \mathcal{Y}$ such that

$$Q_i(y) \neq Q_j(y) \quad \text{or} \quad L_i(y) \neq L_j(y) \quad \text{or} \quad U_i(y) \neq U_j(y).$$

Assumption 3.1 ensures that the dynamic programming problem is well-defined. Assumption 3.2 formalizes the fact that switching between heterogeneous decision modules is not costless. Assumption 3.3 captures the operational reality that response delay becomes more costly as workload, urgency, or congestion increases. Assumption 3.4 excludes the degenerate case in which all modules are operationally equivalent.

The following lemma establishes boundedness of the value functions.

Lemma 3.5 (Bounded value functions)**.** *Under Assumption 3.1, for any stationary policy π,*

$$|V^{\pi}(y, m)| \leq \frac{\bar{R}}{1 - \delta}, \qquad \forall (y, m) \in \mathcal{Y} \times \mathcal{M},$$

where

$$\bar{R} = \max_{y \in \mathcal{Y}, m \in \mathcal{M}, a \in \mathcal{M}} |R(y, m, a)|.$$

Consequently, the optimal value function $V^{\star}$ is bounded.

Proof. For any policy π and any initial state (y, m),

$$|V^{\pi}(y, m)| = \left| \mathbb{E}^{\pi}_{(y,m)} \left[\sum_{t=0}^{\infty} \delta^t R(y_t, m_t, a_t) \right] \right| \leq \mathbb{E}^{\pi}_{(y,m)} \left[\sum_{t=0}^{\infty} \delta^t |R(y_t, m_t, a_t)| \right].$$

Since $|R(y, m, a)| \leq \bar{R}$ for all y, m, a, it follows that

$$|V^{\pi}(y, m)| \leq \sum_{t=0}^{\infty} \delta^t \bar{R} = \frac{\bar{R}}{1 - \delta}.$$

Taking the supremum over policies preserves the boundedness of $V^{\star}$. □

The next lemma gives the contraction property of the Bellman operator.

Lemma 3.6 (Bellman contraction)**.** *Under Assumption 3.1, the Bellman operator $\mathcal{T}$ defined in* (3.18) *is a contraction under the sup norm. Specifically, for any two bounded value functions V and W,*

$$\|\mathcal{T}V - \mathcal{T}W\|_\infty \le \delta \|V - W\|_\infty.$$

Proof. For any state $(y, m) \in \mathcal{Y} \times \mathcal{M}$, define

$$H_a(y, m; V) = R(y, m, a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y, m, a) V(y', a).$$

Then

$$(\mathcal{T}V)(y, m) = \max_{a \in \mathcal{M}} H_a(y, m; V).$$

For any V, W, the difference between two maxima is bounded by the maximum difference between the corresponding terms:

$$|(\mathcal{T}V)(y, m) - (\mathcal{T}W)(y, m)| \le \max_{a \in \mathcal{M}} |H_a(y, m; V) - H_a(y, m; W)|.$$

Therefore,

$$|(\mathcal{T}V)(y, m) - (\mathcal{T}W)(y, m)| \le \delta \max_{a \in \mathcal{M}} \left| \sum_{y' \in \mathcal{Y}} P(y' \mid y, m, a) \{V(y', a) - W(y', a)\} \right|.$$

Because $P(\cdot \mid y, m, a)$ is a probability distribution,

$$|(\mathcal{T}V)(y, m) - (\mathcal{T}W)(y, m)| \le \delta \|V - W\|_\infty.$$

Taking the supremum over $(y, m) \in \mathcal{Y} \times \mathcal{M}$ proves the result. □

The following theorem establishes the existence of an optimal stationary switching policy.

Theorem 3.7 (Existence of a stationary optimal switching policy)**.** *Under Assumption 3.1, there exists a unique bounded optimal value function $V^\star$ satisfying the Bellman equation* (3.16)*. Moreover, there exists a deterministic stationary policy $\pi^\star : \mathcal{Y} \times \mathcal{M} \to \mathcal{M}$ such that*

$$V^{\pi^\star}(y, m) = V^\star(y, m), \qquad \forall (y, m) \in \mathcal{Y} \times \mathcal{M}.$$

Proof. By Lemma 3.6, the Bellman operator $\mathcal{T}$ is a contraction on the complete metric space of bounded real-valued functions on the finite state space $\mathcal{Y} \times \mathcal{M}$, equipped with the sup norm. Hence, by the Banach fixed-point theorem, $\mathcal{T}$ has a unique fixed point $V^\star$, and value iteration converges to this fixed point from any bounded initialization.

Since the action set $\mathcal{M}$ is finite, the maximum in the Bellman equation is attained for every state (y, m). Selecting any maximizer defines a deterministic stationary policy $\pi^\star$. The fixed-point equation then implies that this policy attains $V^\star$ in every state. □

We next characterize optimal switching using the dynamic switching gain defined in (3.12).

Proposition 3.8 (Dynamic switching characterization)**.** *Consider a state* (y, i) *in which module* i *is currently active. Retaining module* i *is optimal at state* (y, i) *if and only if*

$$D_{ij}(y) \le 0 \qquad \text{for all } j \in \mathcal{M}.$$

Switching from module i *to module* $j \neq i$ *is optimal only if*

$$D_{ij}(y) = \max_{k \in \mathcal{M}} D_{ik}(y) \ge 0.$$

Proof. For state (y, i), define the optimal dynamic value of selecting module a as

$$G_a(y, i) = R(y, i, a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y, i, a) V^\star(y', a).$$

By the Bellman equation, an action is optimal if it maximizes $G_a(y, i)$ over $a \in \mathcal{M}$. Since $D_{ij}(y) = G_j(y, i) - G_i(y, i)$, retaining module i is optimal if and only if $G_i(y, i) \ge G_j(y, i)$ for every j, which is equivalent to $D_{ij}(y) \le 0$ for every j. Similarly, switching from i to j is optimal only if $G_j(y, i) \ge G_k(y, i)$ for all k, which is equivalent to $D_{ij}(y) = \max_{k \in \mathcal{M}} D_{ik}(y)$. If $j \neq i$, optimal switching additionally requires $G_j(y, i) \ge G_i(y, i)$ or $D_{ij}(y) \ge 0$. □

The distinction between the immediate differential $\Delta_{ij}(y)$ and the dynamic gain $D_{ij}(y)$ yields the following implication.

Corollary 3.9 (Myopic switching may be dynamically suboptimal)**.** *Consider a state* (y, i) *and a module* $j \neq i$*. If*

$$\Delta_{ij}(y) > 0$$

but the continuation-value loss from selecting j *rather than retaining* i *is sufficiently negative, then*

$$D_{ij}(y) < 0.$$

In such a state, switching from i *to* j *improves the one-period reward but is not dynamically optimal.*

Proof. Using the definition of $D_{ij}(y)$,

$$D_{ij}(y) = \Delta_{ij}(y) + \delta \left[\sum_{y' \in \mathcal{Y}} P(y' \mid y, i, j) V^\star(y', j) - \sum_{y' \in \mathcal{Y}} P(y' \mid y, i, i) V^\star(y', i) \right].$$

The first term is the immediate reward improvement from switching. The second term is the continuation-value difference. If this continuation-value difference is smaller than $-\Delta_{ij}(y)/\delta$, then $D_{ij}(y) < 0$. Thus a myopically attractive switch may be dynamically suboptimal. □

The formulation shows that the analytically strongest module is not necessarily the operationally optimal module in every state. Under stable conditions, high-quality but slower modules may be preferred, whereas under stressed conditions, lower-latency modules may dominate as delay penalties increase. Positive switching friction further prevents myopic oscillation and induces stability-aware continuation behavior. In Section 4, we analyze these structural implications formally.

4. Structural properties of optimal switching

In this section, we derive structural properties of the optimal adaptive switching policy [20, 32, 33]. The results characterize when continuation is optimal, when switching is optimal, why universal module dominance is generally too strong to expect, how positive transition friction generates pairwise hysteresis, and why latency pressure can reverse module preferences across operating regimes [24].

The analysis builds on the dynamic switching gain $D_{ij}(y)$ defined in (3.12) [32, 33]. This quantity compares the full Bellman value of switching from module i to module j against retaining module i, including both immediate reward and continuation effects. The continuation values explicitly reflect that the selected module becomes the active module in the next decision epoch.

4.1. Continuation and switching regions

For each currently active module $i \in \mathcal{M}$, define the continuation region

$$C_i = \left\{(y, i) \in \mathcal{Y} \times \mathcal{M} : D_{ij}(y) \leq 0 \ \text{ for all } j \in \mathcal{M}\right\}. \tag{4.1}$$

When $(y, i) \in C_i$, retaining module i is optimal. For $i \neq j$, define the switching region from module i to module j as

$$\mathcal{W}_{ij} = \left\{(y, i) \in \mathcal{Y} \times \mathcal{M} : D_{ij}(y) = \max_{k \in \mathcal{M}} D_{ik}(y) \geq 0\right\}. \tag{4.2}$$

When $(y, i) \in \mathcal{W}_{ij}$, switching from i to j is an optimal action. If multiple modules attain the same maximal Bellman action value, the stability-preserving tie-breaking rule introduced in Section 5.5 is applied.

Proposition 4.1 (Bellman partition of continuation and switching regions)**.** *For each active module $i \in \mathcal{M}$, the state subset*

$$\mathcal{X}_i = \{(y, i) : y \in \mathcal{Y}\}$$

is covered by the continuation region C_i and the switching regions $\{\mathcal{W}_{ij} : j \in \mathcal{M},\ j \neq i\}$. The cover is a partition whenever the Bellman maximizer is unique in every state; otherwise, regions may overlap only at states with multiple optimal actions.

Proof. Fix a state (y, i). Define the Bellman action value of selecting module a by

$$G_a(y, i) = R(y, i, a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y, i, a) V^\star(y', a).$$

Since $\mathcal{M}$ is finite, the maximum $\max_{a \in \mathcal{M}} G_a(y, i)$ is attained.

If $G_i(y, i) \geq G_j(y, i)$ for all $j \in \mathcal{M}$, then

$$D_{ij}(y) = G_j(y, i) - G_i(y, i) \leq 0 \quad \text{for all } j \in \mathcal{M},$$

so $(y, i) \in C_i$. If retaining i is not optimal, then there exists some $j \neq i$ such that

$$G_j(y, i) = \max_{k \in \mathcal{M}} G_k(y, i) \quad \text{and} \quad G_j(y, i) \geq G_i(y, i).$$

Equivalently,

$$D_{ij}(y) = G_j(y,i) - G_i(y,i) = \max_{k\in\mathcal{M}}\{G_k(y,i) - G_i(y,i)\} = \max_{k\in\mathcal{M}} D_{ik}(y) \geq 0,$$

so $(y,i) \in \mathcal{W}_{ij}$. Therefore, every state with active module i belongs to at least one continuation or switching region. If the maximizer is unique, exactly one region applies. If multiple actions maximize the Bellman expression, the corresponding regions overlap only at such tie states. □

Proposition 4.1 shows that optimal switching is a state-space classification problem induced by the Bellman action values. The classification is dynamic rather than myopic because it depends on continuation values.

4.2. Universal dominance and state-dependent preference

A static model-selection approach is valid only under a strong dominance condition [7, 12, 16]. For any candidate selected module a, define the Bellman action value

$$G_a(y,m) = R(y,m,a) + \delta \sum_{y'\in\mathcal{Y}} P(y' \mid y,m,a)V^\star(y',a). \tag{4.3}$$

For a module $j \in \mathcal{M}$, define dynamic universal dominance as

$$G_j(y,m) \geq G_k(y,m), \qquad \forall (y,m) \in \mathcal{Y}\times\mathcal{M},\ \forall k \in \mathcal{M}. \tag{4.4}$$

If (4.4) holds, module j is an optimal action in every state. If it fails, then no single-module deployment rule can be justified solely by dynamic dominance.

Proposition 4.2 (Necessary and sufficient condition for static dominance)**.** *A module $j \in \mathcal{M}$ is optimal in every state if and only if*

$$G_j(y,m) \geq G_k(y,m), \qquad \forall (y,m) \in \mathcal{Y}\times\mathcal{M},\ \forall k \in \mathcal{M}.$$

If the inequality is strict for all $k \neq j$ and all (y,m), then j is the unique optimal module in every state.

Proof. The Bellman equation selects an action from

$$\arg\max_{a\in\mathcal{M}} G_a(y,m).$$

Thus, module j is optimal in every state if and only if $j \in \arg\max_{a\in\mathcal{M}} G_a(y,m)$ for every (y,m), which is equivalent to $G_j(y,m) \geq G_k(y,m)$ for every (y,m) and every $k \in \mathcal{M}$. If the inequality is strict for all $k \neq j$, then no other module attains the same Bellman action value in any state, so j is the unique optimal module in every state. □

Corollary 4.3 (Pairwise preference reversals rule out pairwise dominance)**.** *Suppose there exist two states $(y_1,m_1),(y_2,m_2) \in \mathcal{Y}\times\mathcal{M}$ and two modules $i, j \in \mathcal{M}$ such that*

$$G_i(y_1,m_1) > G_j(y_1,m_1), \qquad G_j(y_2,m_2) > G_i(y_2,m_2).$$

Then neither i nor j dynamically dominates the other over the state space. In particular, no static rule that always chooses i, or always chooses j, can be uniformly optimal over all states.

Proof. The first inequality shows that j cannot dominate i in all states. The second inequality shows that i cannot dominate j in all states. Therefore neither module is pairwise dominant. A static rule that always selects i is suboptimal at (y_2, m_2), while a static rule that always selects j is suboptimal at (y_1, m_1). □

This result is deliberately stated as a pairwise dominance statement. A third module could still dominate both i and j unless additional inequalities rule it out. Thus, universal dominance should be established by Proposition 4.2, while observed pairwise reversals demonstrate why static comparison among candidate modules is generally insufficient.

4.3. Pairwise switching hysteresis

Positive transition friction can generate hysteresis: Retaining the current module may be optimal even when another module has become attractive under a myopic comparison [20–22]. To state this result rigorously, we impose a local pairwise separability condition for the two modules being compared [23, 26].

Assumption 4.4 (Pairwise active-module separability)**.** For a pair of modules i, j, the module-performance functions $Q_a(y)$, $L_a(y)$, and $U_a(y)$, $a \in \{i, j\}$, depend on the non-module state y and the selected module a, but not on the previously active module except through the direct switching cost. In addition, the non-module transition law under selected module $a \in \{i, j\}$ is independent of whether the previously active module is i or j. That is,

$$P(y' \mid y, i, a) = P(y' \mid y, j, a), \qquad a \in \{i, j\}.$$

Under this assumption, write $P_a(y' \mid y)$ for the common transition law when module a is selected.

Under Assumption 4.4, define the continuation-adjusted pairwise benefit of selecting module j instead of module i, before subtracting direct switching friction, by

$$\begin{aligned} B_{ij}(y) = {} & \alpha\{Q_j(y) - Q_i(y)\} - \beta(d)\{g(L_j(y)) - g(L_i(y))\} \\ & - \eta\{U_j(y) - U_i(y)\} + \delta \sum_{y' \in \mathcal{Y}} \left[P_j(y' \mid y) V^\star(y', j) - P_i(y' \mid y) V^\star(y', i) \right]. \end{aligned} \tag{4.5}$$

Thus, $B_{ij}(y)$ is the dynamic advantage of selecting j instead of i before subtracting the direct switching friction.

For the state (y, i), the pairwise dynamic gain from switching to j is

$$D_{ij}(y) = B_{ij}(y) - \gamma c_{ij}. \tag{4.6}$$

For the state (y, j), the pairwise dynamic gain from switching back to i is

$$D_{ji}(y) = -B_{ij}(y) - \gamma c_{ji}. \tag{4.7}$$

Theorem 4.5 (Pairwise switching hysteresis)**.** *Under Assumption 4.4, suppose $c_{ij} > 0$, $c_{ji} > 0$, and $\gamma > 0$. Then, the interval*

$$-\gamma c_{ji} < B_{ij}(y) < \gamma c_{ij} \tag{4.8}$$

is a pairwise hysteresis band for the module pair (i, j). Within this band, switching from i to j is not pairwise attractive when i is active, and switching from j to i is not pairwise attractive when j is active. Thus, the locally optimal pairwise decision depends on the currently active module.

Proof. When module i is active, switching to j is pairwise attractive only if $D_{ij}(y) = B_{ij}(y) - \gamma c_{ij} \geq 0$. Hence, retaining i is pairwise optimal relative to j whenever $B_{ij}(y) < \gamma c_{ij}$.

Similarly, when module j is active, switching back to i is attractive only if $D_{ji}(y) = -B_{ij}(y) - \gamma c_{ji} \geq 0$, or equivalently, $B_{ij}(y) \leq -\gamma c_{ji}$. Thus, retaining j is pairwise optimal relative to i whenever $B_{ij}(y) > -\gamma c_{ji}$.

Combining these conditions yields

$$-\gamma c_{ji} < B_{ij}(y) < \gamma c_{ij}.$$

Within this interval, neither switching direction generates sufficient dynamic benefit to offset its associated switching cost. Therefore, continuation is pairwise optimal in both active-module states, establishing pairwise switching hysteresis. □

Corollary 4.6 (Width of the hysteresis band)**.** *Under the conditions of Theorem 4.5, the width of the hysteresis band is*

$$\gamma(c_{ij} + c_{ji}).$$

It is increasing in γ and in each pairwise transition cost.

Proof. The lower boundary of the band is $-\gamma c_{ji}$, and the upper boundary is γc_{ij}. The width is therefore

$$\gamma c_{ij} - (-\gamma c_{ji}) = \gamma(c_{ij} + c_{ji}).$$

Since $c_{ij} > 0$, $c_{ji} > 0$, and $\gamma > 0$, the width is increasing in each of these quantities. □

Theorem 4.5 identifies a local pairwise stability mechanism [11, 20, 24]. It does not claim global monotonicity of the full optimal switching regions over the entire state space without additional monotone dynamic programming assumptions. Instead, it establishes the exact pairwise mechanism by which positive transition friction creates continuation bands. This is the structural source of stability-aware switching.

4.4. Pairwise switching-cost comparative statics

The previous result implies a clean pairwise comparative static [20, 22, 23]. For a fixed continuation-adjusted benefit $B_{ij}(y)$, define the pairwise switching set at friction level γ by

$$\mathcal{A}_{ij}(\gamma) = \{y : B_{ij}(y) \geq \gamma c_{ij}\}. \tag{4.9}$$

Proposition 4.7 (Pairwise monotonicity in switching friction)**.** *For any pair $i \neq j$, if $0 \leq \gamma_1 \leq \gamma_2$, then*

$$\mathcal{A}_{ij}(\gamma_2) \subseteq \mathcal{A}_{ij}(\gamma_1). \tag{4.10}$$

Thus, holding the continuation-adjusted benefit fixed, increasing switching friction weakly reduces the set of non-module states in which switching from i to j is pairwise attractive.

Proof. Let $y \in \mathcal{A}_{ij}(\gamma_2)$. By definition, $B_{ij}(y) \geq \gamma_2 c_{ij}$. Since $c_{ij} > 0$ and $\gamma_2 \geq \gamma_1$, it follows that $\gamma_2 c_{ij} \geq \gamma_1 c_{ij}$, and hence $B_{ij}(y) \geq \gamma_1 c_{ij}$. Therefore, $y \in \mathcal{A}_{ij}(\gamma_1)$, which proves $\mathcal{A}_{ij}(\gamma_2) \subseteq \mathcal{A}_{ij}(\gamma_1)$. □

This proposition is intentionally stated at the pairwise comparison level. A stronger global monotonicity result for the full optimal policy would require additional lattice or monotone dynamic programming structure, because changing γ changes the fixed point $V^\star$. The pairwise result is sufficient for the present paper: It explains why switching friction creates wider continuation bands and motivates sensitivity analysis with respect to γ.

4.5. Regime-sensitive latency preference

Let $u = (r, z)$ denote the non-latency operational components, so that the non-module state can be written as $y = (u, d)$. The reward function includes a workload-dependent latency-disutility weight $\beta(d)$ [13, 14, 58]. Under Assumption 3.3, the weight assigned to delay disutility increases as workload, congestion, or urgency becomes more severe. This can reverse the preference between a high-quality but slow module and a lower-quality but faster module.

Consider two modules f and h, where f is faster and h has higher analytical quality at the relevant operational state:

$$L_h(u, d) > L_f(u, d), \qquad Q_h(u, d) > Q_f(u, d). \tag{4.11}$$

Here f denotes the faster module and h denotes the higher-quality module.

The following assumption states the single-crossing condition needed to obtain a clean threshold representation.

Assumption 4.8 (Latency single-crossing)**.** Fix a non-latency operational state $u = (r, z)$, a currently active module m, and two modules f, h. The continuation-adjusted pairwise preference difference

$$\begin{aligned}\Phi_{fh}(d; u, m) = {} & R((u, d), m, h) - R((u, d), m, f) \\ & + \delta \sum_{y' \in \mathcal{Y}} \left[P(y' \mid (u, d), m, h)V^\star(y', h) - P(y' \mid (u, d), m, f)V^\star(y', f)\right]\end{aligned} \tag{4.12}$$

is nonincreasing in d.

Proposition 4.9 (Latency-driven pairwise preference shift)**.** *Under Assumption 4.8, the pairwise incentive to choose the slower high-quality module h over the faster module f weakly decreases as latency pressure d increases. If the preference difference crosses zero, then the pairwise choice admits a latency-threshold representation.*

Proof. By Assumption 4.8, $\Phi_{fh}(d; u, m)$ is nonincreasing in d. This quantity is the continuation-adjusted pairwise advantage of selecting h rather than f, before any additional tie-breaking convention. Hence, the incentive to choose the slower high-quality module weakly decreases as latency pressure increases.

If there exist $d_1 < d_2$ such that $\Phi_{fh}(d_1; u, m) \geq 0$ and $\Phi_{fh}(d_2; u, m) \leq 0$, monotonicity implies that the set of latency-pressure levels for which h is pairwise preferred to f is a lower set. Therefore, there exists a threshold $d^\star$ such that h is pairwise preferred below the threshold and f is pairwise preferred above it, up to tie states. □

This result formalizes a core operational principle: The most accurate module need not be optimal in stressed regimes [13, 38, 58]. When latency pressure is high, the marginal value of speed can dominate the marginal gain in analytical quality [10].

4.6. Why greedy switching can fail

A natural benchmark is a greedy switching rule that selects the module with the highest one-period reward [32, 33]. The following result shows why such a rule can be dynamically suboptimal.

Proposition 4.10 (Existence of greedy switching failure)**.** *There exists a finite discounted adaptive switching instance in which a module switch has a strictly positive immediate reward differential but a strictly negative dynamic switching gain. Consequently, greedy switching is not generally optimal.*

Proof. Consider two non-module states A and B, two modules 1 and 2, and a discount factor $0 < \delta < 1$. Let the initial full state be $(A, 1)$. Suppose that retaining module 1 in state $(A, 1)$ yields reward 0 and keeps the system in $(A, 1)$. Switching to module 2 yields immediate reward 1, but moves the system to $(B, 2)$. In state B, every action yields reward $-M$ and leaves the system in B, where $M > 0$.

The immediate reward differential is $\Delta_{12}(A) = 1 - 0 = 1 > 0$. However, the dynamic value of retaining module 1 is $G_1(A, 1) = \delta V^\star(A, 1)$. Because retaining module 1 keeps the system in $(A, 1)$ with zero reward indefinitely, $V^\star(A, 1) \geq 0$, and hence $G_1(A, 1) \geq 0$.

By contrast, the dynamic value of switching to module 2 is at most $G_2(A, 1) = 1 - \delta M/(1 - \delta)$. Choosing $M > (1 - \delta)/\delta$ gives $G_2(A, 1) < 0 \leq G_1(A, 1)$. It follows that $D_{12}(A) = G_2(A, 1) - G_1(A, 1) < 0$. Thus, switching has a positive immediate reward differential but a negative dynamic switching gain. A greedy rule would switch, whereas the optimal dynamic policy would retain module 1.

This reduced-form instance is consistent with the reward specification in (3.8), because the stated one-period rewards can be generated by appropriate choices of quality, latency, switching-cost, and instability terms. Therefore, a local one-period improvement is insufficient for dynamic optimality when module choices affect continuation values. □

This example is not intended as a special case of poor modeling. It highlights the general reason why the Bellman formulation is necessary: Module choices affect future states, and immediate performance can be outweighed by continuation losses.

4.7. Exact adaptive–greedy Bellman decomposition

Let

$$a^A(s) \in \arg\max_{a \in \mathcal{M}} Q^\star(s, a), \qquad a^G(s) \in \arg\max_{a \in \mathcal{M}} R_g(s, a), \tag{4.13}$$

where

$$Q^\star(s, a) = R_g(s, a) + \delta \sum_{s'} P(s' \mid s, a) V^\star(s').$$

Define

$$\Delta_R(s) = R_g(s, a^A(s)) - R_g(s, a^G(s)), \tag{4.14}$$

$$\Delta_F(s) = \delta \sum_{s'} [P(s' \mid s, a^A(s)) - P(s' \mid s, a^G(s))] V^\star(s'), \tag{4.15}$$

$$\Delta_Q(s) = Q^\star(s, a^A(s)) - Q^\star(s, a^G(s)). \tag{4.16}$$

Proposition 4.11 (Exact disagreement-state decomposition)**.** *For every state s,*

$$\Delta_Q(s) = \Delta_R(s) + \Delta_F(s). \tag{4.17}$$

If the greedy and adaptive maximizers are unique and $a^A(s) \neq a^G(s)$, then

$$\Delta_R(s) < 0, \qquad \Delta_Q(s) > 0, \qquad \Delta_F(s) > -\Delta_R(s) > 0.$$

Proof. The identity follows by subtracting the two state–action values and grouping the immediate and discounted continuation terms. Uniqueness of the greedy maximizer implies that any different action has strictly lower immediate reward, so $\Delta_R(s) < 0$. Uniqueness of the Bellman maximizer implies that the adaptive action has strictly larger optimal state–action value, so $\Delta_Q(s) > 0$. Equation (4.17) then gives $\Delta_F(s) = \Delta_Q(s) - \Delta_R(s) > -\Delta_R(s) > 0$. □

Proposition 4.11 identifies the precise mechanism behind policy disagreement. Whenever the two policies choose different unique actions, the adaptive policy accepts an immediate sacrifice only because the discounted continuation gain more than compensates for it.

4.8. Zero-friction equivalence under exogenous operating dynamics

The absence of switching friction does not make greedy control optimal in an arbitrary MDP. Equivalence requires the stronger computational specialization in (3.7).

Proposition 4.12 (Zero-friction adaptive–greedy equivalence)**.** *Suppose that (i) $C(i, j) = 0$ for all i, j; (ii) the one-period reward, apart from the switching term, depends on y and the selected action but not on the previously active module; and (iii) $P(y' \mid y, m, a) = P(y' \mid y)$. Then the optimal value is independent of the active module, and under a common tie-breaking rule,*

$$\pi^\star(y, m) = \pi^{\text{greedy}}(y, m) \qquad \text{for all } (y, m).$$

Proof. Under the stated conditions, let $W(y)$ denote a value function independent of m. The Bellman operator maps such functions into functions independent of m:

$$(\mathcal{T}W)(y, m) = \max_a \left\{ R_g(y, a) + \delta \sum_{y'} P(y' \mid y) W(y') \right\}.$$

The continuation term is common to every action. Hence the maximizing action is any maximizer of $R_g(y, a)$, which is the greedy action under the common tie-breaking rule. Contraction and uniqueness of the fixed point imply that the optimal value has this module-independent form. □

This proposition isolates the source of dynamic advantage in the computational model: Positive switching friction makes the inherited active module consequential. It should not be interpreted as a general statement for controlled exogenous transitions.

4.9. Near-optimality of greedy and threshold policies

Define the discounted continuation spread at state s by

$$\omega(s) = \max_a \delta \sum_{s'} P(s' \mid s, a) V^\star(s') - \min_a \delta \sum_{s'} P(s' \mid s, a) V^\star(s'). \tag{4.18}$$

Proposition 4.13 (Greedy near-optimality bound)**.** *If $\omega(s) \le \varepsilon$ for every state, then the greedy policy satisfies*

$$\left\| V^\star - V^{\pi^{\text{greedy}}} \right\|_\infty \le \frac{\varepsilon}{1 - \delta}. \tag{4.19}$$

More generally, if a deterministic policy $\widetilde{\pi}$, including a threshold policy, satisfies

$$V^\star(s) - Q^\star(s, \widetilde{\pi}(s)) \le \varepsilon \quad \textit{for all } s,$$

then

$$\|V^\star - V^{\widetilde{\pi}}\|_\infty \le \frac{\varepsilon}{1-\delta}.$$

Proof. Because the greedy action maximizes immediate reward, its Bellman loss relative to the optimal action can be no larger than the spread of continuation terms, and is therefore bounded by ε. The standard approximate-policy performance bound follows by applying the policy Bellman operator and the contraction inequality:

$$\|V^\star - V^\pi\|_\infty \le \varepsilon + \delta \|V^\star - V^\pi\|_\infty.$$

Rearranging yields the result. The same argument applies to any policy with a uniform statewise Bellman loss bounded by ε. □

The bound provides a practical complexity criterion. A simple greedy or threshold rule is adequate when continuation values vary little across actions or when its statewise Bellman margin is uniformly small. Full dynamic optimization is most valuable when disagreement states have material continuation spreads and are visited sufficiently often.

4.10. Path-dependent switching costs and partial observability

Assumption 4.4 abstracts from the cache state, warm starts, recent module use, and synchronization history. These effects can be represented by augmenting the state with a finite memory variable h_t and replacing $C(i, j)$ by $C(i, j, h_t)$. The finite discounted MDP results, including existence, contraction, and the Bellman decomposition, continue to hold on the augmented state (y_t, m_t, h_t). The pairwise hysteresis band then becomes memory-state dependent:

$$-\gamma c_{ji}(h) < B_{ij}(y, h) < \gamma c_{ij}(h).$$

Accordingly, a single fixed hysteresis band need not describe systems with warm-start and cold-start asymmetry.

Full observability can likewise be relaxed. If the planner observes signals rather than the true state, the posterior belief

$$b_t(s) = \Pr(s_t = s \mid o_{0:t}, a_{0:t-1})$$

forms the state of a belief-space MDP.Stationary optimality then applies in belief space under the usual finite partially observable Markov decision process (POMDP) formulation, although the threshold and hysteresis regions are defined over beliefs rather than physical states and computation becomes substantially more demanding. The fully observed model therefore serves as an interpretable benchmark and as a building block for these extensions.

4.11. Structural implications

The structural results establish the logic of adaptive module orchestration. First, optimal deployment is state dependent unless a strong universal dominance condition holds. Second, positive transition

friction creates pairwise continuation regions and local hysteresis. Third, increasing latency pressure can shift preference toward faster modules under a transformed-latency single-crossing condition. Fourth, every unique adaptive–greedy disagreement has an exact interpretation: The adaptive action sacrifices immediate reward only when a larger continuation gain produces a positive Bellman advantage. Fifth, under action-independent exogenous dynamics, removing switching friction eliminates the active-module channel and makes adaptive and greedy control equivalent. Sixth, simple policies are provably near optimal when their statewise Bellman loss is small. Finally, memory-dependent costs and partial observability preserve the dynamic-programming foundation but replace fixed physical-state regions by augmented-state or belief-state regions.

These results provide theoretical support for the computational evaluation reported in Section 6 [20, 32, 33]. Static policies ignore state-dependent preference. Latency-dominant and quality-dominant policies ignore the interaction between speed and analytical value. Greedy switching ignores continuation values and transition-induced instability. The adaptive switching policy explicitly accounts for all of these effects through the Bellman optimization problem [13].

5. Solution method and policy characterization

In this section, we present the solution method for the adaptive switching optimization model and describe how the resulting policy can be characterized computationally [32, 33]. Sections 3 and 4 establish that the adaptive switching problem is a finite discounted Markov decision process and that optimal decisions are determined by Bellman action values [20, 32, 33]. We translate those theoretical objects into a reproducible computational procedure.

The solution approach has five components. First, value iteration is used to compute the optimal value function. Second, an approximately optimal stationary switching policy is extracted from the computed Bellman action values. Third, continuation and switching regions are identified from dynamic switching gains. Fourth, benchmark policies are defined for computational comparison. Fifth, complexity and implementation details are summarized to support reproducibility.

5.1. Bellman action values and value iteration

For any bounded value function $V : \mathcal{Y} \times \mathcal{M} \to \mathbb{R}$, define the Bellman action value of selecting module $a \in \mathcal{M}$ in state (y, m) as

$$G_a(y, m; V) = R(y, m, a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y, m, a) V(y', a). \tag{5.1}$$

The continuation value is evaluated at (y', a) because the selected module a becomes the active module in the next decision epoch. The Bellman operator can then be written as

$$(\mathcal{T}V)(y, m) = \max_{a \in \mathcal{M}} G_a(y, m; V). \tag{5.2}$$

Under Assumption 3.1, $\mathcal{T}$ is a contraction. Hence, the optimal value function can be computed by value iteration [32, 33].

Starting from an arbitrary bounded initialization $V_0 : \mathcal{Y} \times \mathcal{M} \to \mathbb{R}$, define

$$V_{k+1}(y, m) = \max_{a \in \mathcal{M}} \left\{ R(y, m, a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y, m, a) V_k(y', a) \right\}, \qquad (y, m) \in \mathcal{Y} \times \mathcal{M}. \tag{5.3}$$

The sequence $\{V_k\}_{k\geq 0}$ converges uniformly to $V^\star$. At iteration k, the associated greedy policy with respect to V_k is

$$\pi_k(y,m) \in \arg\max_{a\in\mathcal{M}} G_a(y,m;V_k). \tag{5.4}$$

The iteration is terminated when

$$\|V_{k+1} - V_k\|_\infty \leq \varepsilon, \tag{5.5}$$

where $\varepsilon > 0$ is a prescribed numerical tolerance. Since the Bellman operator is a δ-contraction, the residual-based value error satisfies

$$\|V_{k+1} - V^\star\|_\infty \leq \frac{\delta}{1-\delta}\|V_{k+1} - V_k\|_\infty. \tag{5.6}$$

Thus, if the stopping rule (5.5) holds, then

$$\|V_{k+1} - V^\star\|_\infty \leq \frac{\delta}{1-\delta}\varepsilon. \tag{5.7}$$

This bound provides a computable certificate for the accuracy of the value function.

5.2. Policy extraction and approximation guarantee

Let $\widehat{V}$ denote the final value function returned by value iteration, and define the extracted policy

$$\widehat{\pi}(y,m) \in \arg\max_{a\in\mathcal{M}} \left\{ R(y,m,a) + \delta \sum_{y'\in\mathcal{Y}} P(y' \mid y,m,a)\widehat{V}(y',a) \right\}. \tag{5.8}$$

The policy $\widehat{\pi}$ is the implementable adaptive switching policy used in the computational study.

The following proposition provides a standard performance guarantee for the extracted policy [32,33].

Proposition 5.1 (Approximate policy guarantee)**.** *Suppose* $\|\widehat{V} - V^\star\|_\infty \leq e$. *Let* $\widehat{\pi}$ *be greedy with respect to* $\widehat{V}$ *as in* (5.8). *Then,*

$$\|V^{\widehat{\pi}} - V^\star\|_\infty \leq \frac{2\delta}{1-\delta}e.$$

In particular, if value iteration is stopped according to (5.5)*, then*

$$\|V^{\widehat{\pi}} - V^\star\|_\infty \leq \frac{2\delta^2}{(1-\delta)^2}\varepsilon.$$

Proof. Let $T_{\widehat{\pi}}$ denote the policy-specific Bellman operator:

$$(T_{\widehat{\pi}}V)(y,m) = R(y,m,\widehat{\pi}(y,m)) + \delta \sum_{y'\in\mathcal{Y}} P(y' \mid y,m,\widehat{\pi}(y,m))V(y',\widehat{\pi}(y,m)).$$

Since $\widehat{\pi}$ is greedy with respect to $\widehat{V}$,

$$T_{\widehat{\pi}}\widehat{V} = \mathcal{T}\widehat{V}.$$

Using the triangle inequality,

$$\|V^\star - V^{\widehat{\pi}}\|_\infty = \|\mathcal{T}V^\star - T_{\widehat{\pi}}V^{\widehat{\pi}}\|_\infty$$

is bounded by

$$\|\mathcal{T}V^\star - \mathcal{T}\widehat{V}\|_\infty + \|\mathcal{T}\widehat{V} - T_{\widehat{\pi}}V^{\widehat{\pi}}\|_\infty.$$

The first term is at most

$$\delta\|V^\star - \widehat{V}\|_\infty \le \delta e.$$

For the second term, since $\mathcal{T}\widehat{V} = T_{\widehat{\pi}}\widehat{V}$,

$$\|\mathcal{T}\widehat{V} - T_{\widehat{\pi}}V^{\widehat{\pi}}\|_\infty = \|T_{\widehat{\pi}}\widehat{V} - T_{\widehat{\pi}}V^{\widehat{\pi}}\|_\infty \le \delta\|\widehat{V} - V^{\widehat{\pi}}\|_\infty.$$

Moreover,

$$\|\widehat{V} - V^{\widehat{\pi}}\|_\infty \le \|\widehat{V} - V^\star\|_\infty + \|V^\star - V^{\widehat{\pi}}\|_\infty \le e + \|V^\star - V^{\widehat{\pi}}\|_\infty.$$

Combining the inequalities gives

$$\|V^\star - V^{\widehat{\pi}}\|_\infty \le \delta e + \delta e + \delta\|V^\star - V^{\widehat{\pi}}\|_\infty.$$

Therefore,

$$(1-\delta)\|V^\star - V^{\widehat{\pi}}\|_\infty \le 2\delta e,$$

which proves the first claim. The second claim follows from (5.7). □

This guarantee is useful for computational reporting because it links the numerical stopping tolerance to the quality of the extracted adaptive switching policy.

5.3. Algorithmic procedure

Table 3 summarizes the value-iteration procedure used to compute the adaptive switching policy [32,33]. The algorithm returns the approximate optimal value function, the extracted policy, Bellman action values, and dynamic switching gains. These outputs are later used to evaluate performance and to visualize continuation and switching regions.

Table 3. Value-iteration procedure for adaptive switching optimization.

Step	Procedure
1	Initialize $V_0(y,m)$ for all $(y,m) \in \mathcal{Y} \times \mathcal{M}$, choose tolerance $\varepsilon > 0$, and set $k = 0$.
2	For each state $(y,m) \in \mathcal{Y} \times \mathcal{M}$ and each candidate module $a \in \mathcal{M}$, compute $$G_a^k(y,m) = R(y,m,a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,m,a) V_k(y',a).$$
3	Update the value function: $$V_{k+1}(y,m) = \max_{a \in \mathcal{M}} G_a^k(y,m), \qquad \forall (y,m) \in \mathcal{Y} \times \mathcal{M}.$$
4	Extract the provisional policy: $$\pi_{k+1}(y,m) \in \arg\max_{a \in \mathcal{M}} G_a^k(y,m).$$
5	Check convergence. If $$\|V_{k+1} - V_k\|_\infty \le \varepsilon,$$ stop. Otherwise, set $k \leftarrow k+1$ and return to Step 2.
6	Return $\widehat{V} = V_{k+1}$, $\widehat{\pi} = \pi_{k+1}$, $\{G_a^k(y,m)\}_{y,m,a}$, and the induced dynamic switching gains $\widehat{D}_{ij}(y)$.

The procedure is exact for the finite-state discounted model up to the chosen numerical tolerance. It is also transparent: For each state, the Bellman action values reveal whether the selected module is retained because of quality, latency, switching friction, or continuation-value effects.

5.4. Continuation and switching region extraction

After value iteration converges, the computed Bellman action values can be used to extract continuation and switching regions. Let

$$\widehat{G}_a(y,m) = R(y,m,a) + \delta \sum_{y' \in \mathcal{Y}} P(y' \mid y,m,a) \widehat{V}(y',a)$$

denote the final approximate Bellman action value. For a state (y,i), define the estimated dynamic switching gain

$$\widehat{D}_{ij}(y) = \widehat{G}_j(y,i) - \widehat{G}_i(y,i). \tag{5.9}$$

The estimated continuation region for active module i is

$$\widehat{C}_i = \left\{ (y,i) \in \mathcal{Y} \times \mathcal{M} : \widehat{D}_{ij}(y) \le 0 \ \text{ for all } j \in \mathcal{M} \right\}. \tag{5.10}$$

The estimated switching region from module i to module $j \neq i$ is

$$\widehat{\mathcal{W}}_{ij} = \left\{ (y,i) \in \mathcal{Y} \times \mathcal{M} : \widehat{D}_{ij}(y) = \max_{k \in \mathcal{M}} \widehat{D}_{ik}(y) \ge 0 \right\}. \tag{5.11}$$

These regions are not merely computational artifacts. They operationalize the structural results of Section 4 [11, 20, 24]. For example, if switching friction increases, pairwise hysteresis bands should widen. If latency pressure increases, the extracted policy should shift toward faster modules in stressed regimes. Thus, region extraction provides a direct bridge between theory and numerical evaluation.

5.5. Tie-breaking and stability convention

Because multiple modules may attain the same Bellman action value in a state, a tie-breaking convention is needed for deterministic policy implementation. The default convention used in this paper is stability-preserving tie-breaking: If the currently active module is among the Bellman maximizers, the policy retains it. Formally, for state (y, m), define

$$\mathcal{A}^\star(y, m) = \arg\max_{a \in \mathcal{M}} \widehat{G}_a(y, m).$$

The implemented policy is

$$\widehat{\pi}(y, m) = \begin{cases} m, & m \in \mathcal{A}^\star(y, m), \\ \min \mathcal{A}^\star(y, m), & m \notin \mathcal{A}^\star(y, m), \end{cases} \tag{5.12}$$

where the second line uses a fixed deterministic ordering of modules.

This convention is consistent with the stability logic of the model [20–22]. It avoids unnecessary switching in states where continuation and switching are dynamically equivalent. Alternative tie-breaking rules can be used, but they should be reported because they can affect switching-frequency metrics even when value-function performance is unchanged.

5.6. Action-value explanation certificates

The optimal policy can be reported with a state-specific explanation certificate rather than as an opaque module label. For each feasible action, write

$$Q^\star(s, a) = \underbrace{\alpha Q_a(y)}_{\text{quality}} - \underbrace{\beta(d) g(L_a(y))}_{\text{latency}} - \underbrace{\gamma C(m, a)}_{\text{switching}} - \underbrace{\eta U_a(y)}_{\text{instability}} + \underbrace{\delta \mathbb{E}[V^\star(s') \mid s, a]}_{\text{continuation}}. \tag{5.13}$$

For the selected action and the runner-up, an operational interface can report the five components in (5.13), the immediate reward gap, the continuation-value gap, and the Bellman margin. This decomposition makes explicit whether a recommendation is driven by quality, delay, transition friction, stability, or future configuration value. States with a small Bellman margin can be flagged for supervisory review or handled by a simpler threshold rule.

5.7. Benchmark policies

The computational study compares the adaptive switching policy with benchmark policies that isolate different aspects of the decision problem [32–34]. All reported benchmark policies are implementable at the evaluation stage except for the oracle comparator, which is retained only as a nonimplementable upper bound [35]. Table 4 summarizes these benchmark policies.

Table 4. Benchmark policies for computational comparison.

Policy	Definition
Adaptive switching policy	The policy $\widehat{\pi}$ obtained from the Bellman equation. It accounts for state-dependent quality, latency, switching-cost, instability, and continuation values.
Ex-ante best-static policy	A fixed module $j \in \mathcal{M}$ is used for all states and decision epochs. The module is selected from the fixed-module alternatives using a disjoint calibration sample and is then held fixed throughout evaluation. This procedure prevents evaluation-path information from entering the benchmark selection.
Latency-dominant policy	At each state (y, m), the policy selects the module with the smallest response latency: $$\pi^{\text{lat}}(y, m) \in \arg\min_{a \in \mathcal{M}} L_a(y).$$ This benchmark emphasizes speed but ignores analytical quality and continuation values.
Quality-dominant policy	At each state (y, m), the policy selects the module with the largest analytical quality: $$\pi^{\text{qual}}(y, m) \in \arg\max_{a \in \mathcal{M}} Q_a(y).$$ This benchmark emphasizes quality but ignores latency-pressure, switching-cost, and future-state consequences.
Greedy one-period policy	At each state (y, m), the policy selects the module with the highest immediate reward: $$\pi^{\text{greedy}}(y, m) \in \arg\max_{a \in \mathcal{M}} R(y, m, a).$$ This benchmark includes the one-period reward tradeoff but ignores continuation values.
No-switch policy	The system retains its initially active module throughout the horizon. This benchmark isolates the value of switching capability.
Oracle policy	A nonimplementable upper-bound benchmark that uses future regime realizations or hindsight-optimized actions. It is used only to measure an oracle gap.

These benchmarks correspond directly to the limitations identified in Section 4. Static and no-switch policies ignore adaptive state-dependent deployment. Latency- and quality-dominant policies optimize a single attribute. Greedy switching ignores continuation values. The adaptive switching policy accounts for all modeled components simultaneously.

5.8. Performance metrics for policy evaluation

Let π denote a policy evaluated over a finite simulation horizon T. The primary performance metric is cumulative operational reward:

$$\text{Reward}^{\pi}(T) = \sum_{t=0}^{T-1} R(y_t, m_t, a_t^{\pi}). \tag{5.14}$$

For diagnostic purposes, the cumulative reward is decomposed into its operational components:

$$\text{Quality}^{\pi}(T) = \frac{1}{T} \sum_{t=0}^{T-1} Q_{a_t^{\pi}}(y_t), \tag{5.15}$$

$$\text{Latency}^{\pi}(T) = \frac{1}{T}\sum_{t=0}^{T-1} L_{a_t^{\pi}}(y_t), \tag{5.16}$$

$$\text{Switches}^{\pi}(T) = \sum_{t=0}^{T-1} \mathbf{1}\{a_t^{\pi} \neq m_t\}, \tag{5.17}$$

$$\text{SwitchCost}^{\pi}(T) = \sum_{t=0}^{T-1} C(m_t, a_t^{\pi}), \tag{5.18}$$

$$\text{Instability}^{\pi}(T) = \sum_{t=0}^{T-1} U_{a_t^{\pi}}(y_t). \tag{5.19}$$

These metrics distinguish whether performance differences are driven by higher quality, lower latency, fewer switches, lower transition costs, or reduced instability [13, 14, 58].

When an oracle benchmark is available, the oracle gap is defined as

$$\text{OracleGap}^{\pi}(T) = \text{Reward}^{\text{oracle}}(T) - \text{Reward}^{\pi}(T). \tag{5.20}$$

For comparisons against an implementable benchmark policy π^b, define the benchmark regret as

$$\text{Regret}^{\pi|\pi^b}(T) = \sum_{t=0}^{T-1} \left[R(y_t, m_t, a_t^{\pi^b}) - R(y_t, m_t, a_t^{\pi})\right]. \tag{5.21}$$

A negative value of $\text{Regret}^{\pi|\pi^b}(T)$ indicates that policy π outperforms the benchmark over the realized trajectory.

5.9. Common random numbers and fair policy comparison

To ensure fair comparison across policies, the computational study uses common random numbers whenever simulation is required. Specifically, each policy is evaluated on the same set of sampled regime paths, workload shocks, communication disturbances, and auxiliary operational transitions. Let ω_n denote the n-th simulation path, $n = 1, \ldots, N_{\text{MC}}$. For policy π, the Monte Carlo estimate of cumulative reward is

$$\widehat{\text{Reward}}^{\pi}(T) = \frac{1}{N_{\text{MC}}}\sum_{n=1}^{N_{\text{MC}}} \text{Reward}^{\pi}(T; \omega_n). \tag{5.22}$$

Using identical paths $\{\omega_n\}$ for all policies reduces comparison noise and makes paired statistical tests possible in the computational experiments [32, 33].

5.10. Computational complexity and implementation

The value-iteration update in (5.3) requires evaluating each candidate module for each full state (y, m). Because the selected module deterministically becomes the next active module, the structured transition sums only over $y' \in \mathcal{Y}$. Thus, if the transition kernel over $\mathcal{Y}$ is dense, one iteration has computational complexity

$$O(|\mathcal{Y}|^2|\mathcal{M}|^2). \tag{5.23}$$

Equivalently, since $|\mathcal{S}| = |\mathcal{Y}||\mathcal{M}|$, this can be written as $O(|\mathcal{S}||\mathcal{Y}||\mathcal{M}|)$. If the transition kernel is sparse and each state-action pair has at most K successor non-module states, the per-iteration complexity becomes

$$O(|\mathcal{Y}||\mathcal{M}|^2 K). \tag{5.24}$$

The storage requirement is $O(|\mathcal{Y}||\mathcal{M}|)$ for the value function and $O(|\mathcal{Y}||\mathcal{M}|^2)$ for storing Bellman action values or policy diagnostics.

The finite-state formulation is appropriate for the present computational study because operating regimes, workload levels, and auxiliary stress indicators can be discretized [33, 41, 42]. For larger state spaces, approximate dynamic programming, fitted value iteration, rollout, or reinforcement-learning methods could be used. In this paper, we focus on the exact finite-state formulation to preserve interpretability and to make the theoretical mechanisms directly observable in the numerical experiments [57].

5.11. Summary

In this section, we have described how the optimal adaptive switching policy is computed and characterized. Value iteration solves the Bellman equation with a computable approximation guarantee. The extracted policy is obtained from Bellman action values, and continuation and switching regions are recovered from dynamic switching gains. Benchmark policies isolate the roles of static assignment, latency dominance, quality dominance, no-switch behavior, and greedy switching. Performance metrics decompose total reward into quality, latency, switching, and instability components. These elements provide the methodological bridge between the structural results in Section 4 and the computational experiments in Section 6.

6. Computational experiments and results

In this section, we report computational experiments designed to evaluate the adaptive switching optimization framework [20, 32, 33]. Because the preceding sections derive structural predictions from the finite-state discounted model, the computational study tests whether those predictions are reflected in simulated operational trajectories. The experiments focus on four mechanisms: state-dependent module preference, switching behavior under transition friction, latency-driven preference shifts under stressed operating regimes, and computational scalability of the finite-state dynamic programming implementation [13, 24, 58].

The computational study is not intended merely to show that a Bellman-optimal policy outperforms restricted benchmarks [14, 20, 23]. Rather, the purpose is to diagnose how switching friction, latency pressure, continuation value, and state-dependent module preference jointly shape operational behavior. To this end, the experiment suite combines mechanism diagnostics, switching-cost and latency-disutility sensitivity analysis, ablation analysis, scalability diagnostics, and a six-scenario regime stress-test battery. The stress-test battery includes baseline mixed operation, stable-dominant operation, volatility bursts, sustained stress, recovery cycles, and high-latency crisis conditions. Across these experiments, the policies are evaluated on common sampled trajectories, so policy comparisons are paired and not driven by independent sampling variation [36].

The computational workflow is fully output-based. Raw episode-level and step-level simulation outputs are saved first, summary tables are then computed from those files, and all figures are generated

from the saved CSV outputs rather than from hidden intermediate states. The design makes the numerical pipeline auditable: The values reported in the tables correspond directly to the plotted quantities, and the figures can be regenerated by re-running the post-processing scripts on the saved CSV files.

6.1. Experimental design, calibration, and reproducibility

The experiments are organized around four objectives [20, 32, 33]. First, they evaluate whether the adaptive switching policy improves long-run operational reward relative to benchmark policies that ignore one or more components of the dynamic optimization problem. Second, they examine whether switching behavior is shaped by transition friction and continuation-value effects, as predicted by the switching hysteresis result in Theorem 4.5 [21–23]. Third, they test whether latency disutility shifts module usage toward lower-latency modules, consistent with Proposition 4.9. Fourth, they decompose total performance into quality, latency, switching cost, instability, and benchmark losses in order to explain the source of policy differences [13, 38, 58].

The experiment design is summarized in Table 5.

Table 5. Experimental objectives and corresponding theoretical mechanisms.

Experimental objective	Theoretical mechanism	Diagnostic evidence
Evaluate adaptive switching against benchmark policies	State-dependent module preference and Bellman optimality	Discounted reward, benchmark gaps, paired reward differences
Assess switching behavior under transition friction	Continuation values and switching hysteresis	Switching frequency, continuation rate, switching-cost sensitivity
Assess latency-driven module substitution	Latency–quality tradeoff and pairwise preference shifts	Module-use shares under increasing latency pressure
Assess robustness under operational stress tests	Scenario-dependent module preference and operational fragility	Scenario-level losses, regime-specific losses, reward–latency frontier

Simulation environment and implementation. Following the model in Section 3, the computational state is instantiated as $s_t = (r_t, d_t, z_t, m_t)$, where r_t is the operating regime, d_t is the workload or urgency level, z_t captures auxiliary stress indicators, and m_t is the currently active module. The action $a_t \in \mathcal{M}$ selects the next active module, so that module deployment itself is the control decision [2, 32, 33].

The operating-regime set is $\mathcal{R} = \{\text{stable}, \text{volatile}, \text{stressed}, \text{recovery}\}$. Stable regimes represent low workload, reliable communication, and limited urgency. Volatile regimes represent intermittent disruption and uncertain transitions. Stressed regimes represent high workload, severe urgency, degraded communication, and a high delay penalty. Recovery regimes represent declining stress and a gradual return toward stable operation.

The computational environment includes the heterogeneous decision-module set $\mathcal{M} = \{\text{Fast rule-based}, \text{Legacy optimization}, \text{ML predictor}, \text{Advanced AI}\}$, whose elements differ in quality, latency, instability, and switching characteristics [7, 12, 16].

The fast rule-based module has low latency but limited quality. The legacy optimization module has moderate latency and stable quality. The ML predictor has higher analytical capability but greater latency exposure. The advanced AI module provides the highest nominal quality under stable conditions but may incur higher latency and instability under stressed conditions [10].

The numerical module profiles are normalized operational representations rather than direct measurements of a particular deployed system. Their ordering is chosen to reflect the relative quality, latency, and implementation-burden patterns documented in the cited literature. The sensitivity, stress-test, and ablation analyses assess whether the reported mechanisms depend on this baseline normalization.

The one-period reward used in the computational study follows the reward specification in (3.8). The baseline experiment fixes the main reward weights and solves the finite-state discounted Markov decision process by value iteration [32, 33]. The computed value function is then used to extract the adaptive switching policy. We evaluate the extracted policy and benchmark policies over common sampled regime paths.

Table 6. Implementation summary for the computational experiments.

Item	Setting
State representation	$s_t = (r_t, d_t, z_t, m_t)$, where r_t is the regime, d_t is workload, z_t is auxiliary stress, and m_t is the active module
Regimes	Stable, volatile, stressed, and recovery
Modules	Fast rule-based, legacy optimization, machine-learning predictor, and advanced AI
Policies compared	Adaptive switching, greedy one-period, best static, no-switch, always-fast, always-legacy, always-ML, and always-AI benchmarks
Solution method	Value iteration for the finite discounted Markov decision process
Monte Carlo design	Common random numbers across policies for paired comparison
Sensitivity parameters	The sensitivity analysis varies the switching-cost weight γ and the scaling factor applied to the latency-disutility function $\beta(d)$.
Stress-test scenarios	Baseline mixed, stable-dominant, volatility burst, sustained stress, recovery cycle, and high-latency crisis
Computational battery	Mechanism diagnostics, switching-cost sensitivity, latency-pressure sensitivity, ablation analysis, scalability diagnostics, scenario-level stress testing, regime-specific reward-loss decomposition, and module-orchestration analysis
Output structure	Raw simulation logs, policy-level summaries, paired comparison tables, scenario robustness summaries, regime loss summaries, scalability summaries, and figure-generation CSV files
Primary metrics	Reward, quality, latency, switching frequency, switching cost, instability, continuation rate, module-use share, benchmark loss, and runtime
Reproducibility design	Policy comparisons are evaluated on common sampled trajectories. Raw simulation outputs are saved before aggregation, and all tables and figures are generated from saved CSV files through separate post-processing scripts

Table 6 reports the implementation settings used in the computational experiments.

Evaluation protocol. We evaluate the benchmark policies defined in Section 5.7 using the performance metrics introduced in Section 5.8. For Monte Carlo reporting, each metric is averaged over common simulation paths ω_n, $n = 1, \ldots, N_{\mathrm{MC}}$. Reward gaps are reported as the adaptive reward minus the benchmark reward, so positive values indicate that the benchmark performs worse than adaptive switching. This convention is used consistently in the scenario-level robustness table, regime-specific loss summary, and ablation analysis [32, 33].

Computational battery and reproducibility design. The computational battery is organized to test both structural mechanisms and operational robustness [20, 24, 58]. The first layer evaluates the baseline adaptive policy against greedy, best-static, no-switch, and single-module benchmarks. The second layer varies switching-cost and latency-disutility parameters to examine whether the policy exhibits continuation behavior, switching hysteresis, and latency-driven module substitution. The third layer conducts ablation analysis by removing or simplifying selected reward components, allowing the contribution of switching friction, latency pressure, and instability penalties to be separated. The fourth layer evaluates scalability by comparing value-iteration runtime under dense and sparse transition structures. The fifth layer applies a regime stress-test battery across six operating scenarios and reports scenario-level robustness, regime-specific losses, and module-use shares [13, 14].

The multi-layer design is intended to avoid relying on a single aggregate performance comparison. Instead, the experiments examine whether the theoretical mechanisms derived earlier are visible in policy behavior, reward decomposition, switching frequency, module-use composition, and computational runtime. The reporting structure therefore combines policy-level averages, paired benchmark gaps, scenario-level robustness summaries, regime-specific loss decompositions, and visual diagnostics.

All computational outputs are generated through a two-stage reporting pipeline. The simulation stage stores raw run-level outputs, and the reporting stage reads those saved outputs to construct tables and figures. No figure is generated from unreported in-memory quantities. The separation between simulation and reporting makes the computational evidence reproducible and allows the frontier plots, orchestration panels, reward-loss figures, and full-page ablation table to be regenerated from the saved CSV files.

6.2. Baseline comparison, mechanism diagnostics, and sensitivity analysis

Figure 1 summarizes the main computational diagnostics. Figure 1(a) reports the main policy comparison. The adaptive policy achieves the highest average discounted reward, followed closely by the greedy and best-static benchmarks. The small gap relative to the greedy policy indicates that myopic switching can be competitive in benign parameter regions. However, the poor performance of the no-switch and always-AI policies highlights the operational risk of excessive inertia and static reliance on a high-complexity module.

Figures 1(b) and 1(c) examine the role of switching friction. As the switching-cost weight γ increases, the average reward declines gradually and the switching frequency decreases sharply. This pattern is consistent with the continuation-region and hysteresis logic established in the structural analysis: Positive switching friction discourages frequent reassignment and expands the set of states in which retaining the current module is preferable [20, 23, 24].

Figure 1(d) reports module-use shares as latency pressure increases. The fast rule-based module becomes increasingly dominant as the latency-disutility scaling factor increases, while more complex ML and AI modules are used less frequently. The result confirms the central latency–quality tradeoff of the model: The most accurate or sophisticated module need not be operationally optimal when response time is highly penalized [13, 38, 58].

Figure 1(e) presents the adaptive gain over the greedy benchmark across switching-cost and latency-pressure regimes. The gain is not uniform across the parameter space, which is expected because greedy switching can approximate the adaptive policy when one-period preferences are nearly aligned with dynamic continuation values. The gain becomes most informative in regions where latency pressure and switching friction jointly determine whether the planner should switch or continue.

Finally, Figure 1(f) reports a scalability diagnostic for value iteration. Runtime increases with the number of states, particularly under dense transition structures, whereas sparse transition structures remain computationally more tractable. The result supports the use of finite-state dynamic programming as a transparent and reproducible validation framework for the proposed switching model [32, 33].

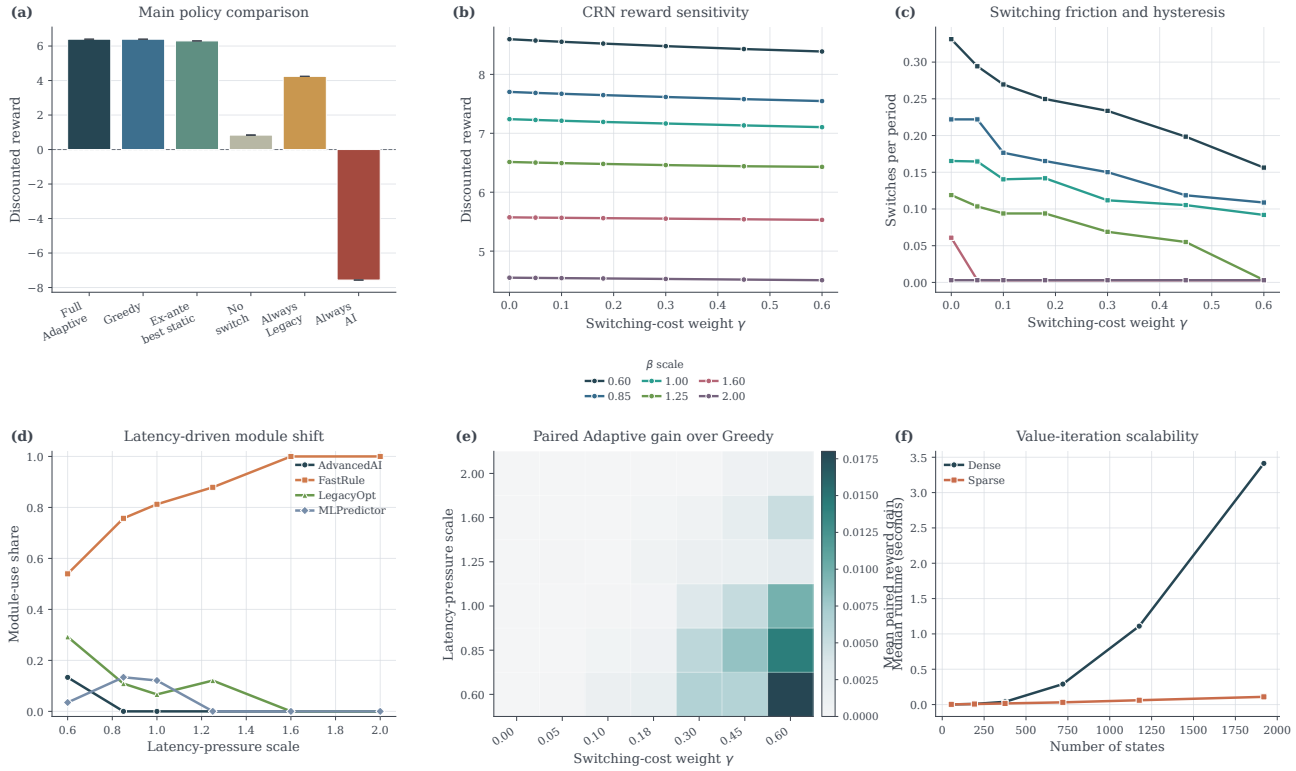

Figure 1. Mechanism diagnostics and robustness of adaptive switching. (a) compares the average discounted reward across adaptive, greedy, static, no-switch, and single-module benchmarks. (b) shows reward sensitivity to the switching-cost weight γ under different latency-disutility scaling factors. (c) reports the corresponding switching frequency and illustrates continuation behavior under switching friction. (d) shows how module-use shares shift toward the fast module as latency pressure increases. (e) reports the adaptive gain over the greedy benchmark across switching-cost and latency-pressure regimes. (f) reports value-iteration scalability under dense and sparse transition structures.

6.3. *Scenario- and regime-level stress-test results*

We next evaluate the policies under a regime stress-test battery. The stress tests include baseline mixed operation, stable-dominant operation, volatility bursts, sustained stress, recovery cycles, and high-latency crisis conditions. The purpose is to assess whether the adaptive switching logic remains operationally robust when the transition environment changes [11, 14, 36].

Figure 2 reports the reward–latency–quality tradeoff under the stress-test scenarios [20, 32, 33]. Figure 2(a) shows that static reliance on the advanced AI module is operationally fragile: It incurs high latency exposure and large reward losses despite its nominal analytical sophistication. The no-switch policy also performs poorly because it cannot adapt module deployment to changing regimes. In contrast, adaptive switching, greedy switching, best-static assignment, and the always-fast benchmark remain in the efficient region under the tested scenarios.

Figure 2(b) provides a magnified view of the efficient operating region. These policies are close in average reward, indicating that simple policies can be competitive when latency pressure is dominant. Nevertheless, adaptive switching remains a principled benchmark because it jointly accounts for latency, switching friction, instability, and continuation value [58].

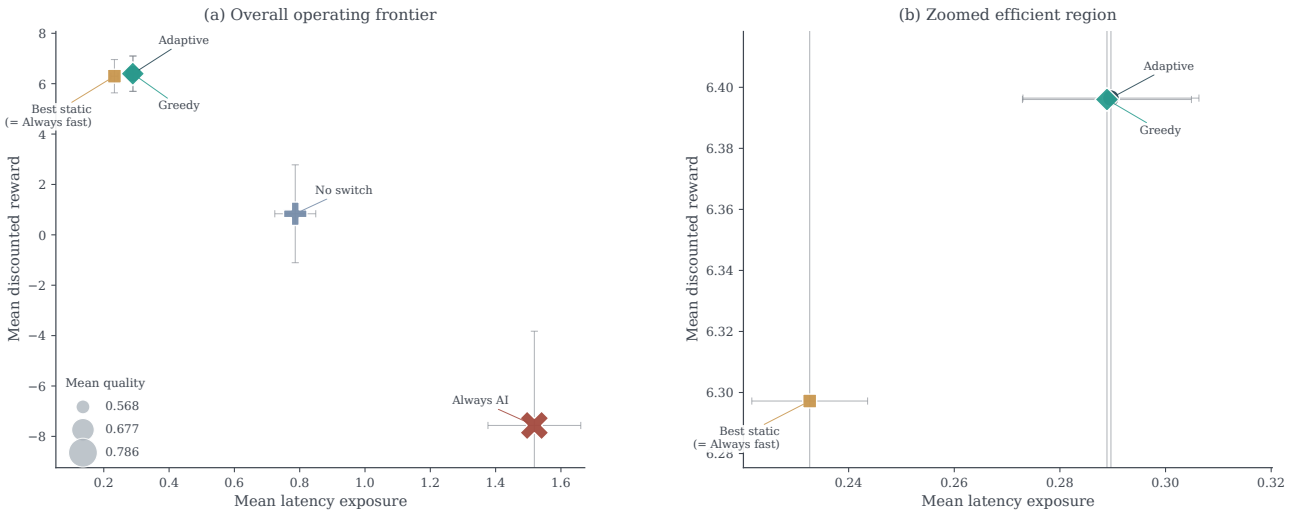

Figure 2. Reward–latency–quality tradeoff under regime stress-test scenarios: (a) overall frontier across policies; and (b) magnified view of the efficient operating region. Bubble size represents mean decision quality. The advanced-AI-only benchmark is operationally fragile under latency and instability exposure, whereas adaptive switching remains in the efficient region.

Table 7 reports scenario-level robustness diagnostics under the same stress-test battery. The table reports full-adaptive reward, latency, and quality, together with benchmark losses relative to full adaptive switching. For benchmark policy π, the reported scenario-level loss is

$$\widehat{L}^{\pi} = \widehat{\text{Reward}}^{\widehat{\pi}_{\text{adaptive}}} - \widehat{\text{Reward}}^{\pi}.$$

Positive values therefore indicate worse benchmark performance. The ex-ante best-static comparator coincides with Always Fast in every scenario because the disjoint calibration sample selects the FastRule

module. The results show that static reliance on the advanced AI module incurs especially large losses under volatility bursts, sustained stress, and high-latency crisis conditions. These losses indicate that nominal analytical sophistication alone is insufficient for operational optimality when latency and instability penalties are material [10, 12, 17].

Table 7. Scenario-level robustness summary under regime stress-test simulations.

Scenario	Adaptive reward	Adaptive latency	Adaptive quality	Greedy loss	Static loss	Fast loss	AI loss	No-switch loss
Baseline mixed	7.190	0.283	0.623	0.000	0.098	0.098	9.999	3.926
Stable-dominant	8.469	0.358	0.713	0.001	0.291	0.291	5.730	2.220
Volatility burst	6.045	0.251	0.569	0.001	0.028	0.028	15.487	6.161
Sustained stress	4.561	0.264	0.517	0.000	0.010	0.010	21.296	8.491
Recovery cycle	7.806	0.318	0.661	0.000	0.161	0.161	7.503	2.924
High-latency crisis	4.308	0.265	0.515	0.000	0.009	0.009	23.764	9.637

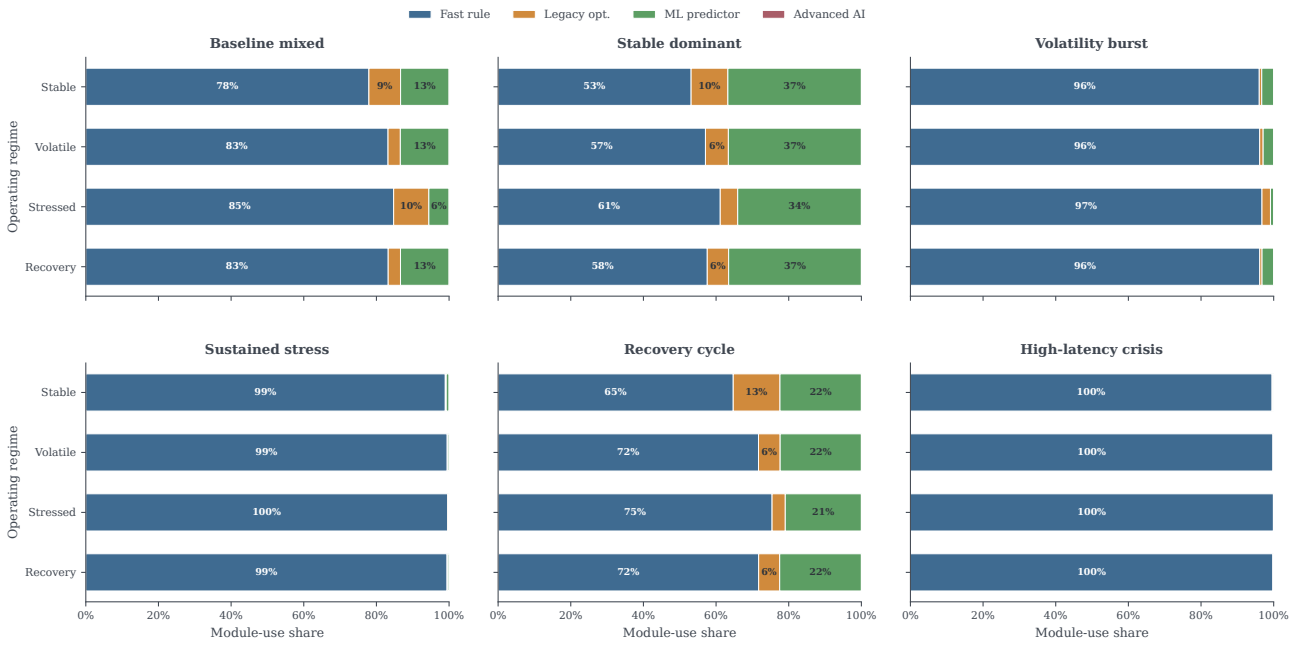

Figure 3. Scenario-specific module orchestration under adaptive switching. Each panel reports regime-level module-use shares within one stress-test scenario. Stable-dominant environments permit greater use of legacy and ML-based modules, whereas volatility bursts, sustained stress, and high-latency crisis conditions drive the policy toward the fast rule-based module.

Scenario-specific module orchestration. Figure 3 reports module-use shares under adaptive switching across the six stress-test scenarios. The figure clarifies that module orchestration is scenario dependent. In stable-dominant scenarios, the adaptive policy diversifies away from the fast rule-based module toward legacy and ML-based modules. Under volatility bursts, sustained stress, and high-latency crisis scenarios, the policy concentrates on the low-latency fast rule-based module. Recovery cycles exhibit an intermediate pattern, with the fast module still used frequently but with nontrivial use of ML-based modules.

The pattern is consistent with the latency–quality tradeoff in the reward function [7, 13, 58]. The adaptive policy does not simply select the most sophisticated module. It shifts toward low-latency operation when stress and urgency dominate, and it uses more quality-oriented modules when the operating environment allows greater analytical depth [16].

Regime-specific benchmark losses. Figure 4 reports regime-specific benchmark losses relative to full adaptive switching. The losses are calculated from regime-conditioned mean immediate rewards and are then averaged across the six stress-test scenarios. Greedy switching remains virtually indistinguishable from adaptive switching in every regime. Its mean loss ranges from approximately −0.00009 to 0.00007, indicating that the two policies select nearly identical actions over most visited states. The small negative mean loss in the stressed regime means that greedy switching attains a marginally higher regime-conditioned immediate reward there. This does not contradict the dynamic-programming result because the adaptive policy optimizes discounted long-run value rather than immediate reward alone.

The ex-ante best-static benchmark, which coincides with the always-fast policy in all six scenarios, also remains close to adaptive switching. Its mean loss ranges from approximately 0.002 to 0.007. By contrast, the no-switch policy incurs mean losses between 0.325 and 0.381, demonstrating the cost of retaining the currently active module when operating conditions change. The always-AI benchmark performs worst in every regime, with mean losses ranging from 0.821 to 0.947. Its largest observed scenario-specific loss occurs in the stressed regime, where the loss reaches 1.605. These results confirm that nominal analytical sophistication does not ensure operational optimality when latency and instability penalties are material [10, 17, 39].

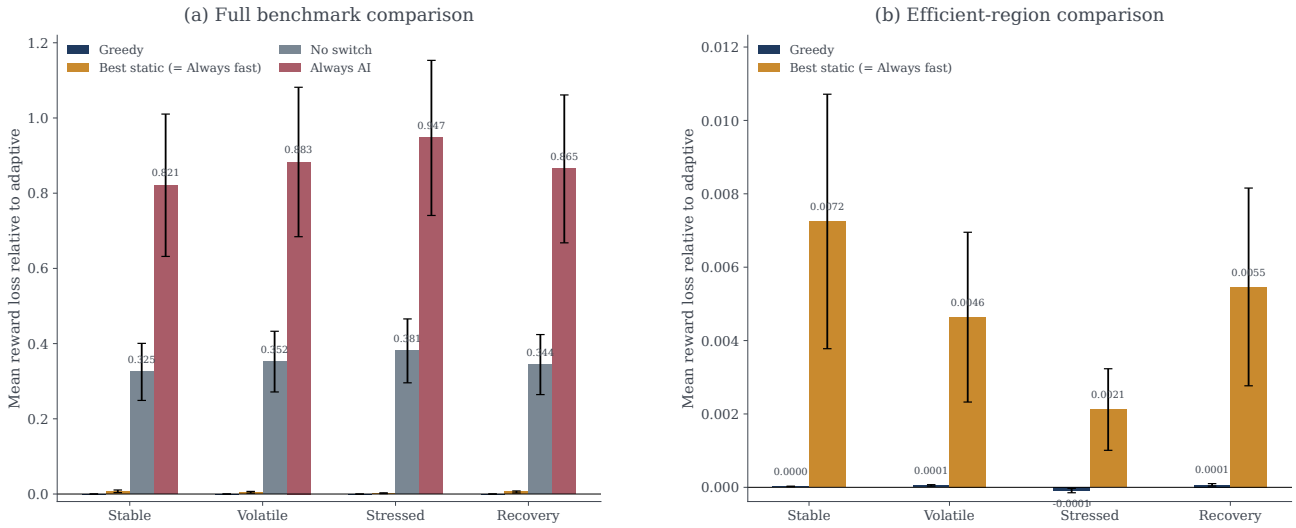

Figure 4. Regime-specific reward loss relative to full adaptive switching. Figure 4(a) reports the full benchmark comparison, including greedy, ex-ante best-static, no-switch, and always-AI policies. Figure 4(b) enlarges the efficient region containing the greedy and best-static benchmarks. Bars report mean loss across the six stress-test scenarios, and error bars report the standard error across scenarios. The ex-ante best-static policy coincides with the always-fast policy in every scenario. Greedy switching remains nearly indistinguishable from adaptive switching, whereas the no-switch and always-AI benchmarks incur substantially larger losses.

Table 8. Regime-specific immediate-reward loss relative to full adaptive switching.

	Greedy		Best static (= always fast)		No switch		Always AI	
Regime	Mean loss	Max loss	Mean loss	Max loss	Mean loss	Max loss	Mean loss	Max loss
Stable	0.0000	0.0001	0.007	0.022	0.325	0.573	0.821	1.429
Volatile	0.0001	0.0001	0.005	0.014	0.352	0.614	0.883	1.517
Stressed	-0.0001	0.0000	0.002	0.007	0.381	0.656	0.947	1.605
Recovery	0.0001	0.0002	0.005	0.017	0.344	0.604	0.865	1.493

Table 8 summarizes the regime-specific benchmark losses. For benchmark policy π, loss in operating regime s is defined as

$$\widehat{L}_s^{\pi} = \widehat{r}_s^{\widehat{\pi}_{\text{adaptive}}} - \widehat{r}_s^{\pi},$$

where $\widehat{r}_s$ denotes the mean immediate reward conditional on regime s. Mean loss is calculated by averaging the six scenario-specific losses, whereas maximum loss is the largest loss observed among the six stress-test scenarios. Positive values therefore indicate worse benchmark performance relative to full adaptive switching.

The results reveal a clear ordering. Greedy switching is effectively equivalent to adaptive switching in terms of regime-conditioned immediate reward. Its small negative mean loss in the stressed regime is approximately -8.7×10^{-5}, indicating a marginal immediate-reward advantage rather than an exact tie. The best-static policy incurs small but consistently positive losses. The no-switch policy incurs moderate losses, whereas static reliance on the advanced-AI module produces the largest losses in every regime. The ex-ante best-static policy coincides with always fast because the disjoint calibration sample selects the FastRule module in all six scenarios. All results are based on 30 independent replications per scenario, 1000 episodes per replication, and a horizon of 250 periods.

6.4. Comprehensive policy comparison and ablation analysis

Table 9 presents the comprehensive policy comparison and objective-ablation analysis. The computational design comprises six prespecified operating scenarios, 30 independent replications per scenario, 1,000 common-random-number episodes per replication, and a horizon of 250 periods. Common random numbers ensure that all competing policies are evaluated on identical operational trajectories within each scenario–replication pair.

The descriptive performance measures reported in Table 9 are computed from all 180 scenario–replication outcomes. For inferential analysis, however, the six scenario-specific outcomes sharing the same replication index are averaged to form one balanced macro-replication. Consequently, standard errors, confidence intervals, paired reward gaps, and paired hypothesis tests are based on 30 macro-replications rather than on the 180 scenario–replication outcomes treated as mutually independent observations. This aggregation assigns equal inferential weight to each prespecified scenario and avoids artificial inflation of the effective sample size.

All policies are evaluated under the same full baseline reward specification. Each ablation policy is first optimized after removing the indicated objective component and is then evaluated under the original full reward function. The best-static comparator is selected before evaluation using a disjoint calibration sample. The calibration procedure selects the always-fast policy; therefore, the ex-ante best-static row represents the fixed-module benchmark selected independently of the evaluation sample.

The full adaptive policy achieves the highest mean reward among all evaluated policies. Relative to the greedy policy, its mean reward advantage is 0.000452, with a paired 95% confidence interval of [0.000439, 0.000465]. Although this difference is small in absolute magnitude, it is positive in all 30 balanced macro-replications and remains statistically significant after Holm adjustment. The result should therefore be interpreted as a small but highly consistent dynamic advantage rather than as a large aggregate performance improvement.

The ablation results clarify the contribution of the individual objective components. Removing switching friction increases switching frequency from 0.108117 to 0.125584 and reduces mean reward by 0.002268. Removing the instability penalty produces a smaller reward reduction of 0.000701, together with modest increases in latency and instability. Removing latency disutility causes a severe deterioration in reward because the resulting policy strongly favors the high-quality but high-latency advanced AI module. Thus, latency disutility acts as the principal safeguard against operationally costly delay, whereas switching friction and the instability penalty provide smaller but systematic refinements to the state-dependent orchestration policy.

In Table 9, gap from full adaptive denotes the mean paired reward difference between the full adaptive policy and the indicated policy. Gap CI-L and Gap CI-U report the lower and upper bounds of the corresponding 95% paired confidence interval. Holm-adjusted p_t denotes the Holm-adjusted p-value from the two-sided paired t-test conducted across the 30 balanced macro-replications.

6.5. Integrated interpretation and experimental summary

The computational results provide four main insights. First, no single module is uniformly best across operating regimes. Static reliance on the advanced AI module can perform poorly when latency disutility and instability penalties are large, even though the module has high nominal quality. Conversely, the fast rule-based module becomes attractive in stressed regimes but may sacrifice decision quality when the system is stable. This confirms that the most accurate module need not be the most valuable module from an operational perspective [10, 13, 58].

Second, switching friction creates stability-aware behavior. As the cost of transition increases, the adaptive policy switches less frequently and retains the current module over a larger set of states. This provides computational evidence for the hysteresis mechanism derived in the structural analysis. The policy does not switch merely because another module has a slightly higher one-period reward; it switches only when the continuation-adjusted value of doing so is sufficiently large [20, 23, 24].

Third, latency disutility changes the composition of module usage. As the latency penalty increases, the adaptive policy shifts toward the fast rule-based module. This behavior is consistent with the latency–quality tradeoff embedded in the reward function. High-complexity analytical modules are valuable when time pressure is moderate, but lower-latency modules may dominate under urgent or degraded operating conditions [13, 38, 58].

Fourth, the stress-test results show that adaptive switching should be understood as a robustness-oriented orchestration principle rather than as a claim of uniform dominance over every restricted policy. Greedy, best-static, and always-fast benchmarks can be competitive in latency-dominant regions. However, static reliance on the advanced AI module is operationally fragile across the stress-test battery. The finding supports the broader claim that AI-enabled decision technologies should be deployed through dynamic, stability-aware orchestration rather than static model selection [12, 15, 39].

Table 9. Comprehensive policy comparison and objective-ablation analysis under a common baseline evaluation criterion.

	Overall performance				Operational decomposition						Paired comparison with full adaptive			
Policy or variant	Reward	SE	CI-L	CI-U	Quality	Latency	Latency penalty	Switching frequency	Switching cost	Instability	Gap from full adaptive	Gap CI-L	Gap CI-U	Holm-adjusted p_i
Full adaptive	6.396460	0.001514	6.393364	6.399557	0.599529	0.289661	0.207489	0.108117	0.001631	0.119036	0.000000	–	–	–
Greedy	6.396009	0.001515	6.392911	6.399106	0.599204	0.288909	0.207188	0.109718	0.001637	0.118987	0.000452	0.000439	0.000465	2.57×10^{-34}
Ex-ante best static (always fast selected)	6.297205	0.001302	6.294542	6.299869	0.567967	0.232626	0.184675	0.002998	0.000051	0.109531	0.099255	0.098744	0.099766	5.94×10^{-55}
No switch	0.836752	0.016373	0.803265	0.870239	0.689771	0.786400	0.643635	0.000000	0.000000	0.197849	5.559709	5.526707	5.592710	2.47×10^{-53}
Always legacy	4.242915	0.002722	4.237349	4.248481	0.667574	0.509872	0.409990	0.003006	0.000042	0.159924	2.153546	2.151018	2.156074	1.67×10^{-73}
Always ML	0.342053	0.004904	0.332024	0.352083	0.736985	0.882258	0.719911	0.002999	0.000042	0.220513	6.054407	6.047391	6.061423	1.32×10^{-73}
Always AI	-7.566655	0.008947	-7.584954	-7.548356	0.786200	1.518633	1.258455	0.002997	0.000051	0.301102	13.963116	13.947812	13.978420	3.27×10^{-74}
Ablation: no switching penalty	6.394193	0.001516	6.391093	6.397293	0.601301	0.293597	0.209064	0.125584	0.001859	0.119841	0.002268	0.002240	0.002296	2.26×10^{-44}
Ablation: no instability penalty	6.395759	0.001514	6.392662	6.398856	0.600668	0.292225	0.208515	0.115195	0.001722	0.119551	0.000701	0.000687	0.000715	2.69×10^{-38}
Ablation: no latency disutility	-7.422233	0.009100	-7.440845	-7.403620	0.786188	1.517752	1.257721	0.003194	0.000053	0.300974	13.818693	13.803040	13.834347	7.67×10^{-74}

Summary of computational evidence. In this section, we report computational experiments evaluating the adaptive switching optimization framework [20, 32, 33]. The results show that the model's structural mechanisms are reflected in simulated policy behavior: Switching friction reduces switching frequency, latency disutility shifts module usage toward faster modules, and static reliance on a high-complexity AI module can be operationally inferior under stressed conditions. The full-page mechanism diagnostics, regime stress-test frontier, scenario-specific module orchestration analysis, regime-level loss decomposition, and comprehensive ablation table jointly support the central claim that deployment of heterogeneous decision technologies should be treated as a stability-aware dynamic orchestration problem rather than as a one-time model-selection problem [7, 12].

7. Managerial and operational insights

The results of the analysis have several implications for organizations that operate portfolios of heterogeneous decision modules in time-sensitive environments [7, 12, 16]. The central point is that module deployment is not well-represented as a one-time model-selection decision. When operating conditions change over time, the preferred module may also change, and the cost of moving between modules becomes part of the decision problem [20, 32, 33]. The structural results developed in this paper clarify how regime dependence, latency sensitivity, switching friction, and continuation value jointly shape module deployment [24].

7.1. State-dependent module value and latency-sensitive deployment

A decision module should not be regarded as universally superior unless it satisfies the dominance condition in Proposition 4.2 [13, 38, 58]. In most applications, module value depends on the operating state. A high-quality AI module may be appropriate in stable conditions with moderate delay sensitivity, whereas a faster rule-based or lightweight analytical module may be preferable when urgency, congestion, or workload is high. Modules with more balanced quality–latency profiles may be preferred in intermediate conditions [10, 15, 17]. This state dependence limits the usefulness of rankings based solely on average accuracy, average latency, or stand-alone benchmark performance. The deployed value of a module also depends on the current regime, workload or urgency level, active module, transition cost, instability exposure, and expected future conditions. The relevant question is therefore not which module is best on average, but which module should be used under a given operational state [12].

The latency threshold result gives a more specific deployment rule [13, 38, 58]. A slower, higher-quality module may be preferred when the cost of delay is limited. As workload, congestion, or urgency increases, however, the latency-disutility term can exceed the quality advantage of that module [49–51]. Under Assumption 4.8, the preferred module changes at a pairwise threshold in the latency-disutility state [48, 53, 54]. This tradeoff is relevant in command-and-coordination systems, emergency operations, cyber defense, transportation control, and infrastructure management, where a delayed recommendation may have less operational value even when it is analytically more accurate. Module evaluation should therefore incorporate the operational cost of delay rather than rely on analytical quality alone [10].

7.2. Switching friction, hysteresis, and greedy control

Switching friction affects both the frequency and timing of module changes [20–22]. Positive transition costs create continuation regions in which retaining the current module remains optimal even when another module has a modest one-period advantage. This effect is important in systems where frequent reconfiguration can reduce reliability, interrupt workflows, or increase supervisory burden [11, 23, 26]. The sources of switching friction may include data synchronization, recalibration, interface modification, supervisory approval, model warm-up, validation, and temporary disruption [1–3]. These costs may be difficult to observe directly, but they still affect the value of changing modules. The hysteresis result formalizes the need for a nontrivial improvement before a switch is justified [4, 6].

This also explains why purely greedy switching can be misleading. A greedy policy compares current-period rewards but does not account for the next active module, future regime transitions, repeated switching, or transition-induced instability. A switch that appears attractive in the current period may therefore reduce discounted long-run value [20, 32, 33]. The computational results show that Greedy remains close to the adaptive policy in many states. This is consistent with environments in which continuation effects are limited and current-period rewards already capture most of the relevant variation. The adaptive policy becomes more useful when regime persistence, switching friction, and continuation-value differences are large enough to affect the timing of module changes. In practice, the choice between Greedy and full dynamic optimization should reflect both the expected value gain and the cost of implementing a more-complex policy.

7.3. Orchestration in mission-critical systems

Mission-critical systems should not deploy advanced analytical modules solely because they offer higher nominal quality [10, 17, 39]. Under severe latency exposure, unstable communications, frequent regime changes, or high transition costs, a simpler module may provide greater operational value than a slower and more complex alternative. This does not imply that advanced AI modules should be excluded [15, 16, 47]. Their use should instead be conditioned on the operating state. An orchestration layer can account for the current regime, workload, active module, switching cost, instability exposure, and expected future state before changing the deployed module. The objective is to select the module that provides the highest operational value under the current and expected conditions, rather than the module with the highest stand-alone capability [12, 18, 19].

The computational results illustrate this distinction [48–50]. Under stable-dominant and recovery conditions, the adaptive policy allocates a nontrivial share of decisions to the ML predictor. Under volatility bursts, sustained stress, and high-latency crisis conditions, module use shifts strongly toward the fast rule-based alternative [51, 53, 54]. The AdvancedAI module is not selected under the calibrated reward structure because its additional nominal quality does not offset its latency and instability penalties. These patterns are relevant beyond the specific simulation setting. In command-and-control systems [52], as well as cyber defense, emergency response, transportation, healthcare operations, and critical infrastructure, module usefulness depends on whether a recommendation can be produced, interpreted, and acted upon within the available time [11].

The policy can also be presented in an interpretable form. State-action lookup tables, regime-specific module shares, pairwise switching thresholds, continuation regions, and decompositions of immediate reward and continuation value provide practical explanations for why a module is retained

or replaced. These representations are particularly important when operators must review or justify module changes.

7.4. Interpretation of the computational evidence

The computational study evaluates whether the mechanisms established in the structural analysis are visible under stochastic regime evolution [20, 32, 33]. The experiments examine whether higher latency sensitivity increases the use of low-latency modules, whether larger switching costs reduce switching frequency, whether module use varies across regimes, and whether greedy or static policies lose value when continuation and transition effects are important. The numerical results should be read as evidence about these mechanisms rather than as a claim that the adaptive policy must produce a large gain in every environment. The adaptive advantage over Greedy is small in the baseline experiments, although it is positive and statistically reliable under paired common-path comparisons. This indicates that Greedy is nearly optimal over a substantial part of the tested state space. The additional value of dynamic optimization arises mainly in states where current-period rewards do not fully capture future switching and regime effects [58].

The stress tests provide a consistent ordering of the benchmark policies. Adaptive and Greedy remain close in average performance, while the ex-ante best-static policy incurs a small loss. The no-switch benchmark performs considerably worse because it cannot respond to changes in operating conditions. The always-AI benchmark performs worst because its quality advantage is outweighed by latency and instability penalties in the tested scenarios. The sensitivity and ablation results support the same interpretation. Perturbing the switching-cost matrix changes continuation behavior in the expected direction. Nonlinear latency penalties preserve the main latency-driven switching pattern. Removing latency, switching, or instability terms alters policy behavior in ways that are consistent with the reward structure. These results show that the principal mechanisms are not tied to a single baseline parameterization.

The practical value of the full adaptive policy therefore depends on the operating environment. A simpler greedy or threshold rule may be sufficient when transition costs, regime persistence, and continuation effects are weak. A dynamic policy is more useful when those features materially affect module choice. This distinction provides a practical basis for deciding whether the additional computational and implementation burden of dynamic programming is warranted.

8. Conclusion

This paper developed a stochastic dynamic optimization framework for the adaptive orchestration of heterogeneous decision modules under operational uncertainty. By including the currently active module in the system state and treating module selection as a sequential control decision, the model captures state-dependent decision quality, response latency, switching friction, operational instability, and stochastic regime evolution. The structural analysis shows that static module selection is generally insufficient unless a single module satisfies a strong Bellman dominance condition over the entire state space. Under a latency single-crossing condition, pairwise preferences may change at workload-dependent latency-disutility thresholds, while positive transition friction creates local continuation regions and pairwise hysteresis. The computational results support these mechanisms. Under common-path paired comparisons, the full adaptive policy provides a small but statistically reliable long-run advantage over

Greedy, although the two policies agree over most frequently visited states. The ex-ante best-static policy selects the fast rule-based module in all six stress-test scenarios and remains competitive when latency exposure dominates, whereas the no-switch and advanced-AI-only benchmarks perform substantially worse. Scenario- and regime-level results further show that the adaptive policy uses a more diverse module mix under stable and recovery conditions but relies primarily on the low-latency module under volatility, sustained stress, and high-latency crises. These findings remain stable under switching-cost matrix perturbations and nonlinear latency penalties.

The results indicate that the value of dynamic orchestration depends on the strength of regime persistence, latency sensitivity, switching friction, and continuation effects. A simple greedy or threshold policy may be adequate when these effects are weak, whereas full dynamic optimization is more useful when current module choices materially affect future operating value or when poorly timed switches create significant transition burdens. The framework therefore provides both an optimal switching policy and a practical basis for determining when its additional implementation complexity is justified. Future research may extend the model to partially observed environments, history-dependent switching costs, online estimation of transition and switching parameters, robust or distributionally robust policies, and multi-level architectures in which human supervisors and heterogeneous analytical modules jointly determine operational actions. The main conclusion is that the module with the highest nominal analytical quality need not provide the greatest operational value; effective deployment requires a state-dependent policy that balances quality, latency, transition friction, instability, and future operating conditions.

Author Contributions

Conceptualization and overall research direction, H.K.S., C.L., J.Cha, M.K., and J.Choi; original manuscript planning, system-level problem framing, defense systems perspective, and initial draft development, H.K.S.; stochastic dynamic optimization model design, mathematical formulation, Bellman modeling, structural analysis, and proof development, J.Cha and J.Choi; verification and refinement of the mathematical formulation, H.K.S., C.L., J.Cha, and J.Choi; operational scenario interpretation, army simulation context, command-and-control relevance assessment, mission-critical decision-module interpretation, and domain expert validation, C.L. and M.K.; computational experiment design, simulation structure, benchmark policy design, robustness analysis, and results interpretation, H.K.S., C.L., and J.Cha; industrial engineering perspective, operational modeling interpretation, decision-analysis refinement, and managerial implications, C.L.; literature positioning and research framework refinement, H.K.S. and J.Cha; writing—original draft preparation, H.K.S. and J.Cha; writing—review and editing, H.K.S., C.L., J.Cha, M.K., and J.Choi; supervision, mathematical validation, overall technical coordination, and corresponding author responsibilities, J.Choi. All authors have read and agreed to the submitted version of the manuscript.

Use of Generative-AI tools declaration

The authors declare that generative-AI tools were used only to assist with language editing, formatting refinement, and drafting support during manuscript preparation. All modeling design, computational analysis, mathematical derivations, interpretation of results, and final scientific

judgments were performed and verified by the authors. The authors take full responsibility for the content of this article.

Acknowledgments

The authors would like to thank their affiliated institutions for supporting the research environment in which this study was conducted.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. D. L. Parnas, On the criteria to be used in decomposing systems into modules, *Commun. ACM*, **15** (1972), 1053–1058. https://doi.org/10.1145/361598.361623
2. M. W. Maier, Architecting principles for systems-of-systems, *Syst. Eng.*, **1** (1998), 267–284. https://doi.org/10.1002/(SICI)1520-6858(1998)1:4¡267::AID-SYS3¿3.0.CO;2-D
3. T. Alabama, *Office of the deputy director for engineering*, Systems Engineering Guidebook, 2021.
4. M. Jamshidi, *Systems of systems engineering: Principles and applications*, CRC Press, 2008. https://doi.org/10.1201/9781420065893
5. P. Acheson, C. H. Dagli, Modeling resilience in system of systems architecture, *Procedia Comput. Sci.*, **95** (2016), 111–118. https://doi.org/10.1016/j.procs.2016.09.300
6. A. M. Madni, D. Erwin, M. Sievers, Constructing models for systems resilience: Challenges, concepts, and formal methods, *Systems*, **8** (2020), 3. https://doi.org/10.3390/systems8010003
7. B. Green, Y. Chen, The principles and limits of algorithm-in-the-loop decision making, *Proc. ACM Hum. Comput. Interact.*, **3** (2019), 1–24. https://doi.org/10.1145/3359152
8. R. Parasuraman, T. B. Sheridan, C. D. Wickens, A model for types and levels of human interaction with automation, *IEEE Trans. Syst. Man Cybern. A*, **30** (2000), 286–297. https://doi.org/10.1109/3468.844354
9. M. R. Endsley, From here to autonomy: Lessons learned from human–automation research, *Hum. Factors*, **59** (2017), 5–27. https://doi.org/10.1177/0018720816681350
10. Y. Wang, S. H. Chung, Artificial intelligence in safety-critical systems: A systematic review, *Ind. Manag. Data Syst.*, **122** (2022), 442–470. https://doi.org/10.1108/IMDS-07-2021-0419
11. D. D. Woods, Four concepts for resilience and the implications for the future of resilience engineering, *Reliab. Eng. Syst. Saf.*, **141** (2015), 5–9. https://doi.org/10.1016/j.ress.2015.03.018
12. E. Miedema, S. Waschull, C. Emmanouilidis, Towards trustworthy artificial intelligence for decision-making: A lifecycle perspective on knowledge- and data-driven artificial intelligence systems, *Comput. Ind.*, **174** (2026), 104409. https://doi.org/10.1016/j.compind.2025.104409
13. Z. Bai, J. Li, H. Shen, A robust two-stage edge cloud resource allocation under edge–edge collaboration, *Eur. J. Oper. Res.* 2026. https://doi.org/10.1016/j.ejor.2026.04.043

14. S. Sajwan, B. K. Panigrahi, A. K. Srivastava, Multi-stage operational resilience metric-driven optimal service restoration in DER-rich power distribution systems, *IEEE Access*, **13** (2025), 95275–95287. https://doi.org/10.1109/ACCESS.2025.3574480

15. R. Dobbe, T. K. Gilbert, Y. Mintz, Hard choices in artificial intelligence, *Artif. Intell.*, **300** (2021), 103555. https://doi.org/10.1016/j.artint.2021.103555

16. B. Shneiderman, Human-centered artificial intelligence: Reliable, safe & trustworthy, *Int. J. Hum. Comput. Interact.*, **36** (2020), 495–504. https://doi.org/10.1080/10447318.2020.1741118

17. B. Gyevnár, A. Kasirzadeh, AI safety for everyone, *Nat. Mach. Intell.*, **7** (2025), 531–542. https://doi.org/10.1038/s42256-025-01020-y

18. J. Zhang, Z. Shao, L. Zhang, J. Benitez, Exploring users' post-adoption use of generative AI: An attitudinal ambivalence perspective, *Decis. Support Syst.*, **197** (2025), 114521. https://doi.org/10.1016/j.dss.2025.114521

19. M. An, J. Lin, J. Benitez, Effects of artificial intelligence usage and knowledge-based dynamic capabilities on organizational innovation: A configurational approach, *Decis. Support Syst.*, **200** (2026), 114573. https://doi.org/10.1016/j.dss.2025.114573

20. M. Yu. Kitaev, R. F. Serfozo, M/M/1 Queues with switching costs and hysteretic optimal control, *Oper. Res.*, **47** (1999), 310–318. https://doi.org/10.1287/opre.47.2.310

21. R. Batta, O. Berman, Q. Wang, Balancing staffing and switching costs in a service center with flexible servers, *Eur. J. Oper. Res.*, **177** (2007), 924–938. https://doi.org/10.1016/j.ejor.2006.01.008

22. M. E. Mayorga, K. M. Taaffe, R. Arumugam, Allocating flexible servers in serial systems with switching costs, *Ann. Oper. Res.*, **172** (2009), 231–242. https://doi.org/10.1007/s10479-009-0575-7

23. Y. Han, Z. Wang, Optimal switching policy for batch servers, *Oper. Res. Lett.*, **51** (2023), 560–567. https://doi.org/10.1016/j.orl.2023.09.004

24. A. Farrokh, V. Krishnamurthy, R. Schober, Optimal adaptive modulation and coding with switching costs, *IEEE Trans. Commun.*, **57** (2009), 697–706. https://doi.org/10.1109/TCOMM.2009.03.070115

25. Y. Dimitrakopoulos, A. Burnetas, The value of service rate flexibility in an M/M/1 1ueue with admission control, *IISE Trans.*, **49** (2017), 603–621. https://doi.org/10.48550/arXiv.1205.4315

26. E. Furman, A. Diamant, M. Kristal, Customer acquisition and retention: A fluid approach for staffing, *Prod. Oper. Manag.*, **30** (2021), 4236–4257. https://doi.org/10.1111/poms.13520

27. F. Pasqualetti, F. Dörfler, F. Bullo, Attack detection and identification in cyber-physical systems, *IEEE Trans. Autom. Control*, **58** (2013), 2715–2729. https://doi.org/10.1109/TAC.2013.2266831

28. N. Mollá, C. Heavin, A. Rabasa, Data-driven decision making: New opportunities for DSS in data stream contexts, *J. Decis. Syst.*, **31** (2022), 1–15. https://doi.org/10.1080/12460125.2022.2071404

29. G. Rinaldi, K. Theodorakos, F. C. Garcia, O. M. Agudelo, B. D. Moor, DSS4EX: A decision support system framework to explore artificial intelligence pipelines with an application in time series forecasting, *Expert Syst. Appl.*, **269** (2025), 126421. https://doi.org/10.1016/j.eswa.2025.126421

30. E. Mahmoodi, M. Fathi, Data-driven simulation-based decision support system for resource allocation in Industry 4.0 and smart manufacturing, *J. Manuf. Syst.*, **72** (2024), 287–307. https://doi.org/10.1016/j.jmsy.2023.11.019

31. G. Musick, T. A. O'Neill, B. G. Schelble, N. J. McNeese, J. B. Henke, What happens when humans believe their teammate is an AI? An investigation into humans teaming with autonomy, *Comput. Hum. Behav.*, **122** (2021), 106852. https://doi.org/10.1016/j.chb.2021.106852
32. M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*, John Wiley & Sons, New York, 1994. https://doi.org/10.1002/9780470316887
33. D. P. Bertsekas, *Dynamic programming and optimal control*, Athena Scientific, 2020.
34. B. Legros, Late-rejection, a strategy to perform an overflow policy, *Eur. J. Oper. Res.*, **281** (2020), 66–76. https://doi.org/10.1016/j.ejor.2019.08.037
35. X. Zhang, W. C. Cheung, Online allocation of reusable resources in nonstationary environments, *Math. Oper. Res.* 2025, 1–31. https://doi.org/10.1287/moor.2023.0250
36. N. Zhang, X. Li, Z. Xu, An evolutionary reinforcement learning framework for joint work package sizing and scheduling with uncertainties, *Eur. J. Oper. Res.*, **332** (2026), 391–409. https://doi.org/10.1016/j.ejor.2026.01.023
37. M. G. Juárez, A. Giret, V. Botti, Semantic and modular orchestration of AI-driven digital twins for industrial interoperability and optimization, *J. Ind. Inf. Integr.*, **48** (2025), 100959. https://doi.org/10.1016/j.jii.2025.100959
38. W. Su, M. Gao, X. Gao, X. Zhu, D. Fang, Adaptive decision-making with deep Q-network for heterogeneous unmanned aerial vehicle swarms in dynamic environments, *Comput. Electr. Eng.*, **119** (2024), 109621. https://doi.org/10.1016/j.compeleceng.2024.109621
39. E. Tabassi, *Artificial intelligence risk management framework (AI RMF 1.0),* NIST Press, 2023. https://doi.org/10.6028/NIST.AI.100-1
40. M. Zinkevich, *Online convex programming and generalized infinitesimal gradient ascent,* AAAI Press, 2003, 928–935. Available from: https://people.eecs.berkeley.edu/ brecht/cs294docs/week1/03.Zinkevich.pdf.
41. E. Hazan, Introduction to online convex optimization, *Found. Trends Optim.*, **2** (2016), 157–325. https://doi.org/10.1561/2400000013
42. S. Shalev-Shwartz, Online learning and online convex optimization, *Found. Trends Mach. Learn.*, **4** (2012), 107–194. https://doi.org/10.1561/2200000018
43. A. Steland, Online detection of changes in moment-based projections: When to retrain deep learners or update portfolios? *J. Mach. Learn. Res.*, **27** (2026), 1–50. Available from: https://jmlr.org/papers/v27/23-0274.html.
44. P. Zhao, Y. J. Zhang, L. Zhang, Z. H. Zhou, Adaptivity and non-stationarity: Problem-dependent dynamic regret for online convex optimization, *J. Mach. Learn. Res.*, **25** (2024), 1–52. Available from: https://jmlr.org/papers/v25/21-0748.html.
45. O. D. Ovuakporie, K. G. Pillai, C. Wang, Y. Wei, Differential moderating effects of strategic and operational reconfiguration on the relationship between open innovation practices and innovation performance, *Res. Policy*, **50** (2021), 104146. https://doi.org/10.1016/j.respol.2020.104146
46. S. Y. Wu, G. Chen, M. J. Shi, J. Alonso-Mora, Decentralized multi-agent trajectory planning in dynamic environments with spatiotemporal occupancy grid maps, *IEEE Int. Conf. Robot. Autom.*, 2024, 7208–7214. https://doi.org/10.1109/ICRA57147.2024.10610670

47. J. D. Schierman, M. D. DeVore, N. D. Richards, M. A. Clark, Runtime assurance for autonomous aerospace systems, *J. Guid. Control Dyn.*, **43** (2020), 2205–2217. https://doi.org/10.2514/1.G004862
48. B. Clark, D. Patt, H. Schramm, Mosaic warfare: Exploiting artificial intelligence and autonomous systems to implement decision-centric operations, 2020. Available from: https://csbaonline.org/resource/mosaic-warfare-exploiting-artificial-intelligence-and-autonomous-systems-to-implement-decision-centric-operations/.
49. Department of defense, summary of the joint all-domain command and control (JADC2) strategy, 2022. Available from: https://ebs.publicnow.com/view/CF7575BE41231B4BBFE5042F0A08BCD7479E1F89.
50. Department of defense, *Hicks announces delivery of initial CJADC2 capability*, DOD News, 2024.
51. Department of defense, FY 2024 annual performance report for the DoD strategic management plan for FYs 2022–2026 and FY 2025 annual performance plan, 2025. Available from: https://insidedefense.com/document/dods-fy-24-annual-performance-report-strategic-management-plan.
52. H. K. Shin, J. Cha, H. S. Kim, Y. J. Park, A study on the technical characteristics of a reference model for the korean command and control system (KCCS) considering the future warfare environment, *J. Korea Acad. Ind. Coop. Soc.*, **27** (2026), 205–215. https://doi.org/10.5762/KAIS.2026.27.5.205
53. B. Clark, D. Patt, T. A. Walton, *Implementing decision-centric warfare: Elevating command and control to gain an optionality advantage*, Hudson Institute, Washington DC, 2021.
54. M. C. Horowitz, Artificial intelligence, international competition, and the balance of power, *Tex. Natl. Secur. Rev.*, **1** (2018), 37–57. https://doi.org/10.15781/T2639KP49
55. M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, U. Topcu, Safe reinforcement learning via shielding, *Proc. AAAI Conf. Artif. Intell.*, 2018, 2669–2676. https://doi.org/10.1609/aaai.v32i1.11797
56. A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, P. Tabuada, Control barrier functions: Theory and applications, *Eur. Control Conf.*, (2019), 3420–3431. https://doi.org/10.23919/ECC.2019.8796030
57. S. D. Gu, L. Yang, Y. L. Du, G. Chen, F. Walter, J. Wang, A review of safe reinforcement learning: Methods, theories, and applications, *IEEE Trans. Pattern Anal. Mach. Intell.*, **46** (2024), 11216–11235. https://doi.org/10.1109/TPAMI.2024.3457538
58. E. J. Pinker, R. A. Shumsky, The efficiency–quality trade-Off of cross-trained workers, *Manuf. Serv. Oper. Manag.*, **2** (2000), 32–47. https://doi.org/10.1287/msom.2.1.32.23268

AIMS Press