iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VIII **Month of publication:** August 2026

DOI: https://doi.org/10.22214/ijraset.2026.84531

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VIII Aug 2026- Available at www.ijraset.com

Restoring Minimum Viable Service under Cascading Failures: Dependency-Aware Multi-Objective Recovery Orchestration for Multi-Site Hybrid Clouds

Adepegba Akindayomi Akintade
School of Computer Science and Information Systems, Osiri University, Nebraska, USA

Abstract: ***Disaster-recovery plans commonly prioritize infrastructure components, virtual machines, or recovery tasks, although a recovered component does not by itself restore a usable business service. This study formulates hybrid-cloud recovery as dependency-complete restoration of weighted business capabilities. A service is counted as available only after its required network, identity, security, data, and application dependencies are jointly recovered and validated. The problem is modelled as a resource-constrained, multi-mode recovery schedule with recovery-point limits and security gating, and it is solved with a dependency-aware multi-objective optimizer, MVS-NSGA-II, that minimizes weighted capability downtime, recovery cost, residual risk, and data-loss exposure. Against an exact reference model on small cases and against conventional priority policies on matched medium and large scenarios, the optimizer reduces weighted capability downtime substantially under moderate resource scarcity while respecting recovery-point and security constraints. The benefit is largest where dependencies are dense and resources are constrained, and it narrows when capacity is abundant or a single objective dominates. Sensitivity analysis over duration uncertainty and resource scarcity shows stable and interpretable behaviour, and planning runtime remains practical for operational use. The results indicate that recovery should be measured and optimized at the capability boundary rather than at the component boundary, and that dependency-aware, multi-objective scheduling with an explicit decision policy provides a defensible basis for recovery runbooks and orchestration platforms in multi-site hybrid clouds.***

Keywords: disaster recovery; hybrid cloud; service dependency; minimum viable service; multi-objective optimization; NSGA-II; operational resilience

I. INTRODUCTION

Modern enterprises rarely deliver a critical service through one server, one virtual machine, or one cloud resource. A usable transaction path can depend on network reachability, name resolution, identity federation, secret management, replicated storage, databases, event transport, application components, security monitoring, and data-integrity checks. These elements are often distributed across on-premises systems, public clouds, regional sites, and third-party services. A disruption can therefore leave many individual components technically restored while the business capability remains unavailable. The distinction is operationally consequential: a database that has restarted but cannot be reached through identity, network, or application dependencies does not restore customer service.

Established continuity guidance recognizes this broader missi0on context. NIST contingency-planning guidance connects recovery priorities to business impact analysis and system interdependencies, while NIST cyber-recovery guidance emphasizes resource prioritization, playbooks, exercises, and continuous improvement (Swanson et al., 2010; Bartock et al., 2016). NIST IR 8286D further treats business impact analysis as a way to identify mission-essential functions and the assets that enable them (Quinn et al., 2025). ISO 22301:2019 frames continuity around delivery of products and services at an acceptable predefined capacity, and ISO/IEC 27031:2025 aligns information and communication technology readiness with business continuity, including dependence on cloud and external services. These standards establish why recovery must be business-driven, but they do not prescribe an optimization method for sequencing interdependent recovery tasks under scarce resources.

Cloud disaster-recovery research has advanced along several related paths. Multi-cloud orchestration research addresses heterogeneous resource selection, configuration, deployment, and control (Tomarchio et al., 2020). Predictive systems such as Narya and CDR-TSP link failure signals to preventive migration, resource allocation, or recovery workflows (Levy et al., 2020;

Meng et al., 2026). Production systems such as RESIN demonstrate that narrowly scoped closed-loop mitigation can reduce operational disruption (Lou et al., 2022). Recent optimization work applies NSGA-II to virtual-machine disaster-recovery placement and jointly considers response time, resource utilization, and cost (Wang et al., 2025). Risk-aware orchestration now also addresses correlated faults and self-healing control, while hierarchical multi-cloud service-chain methods model topology, cost, latency, and stability (Roberts et al., 2026; Xie et al., 2026). However, these approaches primarily optimize hosts, virtual machines, workflows, service-chain operating quality, or selected fault classes. They do not generally count a business capability as restored only after its complete required dependency chain and assurance conditions are operational.

Service dependency is not a secondary detail. Production cloud practice shows that dependencies shape cascading impact, architectural risk, upgrades, and reactive mitigation (Yang et al., 2022). CloudCanary similarly demonstrates that correlated failure can persist despite replication when services share dependencies (Zhai et al., 2020). Conventional scheduling theory supplies useful foundations: the resource-constrained project scheduling problem (RCPSP) formalizes precedence and renewable-resource constraints, and directed-acyclic-graph scheduling provides scalable priority rules and heterogeneous execution models (Hartmann & Kolisch, 2000; Topcuoglu et al., 2002; Hartmann & Briskorn, 2022). Yet a standard makespan objective does not distinguish between restoring a low-value terminal task and completing the final prerequisite of a mission-essential capability.

This study addresses that gap by formulating recovery around dependency-complete business capabilities. A capability is a defined set of technical recovery tasks and validation conditions. Its availability time is the latest completion time among those requirements. The minimum viable service (MVS) time is the earliest time at which every designated essential capability is available. This definition separates partial infrastructure recovery from a defensible operational claim that a minimum service has returned.

The study makes five contributions. First, it presents a graph-based multi-mode recovery model that integrates technical dependencies, business capability weights, resource limits, recovery point objectives, security gates, restore-versus-failover choices, cost, and risk. Second, it introduces weighted capability downtime and dependency-complete MVS time as primary recovery outcomes. Third, it develops a capability-aware heuristic and a tailored MVS-NSGA-II search procedure that produces feasible schedules through a serial schedule-generation scheme. Fourth, it validates the search against an exact time-indexed mixed-integer reference in a small deterministic case and evaluates it through matched simulation across three architecture sizes, scarcity levels, and duration uncertainty. Fifth, it releases transparent scenario definitions, code, experiment seeds, schedules, and supplementary results so that the synthetic assumptions can be inspected and replaced with organization-specific values.

II. RELATED WORK AND RESEARCH GAP

A. Business continuity, disaster recovery, and mission impact

Business continuity and disaster recovery are related but not interchangeable. Business continuity concerns the sustained delivery of products and services at an acceptable capacity, while disaster recovery focuses on restoring supporting information systems, data, and technical operations. Layered continuity models have sought to organize continuity-of-operations, business-continuity, information-system contingency, disaster-recovery, and critical-infrastructure plans into a coherent structure (Stamenkov, 2022). This separation matters for optimization: the technical plan supplies recoverable components and procedures, whereas business impact analysis determines which capabilities are essential, how strongly downtime should be weighted, and what data loss is tolerable.

RTO and RPO remain necessary but incomplete. RTO defines a target time for restoration; RPO constrains tolerable data loss. In a highly coupled service, however, a nominal RTO attached to an individual application can be misleading when upstream identity, storage, network, or security dependencies have different targets. NIST guidance therefore emphasizes system and operational priorities rather than isolated recovery scripts (Swanson et al., 2010; Bartock et al., 2016). The model developed here represents these priorities through capability weights and essential-capability designations, while task-level staleness is compared with task-specific RPO limits.

B. Cloud recovery and proactive orchestration

Cloud platforms enable elastic capacity, geographic redundancy, infrastructure APIs, programmable workflows, and automated traffic redirection. These characteristics make recovery orchestration technically feasible, but they also enlarge the decision space. Multi-cloud frameworks must reconcile heterogeneous interfaces, policies, cost models, and resource semantics (Tomarchio et al., 2020). The 2025 edition of ISO/IEC 27031 explicitly extends ICT continuity consideration to external and cloud dependencies, reinforcing the need to model cross-site and cross-provider relationships.

Several systems reduce disruption by acting before or during failure. Narya predicts imminent Azure host failures and adaptively selects preventive mitigation, reporting reduced virtual-machine interruptions relative to a static strategy (Levy et al., 2020). RESIN combines detection, diagnosis, and automatic mitigation for memory leaks in Azure and reports a substantial reduction in low-memory virtual-machine reboots (Lou et al., 2022). Meng et al. (2026) combine time-series failure perception with a programmable recovery workflow and report large RTO reductions in selected active-standby fault scenarios. These systems establish the feasibility of automated recovery. Their unit of analysis, however, is usually a predicted host event, a bounded fault class, or a predefined workflow rather than a general capability graph with competing recovery sequences.

C. Dependency management and cascading failure

Cloud services form dependency networks through calls, authentication, storage access, network paths, configuration, messaging, and shared infrastructure. Yang et al. (2022) describe an industrial dependency-management system that supports deployment, upgrades, architectural optimization, and failure mitigation. Zhai et al. (2020) show that dependency structure can generate correlated-failure risk even when components are replicated, and that real-time structural auditing can identify risk introduced by updates. These findings imply that recovery order cannot be determined from component criticality alone: a moderately weighted shared prerequisite can unlock several high-value capabilities, while a highly critical downstream application may be unrecoverable until common foundations return.

D. Resource-constrained and multi-objective scheduling

The RCPSP schedules activities under precedence and renewable-resource constraints, traditionally minimizing project makespan. Priority-rule heuristics, genetic algorithms, and other metaheuristics have been compared extensively, with performance affected by project size, network structure, and resource scarcity (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006). Later surveys document extensions involving multiple modes, generalized precedence, uncertainty, multiple projects, and alternative objectives (Hartmann & Briskorn, 2022). Directed-acyclic-graph methods such as HEFT prioritize tasks using graph position and expected execution cost, providing a foundation for heterogeneous workflow scheduling (Topcuoglu et al., 2002).

Cloud workflow research commonly optimizes makespan, cost, resource usage, energy, reliability, or data transfer. Recent studies explicitly address execution-time uncertainty and multi-cloud trade-offs (Li et al., 2026; Wang et al., 2026). Roberts et al. (2026) combine conditional-value-at-risk penalties, correlated-fault stress scenarios, and event-triggered self-healing in cloud resource orchestration. Xie et al. (2026) use hierarchical deep reinforcement learning to optimize latency, cost, and stability for dependency-coupled multi-cloud service chains. Disaster-recovery scheduling has also been cast as a multi-objective problem: Wang et al. (2025) optimize virtual-machine recovery response, utilization, and cost using NSGA-II. These studies strengthen the case for topology-aware, multi-objective orchestration, but their endpoints remain resource allocation, workflow quality, service-chain QoS, or fault-response efficiency. The current study differs in its decision unit and outcome definition: it schedules dependency-linked recovery tasks with alternative restore and failover modes, and it measures when weighted business capabilities become dependency-complete after disruption.

Table 1. Position of the study relative to adjacent research streams

Stream	Primary recovery unit	Dependency treatment	Principal outcome	Representative basis
Continuity standards	Mission functions, acceptable capacity, RTO/RPO	Qualitative dependencies	No computational scheduler	Swanson et al. (2010); Bartock et al. (2016); ISO/IEC 27031:2025
Proactive cloud recovery	Predicted fault, host, or predefined workflow	Partial or scenario-specific	Prediction-triggered mitigation	Levy et al. (2020); Meng et al. (2026)
Virtual-cloud DR optimization	Virtual machines and physical nodes	Capacity and compatibility	Response time, utilization, cost	Wang et al. (2025)
Risk-aware and service-chain orchestration	Resources, servers, or service functions	Topology and correlated faults	Risk, latency, cost, stability, self-healing	Roberts et al. (2026); Xie et al. (2026)
Dependency management	Production service graph	Central concern	Auditing and mitigation support	Yang et al. (2022); Zhai et al. (2020)
RCPSP and DAG scheduling	Activities or workflow tasks	Precedence graph	Makespan and resource objectives	Hartmann & Kolisch (2000); Hartmann & Briskorn (2022)
Current study	Dependency-complete business capabilities	Explicit task and capability graph	Weighted downtime, MVS, cost, risk, RPO, makespan	MVS-NSGA-II plus exact reference

The resulting research question is: how should recovery tasks and alternative modes be sequenced so that dependency-complete essential service is restored early while cost, risk, resource capacity, security requirements, and data staleness remain explicit? The hypothesis is that capability-aware multi-objective scheduling will reduce weighted capability downtime and MVS time relative to conventional recovery-order rules, with the largest gains under complex dependency graphs and constrained resources.

III. PROBLEM FORMULATION

A. Recovery graph, tasks, and modes

A recovery instance is represented by a directed acyclic graph G = (V, E). Each node i in V is a recovery task, such as restoring identity federation, failing over replicated storage, validating data integrity, or activating an application service. An edge (i, j) in E means that task j cannot start before task i has completed. Cycles are rejected during scenario validation because a cyclic runbook has no feasible precedence order without an explicit cycle-breaking procedure.

Each task i has a set of available recovery modes M_i. The experiments provide two modes where feasible: local restore and remote or cloud failover. A mode m has duration d_im; renewable resource demands q_imr for recovery-team effort, bandwidth, and compute; monetary cost c_im; operational risk rho_im; and data staleness l_im. A mode may be unavailable when its site is failed. Local restoration at the failed site is therefore disabled in affected tasks, while failover remains available. This creates a multi-mode scheduling problem in which the algorithm chooses both order and mode.

Start time S_i and completion time C_i are linked to the selected mode. For each precedence edge, the successor must wait until the predecessor is complete.

C_i = S_i + sum_{m in M_i} d_im x_im (1)

S_j >= C_i for every dependency (i, j) in E (2)

Here x_im is one when mode m is selected for task i and zero otherwise, with exactly one available mode selected. At every time t, the sum of active task demands for resource r cannot exceed capacity Q_r.

sum_i sum_m q_imr y_imt <= Q_r for every resource r and time t (3)

B. Business capabilities and minimum viable service

A business capability k is defined by a required task set R_k and a weight w_k derived conceptually from business impact. Capability completion is not assigned to a single application node. It occurs only when every required task has completed, including upstream foundations and any explicit verification task.

T_k = max_{i in R_k} C_i(4)

Weighted capability downtime (WCD) is the integral-equivalent loss under a step-function assumption in which a capability produces its full weight only after activation. From disruption time zero, this reduces to the weighted sum of capability activation times.

WCD = sum_{k in K} w_k T_k (5)

Let K* be the set of capabilities designated essential by the business impact analysis. Minimum viable service is available when all essential capabilities have returned. The MVS time is therefore the latest activation time among the essential set.

T_MVS = max_{k in K*} T_k (6)

This definition is intentionally strict. It prevents an optimizer from declaring recovery after an application process starts while identity, integrity validation, network routing, or other required controls remain absent. It also separates MVS from full recovery: nonessential reporting, analytics, or notification capabilities may be restored after the essential service envelope is available.

C. Cost, risk, RPO exposure, and makespan

Total recovery cost is the sum of selected-mode costs. The synthetic operational-risk score increases with task criticality and mode-specific risk, representing factors such as hurried failover, temporary architecture, reduced redundancy, and security-sensitive execution. RPO exposure is incurred only when selected-mode staleness exceeds the task RPO. Data-criticality weights make the same staleness exceedance more consequential for sensitive data services.

Cost = sum_i sum_m c_im x_im (7)

Risk = sum_i criticality_i sum_m rho_im x_im (8)

RPO exposure = sum_i data_weight_i max(0, l_i - RPO_i) (9)

The makespan C_max is the latest task completion. It remains relevant because full technical recovery should not be delayed indefinitely after MVS activation.

The primary search therefore minimizes the vector (WCD, T_MVS, Cost, Risk + 2 x RPO exposure, C_max). The coefficient applied to RPO exposure is a transparent synthetic scaling choice, not a universal valuation. Organizations should replace it with a value derived from regulatory, contractual, or business-impact requirements.

Table 2. Core notation and operational interpretation

Symbol	Definition	Operational interpretation
G=(V,E)	Recovery-task directed acyclic graph	Defines tasks and precedence
M_i	Available modes for task i	Restore, failover, or other organization-specific modes
S_i, C_i	Start and completion time	Temporal decision/outcome
q_imr, Q_r	Mode demand and capacity for resource r	Team, bandwidth, and compute feasibility
R_k, w_k	Required task set and weight for capability k	Dependency-complete business impact
T_k	Capability activation time	Latest completion among required tasks
T_MVS	Minimum viable service time	Latest activation among essential capabilities
WCD	Weighted capability downtime	Primary service-loss objective
l_i, RPO_i	Mode staleness and recovery-point objective	Data-loss exposure
C_max	Recovery makespan	Full technical recovery time

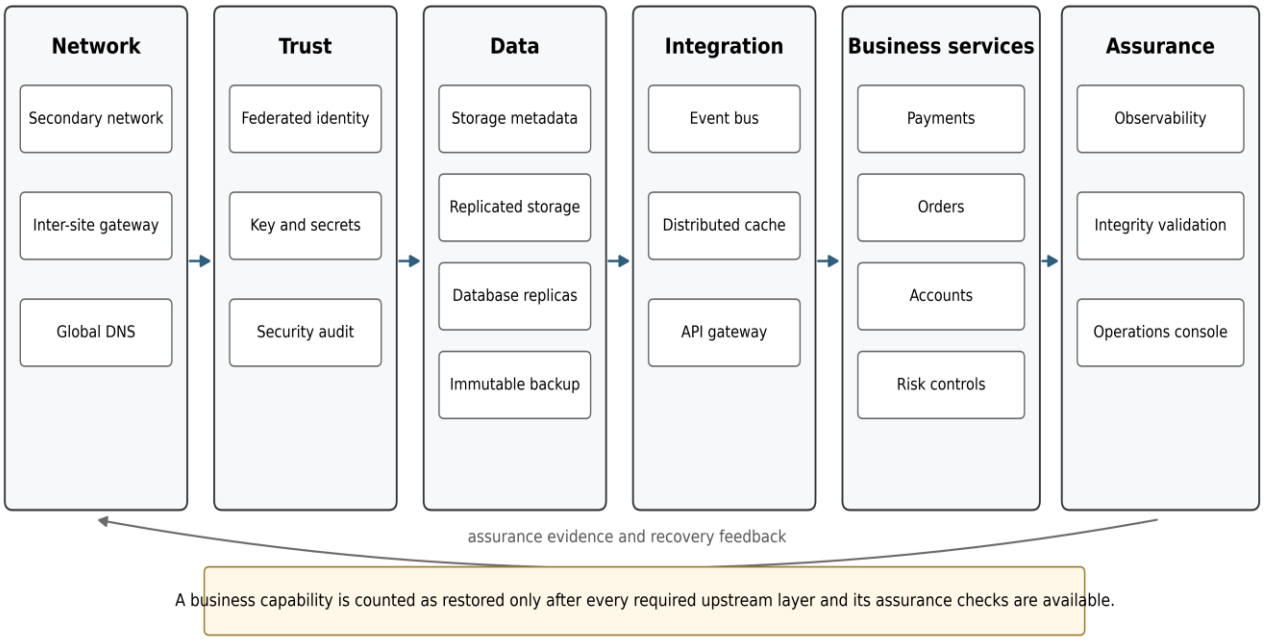


Fig. 1. Reference service layers used to construct the scenarios. The detailed task-level graphs are provided in the supplementary workbook. The figure emphasizes that business service activation depends on upstream network, trust, data, integration, and assurance layers.

IV. SOLUTION METHODS

A. Serial schedule-generation scheme

Every candidate priority and mode vector is converted into a feasible schedule through a serial schedule-generation scheme. The procedure maintains a set of unscheduled tasks, identifies tasks whose predecessors have completed, selects the ready task with the highest priority, and places it at the earliest time satisfying all renewable-resource capacities. If a proposed mode is unavailable, the decision-aware mode selector chooses an available alternative. This representation guarantees precedence and resource feasibility without adding penalty terms for infeasible schedules.

Duration uncertainty is applied before schedule generation. For each matched replication, every policy receives the same task-specific duration multipliers.

This paired design prevents an algorithm from benefiting from an easier realization. Durations are rounded to positive integer time units after perturbation.

B. Conventional recovery policies

Six comparators were implemented. FIFO uses topological order as a neutral first-ready policy. Criticality prioritizes task criticality and downstream capability influence.

Shortest recovery time (SRT) favors the shortest available mode duration. Dependency depth prioritizes tasks with long downstream chains and many descendants.

The static runbook restores layers in a conventional order from network and identity through storage, database, middleware, applications, monitoring, and analytics.

The dependency-aware capability-restoration heuristic (DACR) combines capability influence, descendant count, graph depth, task criticality, essential-capability membership, RPO urgency, security-gate status, and duration. DACR is intended as a strong interpretable comparator rather than a trivial baseline.

C. MVS-NSGA-II

NSGA-II was selected because recovery objectives conflict and a single weighted sum can conceal operationally important alternatives (Deb et al., 2002).

Each individual contains random priority keys for all tasks and an integer mode choice for each task. The serial schedule generator decodes the chromosome into a feasible schedule and returns the five objective values. The initial population includes one DACR-seeded individual so the search starts with at least one operationally plausible solution. Remaining individuals are randomly initialized over feasible modes and priority keys.

Binary tournament selection uses non-dominated rank and crowding distance. Offspring are generated by crossover of priority keys and mode choices, followed by mutation. Parent and offspring populations are combined, non-dominated fronts are calculated, and elitist survival retains complete fronts followed by the most widely spaced solutions from the final accepted front. The primary configuration used a population of 28 for 16 generations. This moderate budget was chosen to make repeated robustness experiments feasible; the exact small-case benchmark used a larger population of 60 for 40 generations.

A Pareto set alone does not determine which plan an incident commander should execute. The study therefore uses a declared service-first selection rule. From the first non-dominated front, solutions with WCD no more than 5% above the minimum WCD are retained. The selected plan then minimizes MVS time, followed by WCD, makespan, combined risk and RPO exposure, and cost. The 5% tolerance prevents a small WCD improvement from forcing a much later MVS restoration, while preserving WCD as the principal loss measure. It is a policy parameter and is exposed in the code.

D. Exact reference model

A time-indexed mixed-integer linear program was constructed for the small deterministic scenario. Binary variables assign each task to one available mode and start time. Constraints enforce single-mode selection, precedence, renewable-resource capacities, capability completion, and makespan.

The scalar reference objective gives dominant weight to WCD, with small secondary coefficients for makespan, cost, and risk. It is therefore an exact reference for that declared scalarization, not proof of global optimality across the full five-objective vector. The exact model was solved with SciPy's MILP interface and a 120-second limit; the reported instance reached a successful optimum.

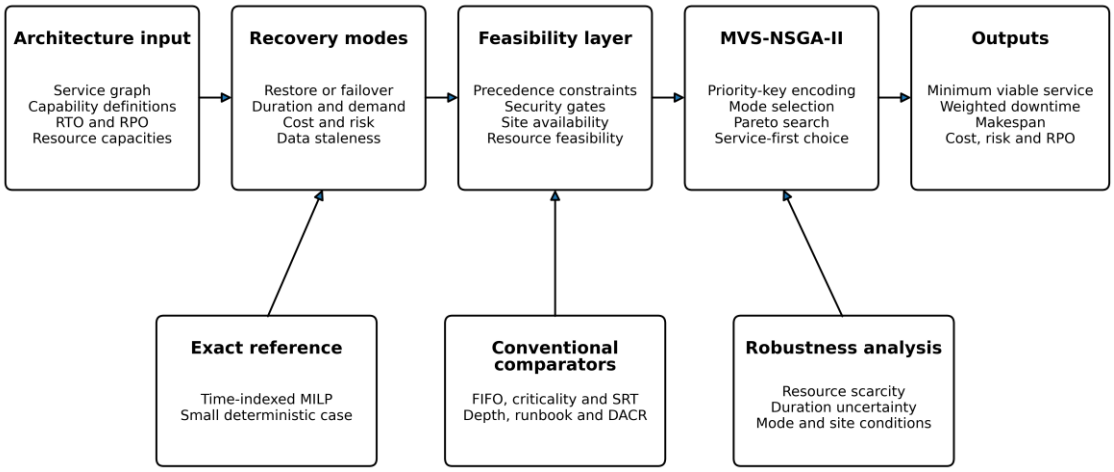


Fig. 2. Dependency-complete recovery optimization and evaluation workflow. The same architecture, mode, feasibility, and outcome definitions are used by the exact model, conventional policies, DACR, and MVS-NSGA-II.

V. EXPERIMENTAL DESIGN

A. Scenario construction

No proprietary employer or client data were used. Three transparent architecture templates were constructed to represent increasing dependency and recovery complexity. Parameters are synthetic but structurally informed by continuity guidance, cloud-architecture practice, multi-cloud orchestration literature, and production dependency research. The objective is controlled method evaluation, not estimation of recovery performance for a specific organization.

Table 3. Scenario architecture characteristics

Size	Template	Tasks	Dependencies	Capabilities	Essential	Failed site
Small	Layered enterprise	14	22	4	3	A
Medium	Multi-site transaction platform	29	60	8	6	A
Large	Hybrid cloud enterprise	44	100	8	6	ONPREM

Note. Detailed tasks, modes, capability definitions, resource demands, and parameter values are supplied in the supplementary workbook.

The small layered-enterprise scenario contains 14 tasks and four capabilities. A site-A outage requires recovery or failover of network, identity, keys, storage, database, messaging, APIs, customer access, operations, monitoring, backup, and integrity validation. The medium multi-site transaction platform contains 29 tasks, 60 dependencies, and eight capabilities, including payments, order processing, account access, risk controls, notifications, reporting, secure access, and operations control. The large hybrid-cloud enterprise contains 44 tasks and 100 dependencies spanning on-premises and cloud network, identity, security, storage, databases, streaming, messaging, multiple business domains, analytics, operations, and enterprise integrity reconciliation.

Each task has criticality, RPO, data criticality, resource demands, and one or two recovery modes. Failed-site tasks cannot use local restore. Failover is generally faster but can consume more bandwidth or compute, incur higher cost, add data staleness, and carry a higher synthetic risk score. Resource capacities are scaled to 100%, 75%, and 55% for low, moderate, and severe scarcity, with a feasibility floor ensuring that every individually available mode can still run.

B. Uncertainty and experiment sets

Task duration multipliers follow a lognormal distribution with mean approximately one. Sigma values of 0.10, 0.25, and 0.40 define low, moderate, and high uncertainty. Lognormal perturbation preserves positive durations and represents multiplicative delay from transfer rates, operator coordination, validation, retries, or environmental variability.

Four experiment sets were executed. First, 8,100 conventional-policy schedules were generated from 50 replications across three sizes, three scarcity levels, three uncertainty levels, and six policies. Second, the primary matched experiment used moderate uncertainty, 10 replications for every size-scarcity condition, six conventional policies, and MVS-NSGA-II, producing 630 schedules. Third, the sensitivity experiment used five replications for all 27 size-scarcity-uncertainty combinations and all seven policies, producing 945 schedules. Fourth, the exact deterministic small case compared the scalar MILP optimum with all policies and MVS-NSGA-II.

The primary optimizer used fixed seeds, population 28, and 16 generations. The complete seed rule, configuration, raw schedules, and scenario definitions are included in the reproducibility package. The representative Pareto and Gantt figures use the medium scenario under moderate scarcity and moderate uncertainty.

C. Outcomes and statistical analysis

Primary outcomes were WCD and MVS time. Secondary outcomes were makespan, cost, risk, and RPO exposure. Within each condition and replication, MVS-NSGA-II was compared with the conventional policy having the lowest mean WCD for that condition. This is conservative because the comparator is selected from six policies rather than fixed in advance. Paired Wilcoxon signed-rank tests evaluated WCD and MVS differences; Holm adjustment controlled multiplicity across the nine size-scarcity conditions. Rank-biserial correlation was calculated as a paired effect measure. Friedman tests compared conventional-policy rankings within each architecture size. Overall percentage-effect confidence intervals were obtained through 10,000 bootstrap resamples of the 90 matched records with a fixed seed.

The study is computational and uses no human participants, animals, personal data, or confidential operational records. Ethics approval was therefore not required.

VI. RESULTS

A. Exact small-case benchmark

Table 4. Deterministic small-scenario comparison with the exact scalarized reference

Policy	WCD	MVS	Makespan	Cost	Risk
FIFO	1233	60	60	502.4	81.7
Criticality	1181	60	60	502.4	81.7
SRT	1227	69	69	502.4	81.7
Dependency depth	1206	60	60	502.4	81.7
Static runbook	1128	60	60	502.4	81.7
DACR	1220	64	64	502.4	81.7
MVS-NSGA-II	1125	60	60	510.9	85.7
Exact MILP	1125	60	60	510.9	85.7

Note. The exact MILP optimizes a declared scalar objective dominated by WCD. MVS-NSGA-II matched the exact solution on WCD, MVS time, makespan, cost, and risk for this instance.

The exact model produced WCD 1,125, MVS time 60, and makespan 60. MVS-NSGA-II returned the same outcome and the same selected-mode cost and risk. The best conventional WCD was 1,128 from the static runbook, followed by 1,137 from dependency depth. SRT delayed MVS to 69, showing that short individual tasks do not necessarily unlock the essential capability chain quickly. The exact result provides a correctness anchor for the schedule generator and demonstrates that the evolutionary representation can recover the scalarized optimum in the small case. It does not establish exactness for medium or large instances.

B. Conventional-policy performance

Across the 27 size-scarcity-uncertainty conditions in the conventional Monte Carlo experiment, DACR had the best average rank (1.85) and the most condition wins (16). Criticality-only was second by average rank (2.30), and dependency depth was third (3.63). Static runbook had 3 condition wins and the worst average rank (4.59). The ranking confirms that business-capability influence and dependency structure improve an interpretable heuristic, while no simple rule dominates every architecture and operating condition.

Friedman tests rejected equal policy performance within every size: small chi-square = 113.54, medium chi-square = 109.91, and large chi-square = 142.67, with $p < 10^{-20}$ in each case. Resource scarcity and network complexity changed which conventional rule was strongest, supporting the use of matched best-baseline comparisons rather than a single weak benchmark.

C. Primary matched comparison

Table 5. MVS-NSGA-II versus the best conventional policy in each primary condition

Condition	Best comparator	WCD change	MVS change	Makespan change	Holm p
Small - Low	Static runbook	3.35%	1.58%	1.58%	0.010
Small - Moderate	Static runbook	-0.37%	0.49%	0.49%	0.451
Small - Severe	Shortest recovery time	2.36%	2.83%	2.83%	0.009
Medium - Low	Criticality only	3.90%	5.43%	4.67%	0.010
Medium - Moderate	Dependency depth	4.15%	6.92%	1.52%	0.009
Medium - Severe	Dependency-aware heuristic	3.37%	4.28%	4.67%	0.009
Large - Low	Dependency-aware heuristic	2.25%	5.08%	2.72%	0.009
Large - Moderate	Dependency-aware heuristic	1.95%	5.25%	2.93%	0.064
Large - Severe	Dependency-aware heuristic	4.55%	6.79%	5.01%	0.010

Note. Positive percentages are reductions (improvements). The small moderate-scarcity condition had a 0.37% WCD disadvantage despite a 0.49% MVS improvement.

Across all 90 matched records, MVS-NSGA-II reduced WCD by 2.83% (bootstrap 95% CI 2.34-3.33), MVS time by 4.29% (95% CI 3.64-4.94), and makespan by 2.94% (95% CI 2.36-3.52). WCD improved in 80 of 90 matched records, MVS time in 75, and makespan in 68. The paired Wilcoxon p-values were 7.12e-14 for WCD and 1.96e-14 for MVS.

The strongest mean primary WCD reduction occurred in the large severe-scarcity condition (4.55%), where MVS time declined by 6.79%. Under severe scarcity, WCD reductions were 2.36% for small, 3.37% for medium, and 4.55% for large architectures. 7 of nine condition-specific WCD comparisons remained significant after Holm correction. The large moderate condition had adjusted p = 0.064, while the small moderate condition was not significant.

The principal exception was the small moderate-scarcity condition, where MVS-NSGA-II increased WCD by 0.37% while reducing MVS time by 0.49%. This is not treated as noise to be hidden. It reflects the declared service-first selection rule: a Pareto plan can prioritize the completion of all essential capabilities while allowing a small increase in the weighted sum across essential and nonessential capabilities. The result also indicates that an evolutionary search is unnecessary for some small instances where a static runbook already lies near the relevant frontier.

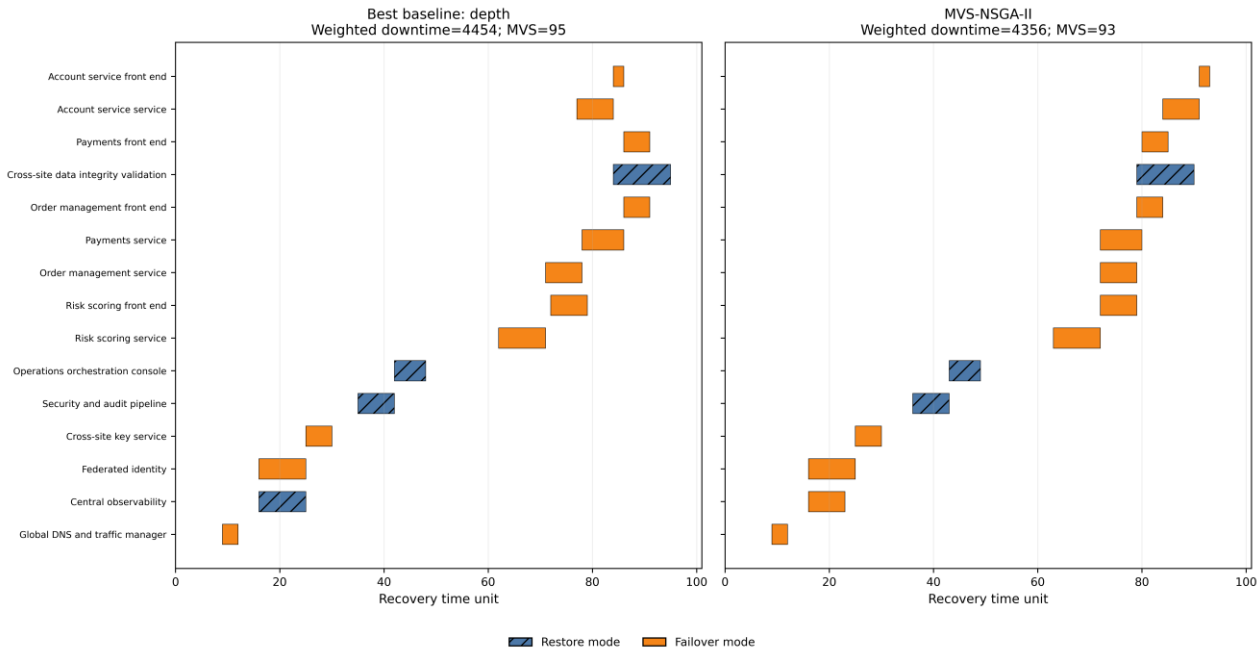


Fig. 3. Condition-specific effect of MVS-NSGA-II on weighted capability downtime in the primary matched experiment. Positive values indicate improvement relative to the best conventional policy for that condition.

D. Cost, risk, and trade-offs

Table 6. Overall paired effects with bootstrap confidence intervals

Metric	Mean effect	95% bootstrap CI	Positive cases
Weighted capability downtime reduction	2.83%	2.34 to 3.33%	80/90
Minimum viable service time reduction	4.29%	3.64 to 4.94%	75/90
Makespan reduction	2.94%	2.36 to 3.52%	68/90
Cost change	1.91%	1.68 to 2.15%	81/90
Risk change	5.95%	5.24 to 6.67%	81/90

Note. For cost and risk, positive values indicate increases. Confidence intervals are percentile intervals from 10,000 bootstrap resamples.

Service restoration was not free. Relative to each condition's best conventional WCD policy, selected MVS-NSGA-II plans increased cost by 1.91% (95% CI 1.68-2.15) and the synthetic risk score by 5.95% (95% CI 5.24-6.67). Failover modes often completed essential paths faster but used additional bandwidth or compute, cost more, or represented a less stable temporary configuration. The optimizer therefore exposes a decision that runbooks often leave implicit: whether earlier service restoration justifies higher temporary expenditure and operational risk.

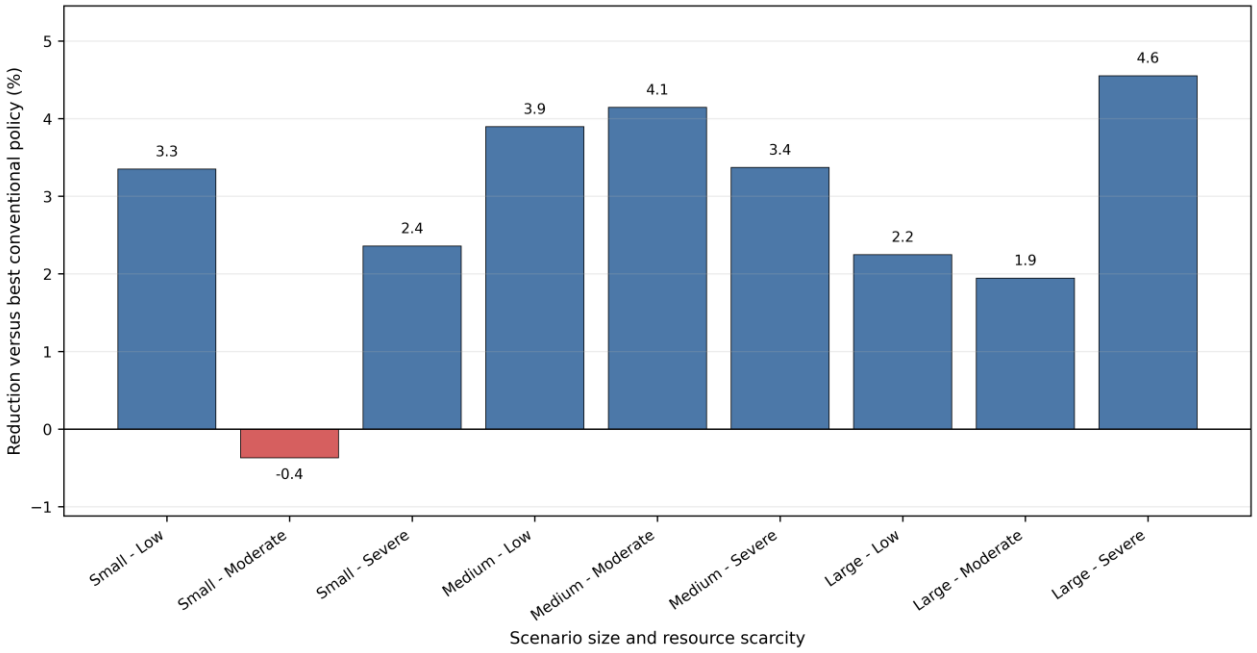


Fig. 4. Representative medium-scenario schedules under moderate scarcity and duration uncertainty. The optimized schedule advances selected capability-unlocking tasks and completes full recovery earlier than the best conventional schedule for the same duration realization.

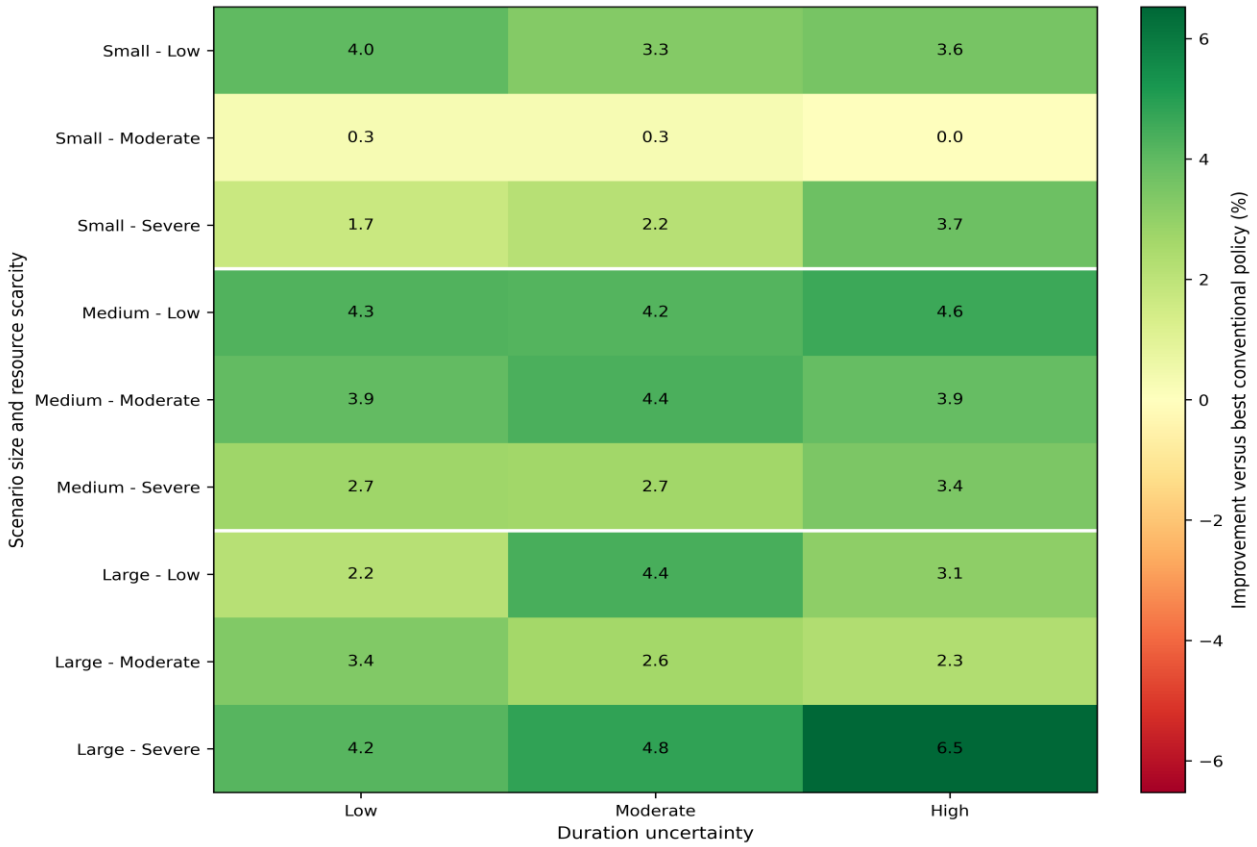


Fig. 5. Representative first-front solutions for the medium multi-site scenario. The service-first selection rule chooses a compromise after restricting attention to solutions close to the best weighted capability downtime.

E. Sensitivity to duration uncertainty and scarcity

The sensitivity experiment retained the broad pattern across low, moderate, and high duration uncertainty. Medium-scenario WCD improvements ranged from 2.67% to 4.60% across the nine scarcity-uncertainty cells. Large-scenario improvements ranged from 2.17% to 6.53%. Small low- and severe-scarcity conditions improved by 1.69-4.00%, while small moderate-scarcity gains were only 0.04-0.27%. Thus uncertainty did not eliminate the value of capability-aware search, but architecture scale and the presence of meaningful sequencing alternatives mattered more than uncertainty alone.

MVS improvements were also heterogeneous. In the large scenario they ranged from 2.54% to 11.21%; in the medium scenario, from 4.45% to 10.87%; and in the small scenario, from 0.00% to 3.72%. The absence of a monotonic relationship with sigma is expected because each cell uses different matched duration realizations and the discrete schedule can change when a critical task crosses a resource or predecessor threshold.

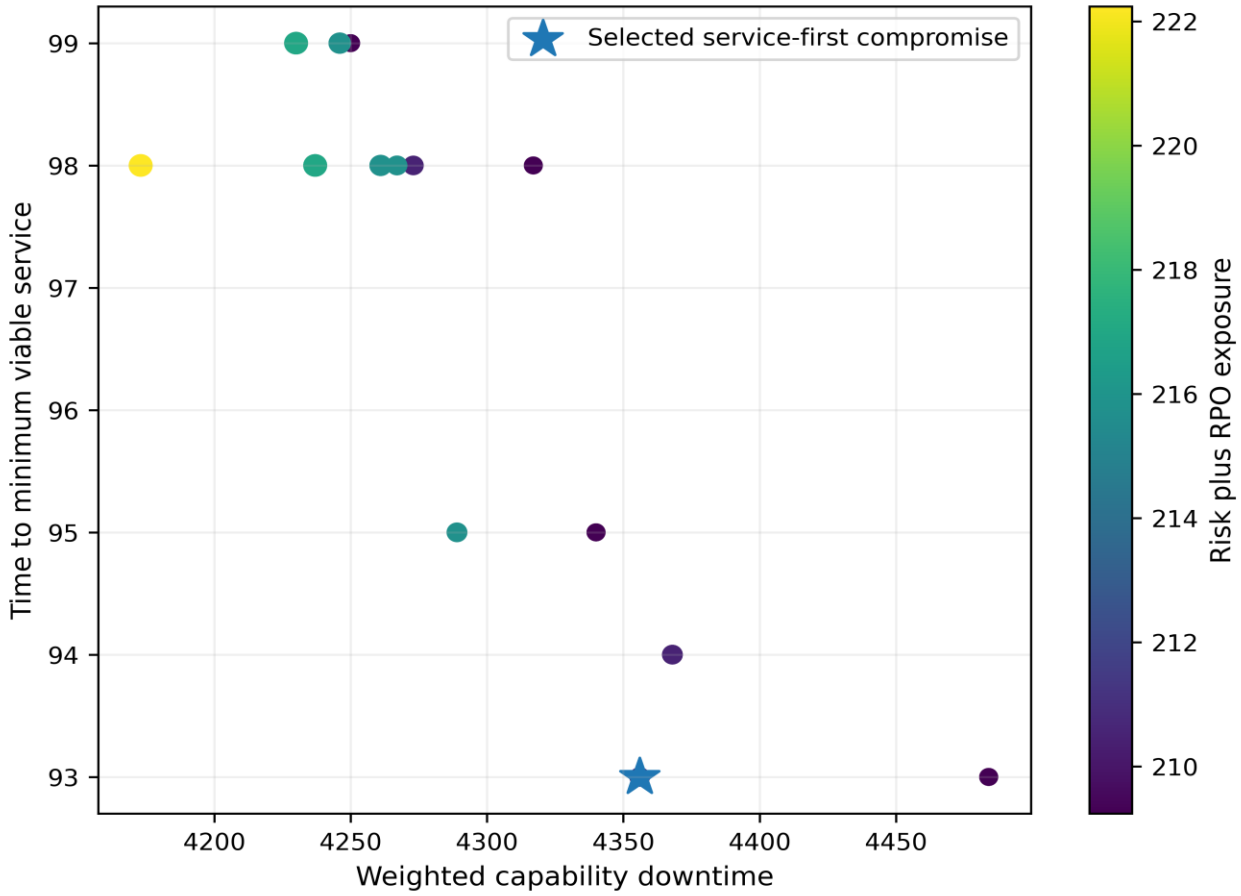


Fig. 6. Sensitivity of weighted capability downtime improvement to scenario size, resource scarcity, and duration uncertainty. Positive cells favor MVS-NSGA-II; near-zero cells indicate little advantage over the best conventional policy.

F. Computational Runtime

Table 7. Planning runtime under the primary optimizer configuration

Size	Tasks	Edges	DACR mean (s)	MVS-NSGA-II mean (s)	MVS-NSGA-II range (s)
Small	14	22	0.0010	0.442	0.421-0.457
Medium	29	60	0.0020	0.939	0.922-0.949
Large	44	100	0.0032	1.512	1.468-1.530

Note. Five timed runs per architecture at moderate scarcity and uncertainty on the execution environment used for this study. Times are implementation- and hardware-dependent and are reported to characterize computational order, not as universal latency claims.

With population 28 and 16 generations, mean MVS-NSGA-II planning time increased from 0.442 seconds for 14 tasks to 0.939 seconds for 29 tasks and 1.512 seconds for 44 tasks. The DACR heuristic required approximately 0.001-0.003 seconds. These measurements show that the tested optimizer is suitable for offline plan generation and rapid replanning at the studied scales, but the heuristic remains preferable when decisions must be produced with negligible computation or when the dependency state changes faster than an evolutionary search can complete.

VII. DISCUSSION

A. Recovery should be measured at the capability boundary

The central result is not that NSGA-II outperforms every heuristic by a large percentage. It is that the outcome definition changes the recovery problem. Component completion, virtual-machine startup, and task makespan are useful engineering metrics, but they can reward a plan that restores many isolated elements while a critical end-to-end service remains broken. WCD and MVS time require the recovery plan to demonstrate a complete path from foundational infrastructure through application and assurance conditions.

This capability boundary aligns technical scheduling with business impact analysis. The essential-capability set K* translates the question 'what must go right first?' into the optimization model, while capability weights translate the relative cost of delayed service. The approach also makes disagreement visible. Business owners can challenge weights and essential designations; architects can challenge required-task sets; recovery teams can challenge durations and resource demands. That is preferable to treating a static recovery order as self-evidently correct.

B. Dependencies create leverage and cascading delay

The performance of DACR and MVS-NSGA-II supports the proposition that shared prerequisites create recovery leverage. Identity, keys, network, databases, event transport, and assurance tasks often unlock multiple capabilities. A criticality-only rule can prioritize a high-value downstream application before its prerequisites, leaving resources occupied by work that cannot complete a service. SRT can complete many short tasks but postpone a long shared dependency. Dependency depth improves ordering but ignores business weights and recovery modes. Capability-aware scheduling combines these signals.

The finding is consistent with production evidence that dependency structure drives correlated failure and mitigation complexity (Yang et al., 2022; Zhai et al., 2020). It also extends conventional RCPSP logic. In ordinary project scheduling, all activities contribute to a common project completion; in recovery, different subsets create distinct business capabilities with different weights and minimum-service status. The same task can be valuable because it belongs directly to a capability or because it unlocks downstream tasks required by several capabilities.

C. Multi-objective recovery requires declared decision policy

A Pareto front is not an incident command decision. It presents alternatives that are non-dominated under the modeled objectives. Selection still requires policy. The service-first rule used here is transparent: remain close to the best WCD, then minimize MVS. Another organization could require a hard cost ceiling, prohibit risk above a threshold, or prioritize a regulated capability. The important requirement is to declare the rule before evaluating results.

The observed cost and risk increases demonstrate why claims of universal superiority would be misleading. Faster dependency-complete recovery can require more expensive failover resources, increased data transfer, temporary loss of redundancy, or a mode with higher operational uncertainty. A robust orchestration platform should show these consequences to the decision maker and reject plans that violate non-negotiable security or RPO constraints. Multi-objective optimization is useful because it preserves these alternatives instead of hiding them inside one arbitrary scalar score.

D. Implications for recovery runbooks and platforms

For practice, the model suggests five changes to recovery planning. First, runbooks should define business capabilities and their complete technical prerequisites, not only ordered infrastructure components. Second, recovery exercises should capture actual task durations, resource consumption, mode availability, data staleness, and validation time. Third, RTO should be evaluated at the capability boundary, while task-level targets remain supporting constraints. Fourth, security and integrity validation should be modeled as recovery tasks rather than postponed to an undefined post-recovery phase. Fifth, orchestration platforms should maintain alternative plans that can be recalculated when sites, people, links, or cloud resources are unavailable.

The model can be integrated with infrastructure-as-code, topology inventories, configuration-management databases, dependency telemetry, and business-impact records. A production implementation would require continuous validation because dependency graphs drift. It would also need human authorization, access control, immutable logging, rollback, and separation of duties. The optimization algorithm is not an autonomous authority; it is a decision engine operating within recovery governance.

E. When a heuristic may be preferable

The small moderate-scarcity result is an important boundary condition. Optimization overhead is difficult to justify when the graph is small, the runbook is already near the frontier, or the operational state is changing faster than the search can run. DACR provides an interpretable fallback with strong average conventional performance. An operational design could therefore use a tiered approach: exact optimization for small stable instances, MVS-NSGA-II for complex planning and precomputed contingencies, and DACR for rapid online replanning when compute time or information is limited.

VIII. THREATS TO VALIDITY AND LIMITATIONS

The principal limitation is external validity. The scenarios are synthetic. Their architecture patterns and parameter relationships are plausible, but they do not estimate the performance of a named organization. Absolute time, cost, risk, and staleness values should not be transferred to production. The value of the study lies in the formulation, comparative design, and reproducible test bed. Validation with real runbooks, recovery-exercise logs, dependency inventories, and incident records is required before operational deployment.

The dependency graph is static and acyclic. Real incidents can reveal unknown dependencies, degraded rather than binary states, cyclic service interactions, common-cause failures, and changing resource availability. Tasks are non-preemptive, and resources are aggregated into team, bandwidth, and compute units. Travel, approval delays, skill-specific personnel, communication contention, cloud quota, regional capacity, and supplier dependencies could be modeled more explicitly.

Risk and cost are synthetic scores. They are internally consistent but not monetized from empirical loss data. The risk objective sums task-level terms and cannot fully represent nonlinear blast radius, correlated security exposure, or organizational risk appetite. RPO exposure is also simplified to selected-mode staleness above a task threshold. Future work should incorporate scenario probabilities, distributionally robust duration models, chance constraints, risk limits, and explicit data-consistency states.

The primary optimizer uses a moderate population and generation budget. Timed runs were short at the studied scales, but runtime grows with graph size, mode count, objective evaluation, and search budget; the benchmark therefore does not establish production latency for substantially larger or continuously changing dependency graphs. More extensive hyperparameter tuning might improve the Pareto set, but tuning on the same scenarios could overfit the experimental benchmark. The exact comparison is limited to one small deterministic instance and one scalarization. It verifies feasibility and provides a useful anchor, but it does not prove optimality of the medium and large results or of the selected multi-objective compromise.

Statistical inference applies to the generated scenario realizations, not to a random sample of enterprises. The best conventional comparator was selected by mean WCD within each condition, which is intentionally demanding but introduces a data-dependent reference. Paired seeds, Holm correction, effect sizes, bootstrap intervals, and complete raw outputs reduce analytical ambiguity, but independent replication on alternative architecture generators remains necessary.

Finally, MVS is only as defensible as its capability definition. Omitting a necessary security, data, or third-party dependency will make the modeled service appear available too early. Organizations should therefore govern capability definitions through joint review by business owners, architecture, operations, security, continuity, and data governance teams.

IX. CONCLUSION

Disaster recovery is not complete when a collection of components has restarted. It is complete at a chosen service level when the dependencies and assurance conditions required for that service are functioning. This study formalized that principle through weighted capability downtime and dependency-complete minimum viable service time, embedded it in a multi-mode resource-constrained recovery model, and evaluated a capability-aware NSGA-II procedure across transparent synthetic hybrid-cloud scenarios.

MVS-NSGA-II matched the exact scalarized solution in the small deterministic case and, across 90 matched primary instances, reduced weighted capability downtime, MVS time, and makespan relative to the best conventional policy. The gains were meaningful but not universal, and they involved modest cost and larger synthetic risk increases. These trade-offs are a feature of the decision problem, not a defect to conceal.

The practical implication is direct: recovery plans should define capabilities, dependencies, alternative modes, validation tasks, and resource constraints in a machine-readable form. Optimization can then support incident commanders with feasible trade-offs, while policy determines which plan is authorized. Future validation should use real recovery exercises and production dependency data, but the released model and scenarios provide a reproducible foundation for that work.

A. Declarations

1) *Funding:* The author received no specific funding for this study.

2) *Competing interests:* The author declares no competing interests.

3) *Ethics approval:* Not applicable. The study used synthetic computational scenarios and no human participants, animals, personal data, or proprietary operational records.

4) *Data and code availability:* The complete scenario definitions, source code, fixed seeds, raw experiment results, exact-model outputs, figures, and supplementary workbook accompany the manuscript.

5) *Author contribution:* Adepegba Akindayomi Akintade: Conceptualization, domain interpretation, methodology oversight, validation, and writing - review and editing.

REFERENCES

[1] Bartock, M., Cichonski, J., Souppaya, M., Smith, M. C., Witte, G., & Scarfone, K. (2016). Guide for cybersecurity event recovery (NIST SP 800-184). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-184

[2] Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). Site reliability engineering: How Google runs production systems. O'Reilly Media.

[3] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182-197. https://doi.org/10.1109/4235.996017

[4] Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research, 297(1), 1-14. https://doi.org/10.1016/j.ejor.2021.05.004

[5] Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research, 127(2), 394-407. https://doi.org/10.1016/S0377-2217(99)00485-3

[6] International Organization for Standardization. (2019). ISO 22301:2019 Security and resilience - Business continuity management systems - Requirements. ISO.

[7] International Organization for Standardization. (2025). ISO/IEC 27031:2025 Cybersecurity - Information and communication technology readiness for business continuity. ISO.

[8] Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research, 174(1), 23-37. https://doi.org/10.1016/j.ejor.2005.01.065

[9] Levy, S., Yao, R., Wu, Y., Dang, Y., Huang, P., Mu, Z., Zhao, P., Ramani, T., Govindaraju, N., Li, X., Lin, Q., Shafriri, G. L., & Chintalapati, M. (2020). Predictive and adaptive failure mitigation to avert production cloud VM interruptions. In 14th USENIX Symposium on Operating Systems Design and Implementation (pp. 1155-1170). USENIX Association.

[10] Li, J., Zhong, Y., Zhu, S., & Zhang, X. (2026). Grey-number PPO for uncertainty-aware multi-objective DAG scheduling in multi-cloud environments. Journal of King Saud University Computer and Information Sciences. https://doi.org/10.1007/s44443-026-00894-1

[11] Lou, C., Chen, C., Huang, P., Dang, Y., Qin, S., Yang, X., Li, X., Lin, Q., & Chintalapati, M. (2022). RESIN: A holistic service for dealing with memory leaks in production cloud infrastructure. In 16th USENIX Symposium on Operating Systems Design and Implementation (pp. 109-125). USENIX Association.

[12] Meng, K., Li, M., Ding, J., & Zhou, H. (2026). Cloud disaster recovery model based on failure prediction. Journal of Cloud Computing, 15, Article 33. https://doi.org/10.1186/s13677-026-00846-0

[13] Quinn, S., Ivy, N., Chua, J., Barrett, M., Feldman, L., Topper, D., Witte, G., & Gardner, R. (2025). Using business impact analysis to inform risk prioritization and response (NIST IR 8286D-upd1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8286D-upd1

[14] Roberts, M. K., Shanmugam, S. K., Alhammad, S. M., & Khafaga, D. S. (2026). Risk-aware resilient cloud orchestration under correlated faults using adaptive dual-regime search with self-healing control. Engineering Science and Technology, an International Journal, 77, Article 102365. https://doi.org/10.1016/j.jestch.2026.102365

[15] Stamenkov, G. (2022). Layered business continuity and disaster recovery model. Continuity & Resilience Review, 4(3), 267-279. https://doi.org/10.1108/CRR-05-2022-0008

[16] Swanson, M., Bowen, P., Phillips, A. W., Gallup, D., & Lynes, D. (2010). Contingency planning guide for federal information systems (NIST SP 800-34 Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-34r1

[17] Tomarchio, O., Calcaterra, D., & Di Modica, G. (2020). Cloud resource orchestration in the multi-cloud landscape: A systematic review of existing frameworks. Journal of Cloud Computing, 9, Article 49. https://doi.org/10.1186/s13677-020-00194-7

[18] Topcuoglu, H., Hariri, S., & Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 13(3), 260-274. https://doi.org/10.1109/71.993206

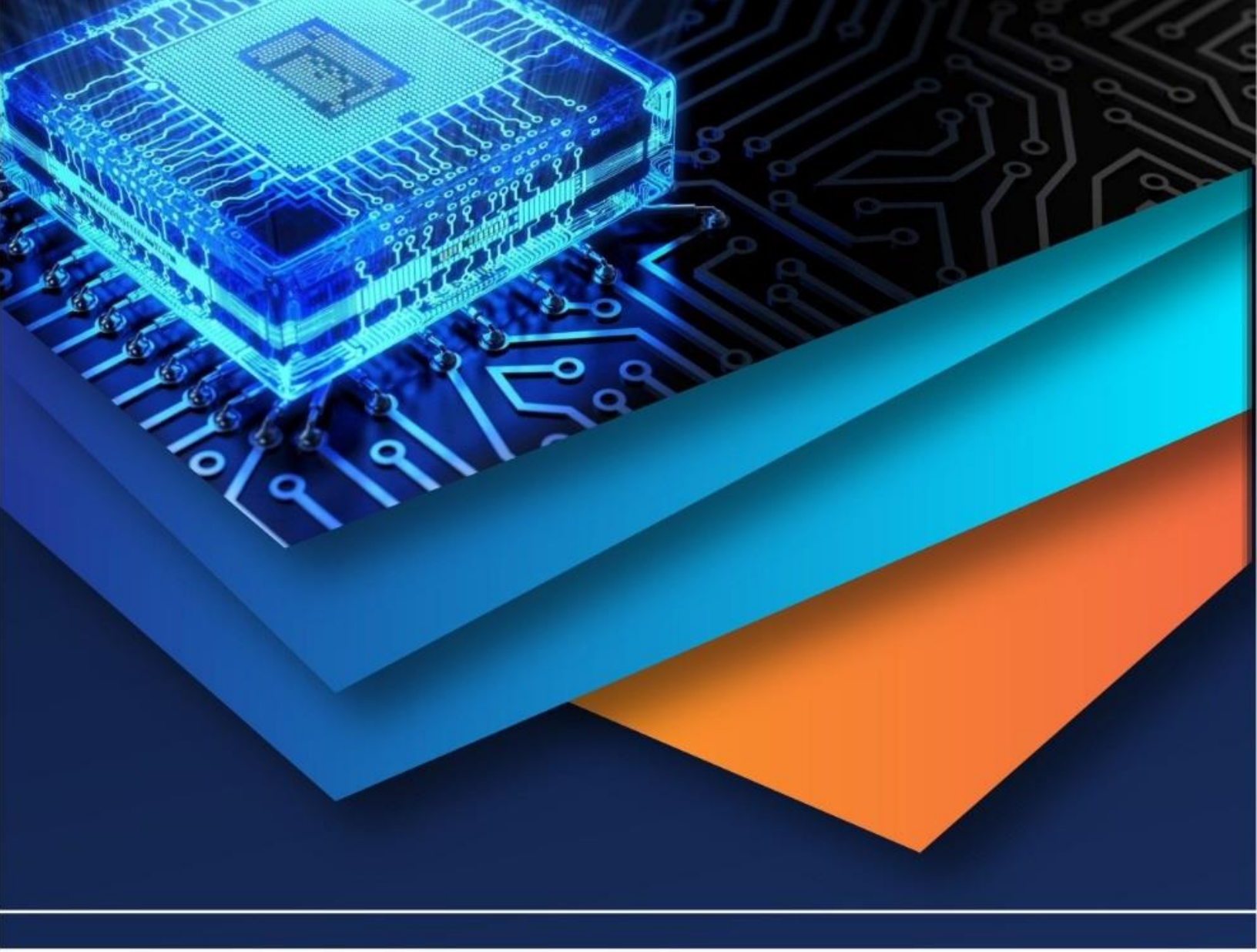
[19] Wang, L., He, J., Peng, J., Zhou, L., & Zhang, Z. (2025). NSGA-II based multi-objective disaster recovery scheduling for virtual cloud platforms. Informatica, 49(36). https://doi.org/10.31449/inf.v49i36.11126

[20] Wang, W., Ren, Q., Yao, G., & Zhu, S. (2026). Reliable cloud workflow scheduling with uncertain task execution time. Journal of King Saud University Computer and Information Sciences, 38, Article 216. https://doi.org/10.1007/s44443-026-00585-x

[21] Xie, Y., Wu, K., Jiang, Y., Zhang, X., & Cui, W. (2026). Hierarchical service chain orchestration for multi-cloud environments enabled by deep reinforcement learning. Journal of Cloud Computing, 15, Article 69. https://doi.org/10.1186/s13677-026-00874-w

[22] Yang, T., Li, B., Shen, J., Su, Y., Yang, Y., & Lyu, M. R. (2022). Managing service dependency for cloud reliability: The industrial practice. In 33rd IEEE International Symposium on Software Reliability Engineering Workshops (pp. 67-68). IEEE.

[23] Zhai, E., Chen, A., Piskac, R., Balakrishnan, M., Tian, B., Song, B., & Zhang, H. (2020). Check before you change: Preventing correlated failures in service updates. In 17th USENIX Symposium on Networked Systems Design and Implementation. USENIX Association.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY