

Article

ICG-Restore: Intent-Constrained, Graph-Enhanced LLM Planning with Minimal-Edit Repair for Post-Disaster Emergency Communication Recovery

Jinyin Bai ¹ , Wei Zhu ^{2,*}, Xiangchen Wang ¹ , Shiluo Guo ², Zongzhe Nie ², Tianjin Ni ¹, Jinji Zhou ¹ , Kaiyang Kou ³, Lingxin Xu ¹  and Yihao Zhong ¹

¹ School of Information Communication, National University of Defense Technology, Changsha 410073, China; baijinyin20@nudt.edu.cn (J.B.); wangxiangchen20@nudt.edu.cn (X.W.); nitianjin@nudt.edu.cn (T.N.); zhoujinji@nudt.edu.cn (J.Z.); xulingxin20@nudt.edu.cn (L.X.); zhongyihao@nudt.edu.cn (Y.Z.)

² Information and Communication Command Department, Information Support Force Engineering University, Wuhan 430035, China; guoshiluo23@nudt.edu.cn (S.G.); niezongzhe17@163.com (Z.N.)

³ Beijing Satellite Navigation Center, Beijing 100006, China; jacruise@126.com

* Correspondence: zhuwei929@hotmail.com

Abstract

Post-disaster emergency communication recovery is not merely a link-repair task but a high-level planning problem constrained by service priorities, inter-object dependencies, resource budgets, and time windows. Existing restoration optimization methods generally rely on fully structured inputs, whereas direct large language model (LLM) planning may produce fluent candidates that violate encoded prerequisites, stage-order relations, budget limits, or temporal constraints. To address this challenge, we propose ICG-Restore, an intent-constrained, graph-enhanced LLM planning framework with rule-consistent minimal-edit repair. ICG-Restore transforms mixed restoration requests and structured network observations into task packages that can be checked for validator-level feasibility under an encoded high-level constraint model and evaluated by downstream abstract executors or schedulers. The framework compiles natural-language requests, structured observations, and operational rules into a task-intent object; retrieves task-relevant context from a heterogeneous scenario graph and a restoration knowledge graph; generates stage-wise restoration candidates; and applies bounded local corrections to candidates that violate encoded constraints. In this paper, “minimal-edit” is a descriptive label for a bounded local repair principle that prioritizes less disruptive corrections. Candidates accepted by the validators are evaluated and ranked by a safety-aware agent executor operating in an abstract restoration action space. Experiments on controlled abstract topologies covering three scales, four restoration tasks, and five environmental evolution modes show that ICG-Restore improves validator-level constraint satisfaction and benchmark-estimated recovery utility. Compared with Direct-LLM, it improves CSR and CRS by 1.99% and 24.56%, respectively; benchmark-specific WCTC@5 structural-alignment diagnostic increases by 38.87%.



Academic Editor: Demos T. Tsahalidis

Received: 20 April 2026

Revised: 27 July 2026

Accepted: 27 July 2026

Published: 2 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: large language models; emergency communications; service recovery planning; task-intent constraints; minimal-edit repair

1. Introduction

Natural disasters, extreme weather events, major accidents, and other public emergencies can severely disrupt communication infrastructure by damaging backbone links, dis-

abling critical service nodes, and degrading regional access capability [1–3]. In post-disaster settings, emergency communication systems are essential for command and coordination, on-site dispatch, situation reporting, and public safety support, such that their restoration speed and quality directly affect rescue efficiency and overall system resilience [4,5]. Accordingly, post-disaster emergency communication recovery should not be regarded as a simple link-repair or resource-rescheduling problem. Instead, it is a high-level planning task shaped jointly by service priorities, dependencies among recovery objects, stage coordination requirements, budget limits, and risk-control constraints.

A substantial body of prior work has studied post-disaster restoration from the perspectives of critical infrastructure recovery sequencing [6], service continuity assurance [7], multi-objective recovery optimization [8], stochastic planning [9], and resilience-driven resource reconfiguration [10]. These approaches are strong in formal modeling, objective design, and interpretability, but they typically assume that system states, task goals, and constraint boundaries have already been specified in a highly structured form [11]. In disaster-response settings, however, decision makers may face heterogeneous, uneven, and partially unstructured information, including natural-language restoration requests and field reports, structured topology observations, service-dependency relations, scheduling rules, and evolving operating conditions [12–14]. How to transform such mixed inputs into interpretable and machine-consumable high-level task packages that satisfy an explicitly encoded constraint model remains insufficiently addressed. Here, satisfaction of the encoded model refers to validator-level feasibility and should not be interpreted as direct physical-network executability.

Recent advances in LLMs have shown strong capabilities in complex reasoning, task decomposition, natural language understanding, and structured generation, suggesting that they can serve as candidate generators for high-level planning [15–17]. In scenarios where textual requirements must be absorbed, phase goals summarized, and action logic organized, LLMs can substantially reduce the burden of manual rule engineering [18–20]. Yet, directly relying on an LLM for one-shot free-form generation in a high-constraint restoration setting is still problematic. First, the model may overlook structural dependencies among networks, services, regions, and channels, resulting in the wrong choice of recovery targets. Second, multi-stage plans often suffer from missing preconditions, mismatches between actions and targets, phase-order conflicts, budget overruns, and time-window violations. Third, the outputs often remain at the level of natural-language descriptions and cannot be readily converted into structured task packages that downstream schedulers or executors can consume. For complex recovery tasks, a more effective paradigm is not to treat the LLM as an end-to-end black-box decision maker, but to place it within a controlled workflow of explicit planning, structured retrieval, validation feedback, and constraint repair [21]. Graph-structured context helps the model reason over entities, relations, and dependency chains; stage-wise generation mitigates long-horizon drift; and plan repair is preferable to full regeneration because it preserves useful parts of the existing plan with minimal unnecessary disturbance [22].

Against this background, this study focuses specifically on the high-level planning problem of post-disaster emergency communication service recovery. Our objective is not to generate a natural-language description that merely sounds plausible or to produce device-level operational instructions. Instead, the objective is to produce a structured task package that can be checked against an encoded high-level constraint model, subjected to bounded local repair when violations are detected, evaluated under a common abstract executor, and represented in a machine-consumable schema for subsequent downstream processing. More concretely, given a natural-language recovery request, structured network observations, and operational rules, the planner must determine what should be restored,

what should be prioritized, how stage-wise actions should be organized under budget and time-window constraints, and how the resulting high-level plan should be exported for further engineering translation or scheduling.

To this end, we develop the ICG-Restore framework, which integrates task-intent modeling, graph-enhanced context retrieval, stage-wise candidate generation, explicit validation, bounded local repair, and abstract executor-based assessment into a controlled planning loop. The framework first compiles free-form restoration requests, structured observations, and rule boundaries into a task-intent object. It then constructs a heterogeneous scenario graph and a restoration knowledge graph and retrieves local structural context relevant to the current task. Conditioned on this context, an LLM generates stage-wise candidate restoration plans. The candidates are checked for structural completeness, encoded rule consistency, and causal compatibility. Candidates with supported and locally correctable violations are processed by the repair module, whereas candidates that remain invalid at the stopping boundary are classified as repair failures. Only validator-feasible candidates are evaluated and ranked by the safety-aware abstract executor. The resulting scores represent benchmark-estimated planning utility under the declared abstraction rather than measured performance on a physical communication network.

The main contributions of this work are as follows:

- We propose a rule-consistent minimal-edit repair component for LLM-based high-level planning in post-disaster emergency communication recovery. In this paper, “minimal-edit” denotes a bounded local repair principle that prioritizes corrections intended to preserve the original priority targets, stage organization, and restoration primitives whenever possible. The output is a structured task package whose feasibility can be checked only with respect to the encoded high-level validator model.
- We introduce a task-intent object that compiles natural-language restoration demands, structured observations, and rule boundaries into a stable intermediate representation. This design explicitly preserves service priorities, resource budgets, time windows, forbidden primitives, and completion criteria before stage-wise generation.
- We design a joint retrieval mechanism over a heterogeneous scenario graph and a restoration knowledge graph. The retrieved local context links damaged communication objects, service dependencies, restoration primitives, and rule constraints, thereby supporting more structurally grounded stage-wise planning.
- We build a safety-aware agent-based abstract execution environment for comparing validator-feasible restoration task packages under a common benchmark abstraction. The executor supports benchmark-estimated utility comparison and empirical cross-mode stability assessment across predefined environmental modes.

The remainder of the paper is organized as follows. Section 2 reviews related work and clarifies the positioning of this study. Section 3 formulates the problem. Section 4 presents the ICG-Restore framework. Section 5 describes the experimental platform and reports the results. Section 6 discusses the method’s effectiveness, limitations, and implications. Section 7 concludes the paper.

2. Related Work

2.1. Post-Disaster Recovery and Resilience Planning for Critical Infrastructure

Research on post-disaster recovery has long focused on sequencing restoration, allocating limited resources, and optimizing resilience for critical infrastructure such as power systems, transportation, communications, and water networks [23,24]. Existing studies have established a relatively mature theoretical foundation from the perspectives of multi-objective optimization, stochastic planning, service continuity, and restoration

sequencing [25,26]. These works provide important inspiration for high-level recovery planning under resource scarcity.

That said, most of this literature assumes that the recovery targets, constraints, and operational objects have already been formally specified. The core challenge is then to solve a structured optimization problem. Much less attention has been paid to how one should generate a high-level restoration plan from a mixture of natural-language requests, topological observations, and operational rules. For the problem addressed in this paper, the heterogeneity and irregularity of the input are themselves nontrivial. A stable structured intermediate layer is therefore needed between input understanding and downstream planning.

2.2. Emergency Communication Recovery and Service Continuity

Within emergency communications, prior research has investigated UAV-assisted wireless communication, post-disaster network reconfiguration, regional coverage restoration, relay deployment, spectrum reallocation, service migration, and priority-service assurance [27–30]. Compared with general infrastructure recovery, emergency communication restoration must reason not only over link connectivity but also over core service continuity, regional access capability, channel conflicts, and coordination latency across multiple layers.

Most existing studies formulate algorithms for specific subproblems—such as coverage recovery, trajectory planning, resource allocation, or service migration in UAV-assisted and post-disaster emergency communication systems [31,32]. They pay comparatively little attention to how high-level restoration intent can be transformed into stage-wise, structured task packages that downstream systems can directly consume. This is precisely the gap that the present study addresses: rather than solving one low-level control problem in isolation, we build a structured planning interface between high-level restoration intent and downstream execution modules.

2.3. LLM Planning, Retrieval-Augmented Generation, and Graph-Enhanced Generation

Retrieval-augmented generation further improves the knowledge grounding of language-model outputs by combining parametric generation with retrieved external evidence [33]. Recent graph-enhanced generation studies extend this idea by introducing graph-structured indices, relation-aware retrieval, or knowledge graph-based context organization [34,35]. These methods improve contextual relevance and structural consistency, but they do not by themselves guarantee executable plans under domain-specific budgets, time windows, and causal constraints.

LLMs have increasingly been applied to complex decision tasks because of their strong performance in multi-step reasoning, task decomposition, and structured generation [36]. Prior work suggests that, compared with one-shot free-form generation, explicit planning, retrieval augmentation, and graph-structured context can improve consistency and interpretability in complex tasks [37,38]. In particular, when dependencies are dense and object coupling is strong, graph structures help preserve the organization of entities, relations, and constraints, thereby reducing shallow matching and semantic drift caused by purely textual prompting.

Recent critical studies on LLM planning also suggest that LLM-generated plans may remain unreliable when executable action sequences and strict preconditions are required, which motivates the use of validation and repair mechanisms in high-constraint planning settings [39]. Even when graph-enhanced retrieval helps the model choose better targets and organize phases more sensibly, the resulting plan may still violate budgets, ordering constraints, or preconditions. Graph-enhanced retrieval should therefore be regarded

as an important support for high-level planning quality, not as a sufficient condition for executable plans.

2.4. Plan Repair and Rule-Consistent Minimal-Edit Repair

Classical AI planning has shown that, when a plan fails, repairing the existing plan is often preferable to replanning from scratch because it preserves valid structure and avoids unnecessary disruption [40,41]. In LLM-based structured generation, related work has also shown that explicit planning and self-planning can improve output organization, but these methods still do not replace domain-specific validation and repair when strict executability constraints are present [42,43].

This idea is highly compatible with emergency communication recovery planning. From the perspective of human decision makers, a plan that remains logically intact after a small number of targeted repairs is usually easier to review, more credible, and more practical for downstream coordination than a completely rewritten alternative. Yet, in current LLM-based planning studies, repair is often hidden inside “regenerate once more” or “filter multiple generations” pipelines, without an explicit, rule-driven, minimal-disturbance repair mechanism. This is the central gap targeted in our work.

2.5. Positioning of This Study

Compared with the above lines of work, this study focuses on high-level restoration planning for post-disaster emergency communication recovery, rather than low-level network control, device configuration, or a general-purpose LLM planning benchmark. The goal is to transform heterogeneous inputs—including natural-language restoration requests, structured topology observations, service priorities, and operational rules—into structured task packages that can be validated, repaired, evaluated, and consumed by downstream systems.

Recent RAG-based and graph-enhanced LLM planners have shown that retrieval and structural context can improve the relevance and consistency of generated plans. In emergency communication recovery, however, contextual relevance alone is insufficient. A high-level candidate must also be checked against service-priority constraints, cross-layer dependencies, action–target compatibility, budget envelopes, time windows, and fallback requirements. The novelty of ICG-Restore therefore does not lie simply in adding graph information to an LLM prompt, but in organizing generation, validation, bounded local repair, and abstract evaluation around validator-level feasibility under an explicitly encoded high-level constraint model.

Specifically, ICG-Restore differs from existing graph-based LLM planners in three aspects. First, it introduces a task-intent layer that explicitly constrains the planning process before generation. Second, it retrieves local planning context jointly from a heterogeneous scenario graph and a restoration knowledge graph, so that damaged objects, service dependencies, and restoration primitives are considered together. Third, it treats repair as a first-class planning component: infeasible candidates are not discarded or fully regenerated, but are corrected through rule-consistent minimal edits that preserve the original stage backbone whenever possible.

Accordingly, the present contribution should be understood as a controlled high-level planning workflow evaluated under the declared benchmark abstraction. The results provide evidence of framework-level consistency across the four evaluated backbone families.

3. Problem Definition

3.1. Object of Study and Boundary of the Planning Layer

This paper studies the high-level planning problem of post-disaster emergency communication service recovery rather than device-level control, protocol-level scheduling, or direct generation of real-network operations. Decision makers may need to reason over heterogeneous inputs that differ in source, granularity, and reliability, including natural-language restoration requests, structured topology observations, critical-service priorities, limited resource budgets, stage-specific time windows, and operational rule boundaries. The objective of the planning layer is to transform these inputs into stage-wise, structured, and evaluable task packages whose validator-level feasibility can be checked under the encoded high-level constraint model.

Compared with traditional path-restoration or resource-allocation problems, the problem considered here has three salient characteristics. The first is stage-wise progression: recovery is represented through stages such as assessment, localized restoration, priority-service assurance, and stabilization, each with different targets, resources, and completion criteria. The second is cross-layer coupling among connectivity, core services, regional access, dependencies, and channel resources. The third is interpretability and schema-level downstream compatibility: the output should be suitable for human review and machine processing by subsequent abstract executors or schedulers.

Throughout this paper, “executability” refers exclusively to validator-level feasibility under the encoded high-level constraint model. A plan is validator-feasible when it conforms to the task-package schema and satisfies the represented structural, rule-based, causal, budget, and temporal constraints. This definition does not imply physical-network executability, protocol correctness, equipment compatibility, field safety, or real-network deployment readiness. The experiments are conducted on controlled, self-constructed abstract topologies and are intended to compare relative planning behavior under common assumptions rather than reproduce a specific disaster-response communication network.

In this study, executability is defined at the high-level planning and validator layer. A plan is considered validator-feasible when it conforms to the task-package schema and satisfies the encoded structural, rule, causal, budget, and temporal constraints. This definition does not imply that the plan can be directly executed on a physical communication network without further engineering translation, scheduling, configuration, safety review, and human authorization.

3.2. State Representation and Heterogeneous Scenario Graph

To formalize the recovery process, the system state at discrete decision time t is defined as

$$z_t = \langle G_t, b_t, I_t \rangle \quad (1)$$

where G_t is the current heterogeneous scenario graph, b_t is a belief distribution over environmental evolution modes, and I_t is the compiled task-intent object.

The scenario graph is defined as

$$G_t = (V_t, E_t, T_V, T_E, X_t, W_t) \quad (2)$$

where V_t is the node set, E_t is the edge set, T_V is the node-type set, T_E is the edge-type set, X_t contains dynamic state attributes, and W_t contains structural weight attributes. The node types considered in this study include coordination centers, backbone gateways, core service nodes, relay nodes, frontier access nodes, and channel-resource nodes. The edge types include physical connectivity edges, logical dependency edges, service-carrying edges, and channel-mapping edges.

The scenario graph is not merely a topological container. It is a multi-layer heterogeneous representation that simultaneously captures who is connected to whom, who depends on whom, who carries which service, and who is constrained by whom. To support high-level planning, X_t further maintains key dynamic state variables such as node availability, residual link capacity, service integrity, channel availability, target confidence, and coordination delay. In this way, the graph evolves from a bare structural object into a unified information substrate for high-level recovery planning.

3.3. Task-Intent Object

A central difficulty in high-level recovery planning is that the input is not naturally computable. We therefore introduce a task-intent object that compiles free-text restoration requests, structured situational observations, and operational rule boundaries into a fixed-slot representation.

$$I_t = \langle g_t, O_t, B_t, T_t, E_t, A_t \rangle \quad (3)$$

where g_t denotes the overall restoration goal, O_t the set of prioritized objects, B_t the resource budget, T_t the execution time window, E_t the service priorities, risk preferences, and forbidden constraints, and A_t the completion criteria.

This representation serves two purposes. First, it compresses potentially long, ambiguous, or partially conflicting requests into a stable semantic slot structure on which subsequent modules can operate. Second, it explicitly preserves the most important boundary conditions of the restoration task: what must be restored first, under what budget, which primitives must not be used, and what counts as completion. The task-intent object is therefore not a simple summary of the input, but an intermediate layer that connects human restoration intent to the machine planning process.

3.4. Stage-Wise Restoration Plan

The output of high-level planning is not a single action but a stage sequence of length H .

$$\Pi_t = \{\phi_1, \phi_2, \dots, \phi_H\} \quad (4)$$

The h -th stage block is defined as

$$\phi_h = \langle T_h, P_h, A_h, R_h, W_h, \hat{e}_h, F_h \rangle \quad (5)$$

where T_h is the stage goal, P_h the prioritized objects in the stage, A_h the bundle of abstract restoration primitives, R_h the resource allocation, W_h the stage time window, $\hat{e}_h$ the expected restoration effect, and F_h the failure fallback strategy.

A stage-wise representation is used rather than a flat action sequence for three reasons. First, stage blocks better match how human decision makers typically think in disaster recovery—e.g., “assess first, recover critical services next, then stabilize the system”—and are therefore easier to interpret and audit. Second, stage blocks natively include targets, resources, and temporal bounds, making them more suitable for expressing constraints. Third, stage-wise plans can be exported naturally into downstream structured task packages without requiring secondary parsing from long-form natural language.

To connect with downstream execution modules, each stage is finally exported into a structured task package.

$$\Omega_h = \langle T_h, P_h, C_h, B_h, W_h \rangle \quad (6)$$

where C_h denotes the set of control-side constraints, including forbidden primitives, priority targets, service priorities, coordination windows, and fallback conditions, while B_h denotes the stage-level budget envelope.

3.5. Optimization Objective and Constraints

Given the current state z_t , the high-level planning objective is defined as

$$\max_{\Pi_t} J(\Pi_t) = \mathbb{E}_{m \sim b_t} [E(\Pi_t, m)] - \lambda_1 C(\Pi_t) - \lambda_2 R(\Pi_t) + \lambda_3 Q(\Pi_t) \quad (7)$$

where $E(\Pi_t, m)$ denotes the benchmark-estimated restoration utility of plan Π under predefined environmental mode m , $C(\Pi_t)$ denotes resource cost, $R(\Pi_t)$ denotes encoded operational risk, and $Q(\Pi_t)$ denotes an empirical cross-mode stability term. The utility is produced by the abstract executor and reflects the modeled critical-path reachability, service continuity, regional coverage improvement, and coordination delay. It should therefore be interpreted as benchmark-estimated utility under the declared abstraction rather than measured recovery quality in a real communication network.

Robustness is defined as

$$Q(\Pi_t) = \mu(E) - \eta\sigma(E) \quad (8)$$

The cross-mode stability component is calculated from the variation in abstract executor-estimated utility across the five predefined environmental modes. A higher value indicates that the plan exhibits more consistent benchmark performance across those modes.

At the same time, the plan must satisfy four classes of constraints:

$$\chi_{rule}(\Pi_t) = 1, \chi_{causal}(\Pi_t) = 1, \chi_{budget}(\Pi_t) = 1, \chi_{window}(\Pi_t) = 1 \quad (9)$$

These correspond to rule consistency, causal consistency, budget feasibility, and time-window feasibility, respectively. Rule consistency ensures compliance with service priorities and forbidden-primitives constraints. Causal consistency ensures that precedence relations, support relations, and action–target compatibility hold across stages. Budget feasibility prevents resource overruns, whereas time-window feasibility ensures that each stage can be completed sequentially within the allowed temporal limits.

Accordingly, the LLM is not treated as the final decision maker. Instead, it is embedded in a controlled planning pipeline consisting of task-intent constraints, graph-enhanced retrieval, stage-wise generation, rule-consistent repair, and agent-based execution evaluation. This formulation directly motivates the method presented next.

4. The Proposed Method: ICG-Restore

4.1. Overall Framework

Based on the formalization in Section 3, we propose the ICG-Restore framework. Its central idea is not to let the LLM output the final restoration plan directly, but to use the model to generate stage-wise candidate plans under task-intent constraints, graph-structured context, and explicit rules, after which validation, repair, and execution assessment complete the selection process. The framework comprises five components: task-intent compilation, unified graph substrate construction and local context retrieval, stage-wise candidate plan generation, rule-consistent minimal-edit repair, and safety-aware agent-based execution evaluation.

As illustrated in Figure 1, free-text restoration requests, structured network observations, and operational rules are first compiled into a unified task-intent object. A joint retrieval module then extracts local structural context from the heterogeneous scenario graph and the restoration knowledge graph. Under schema constraints and stage memory, the LLM generates candidate restoration plans. These candidates are passed through structural, rule-based, and causal validation; the repair module then edits invalid candidates into

executable task packages whenever possible. Finally, all feasible candidates are assessed and ranked by a safety-aware agent executor, and the framework outputs a preferred plan together with alternatives.

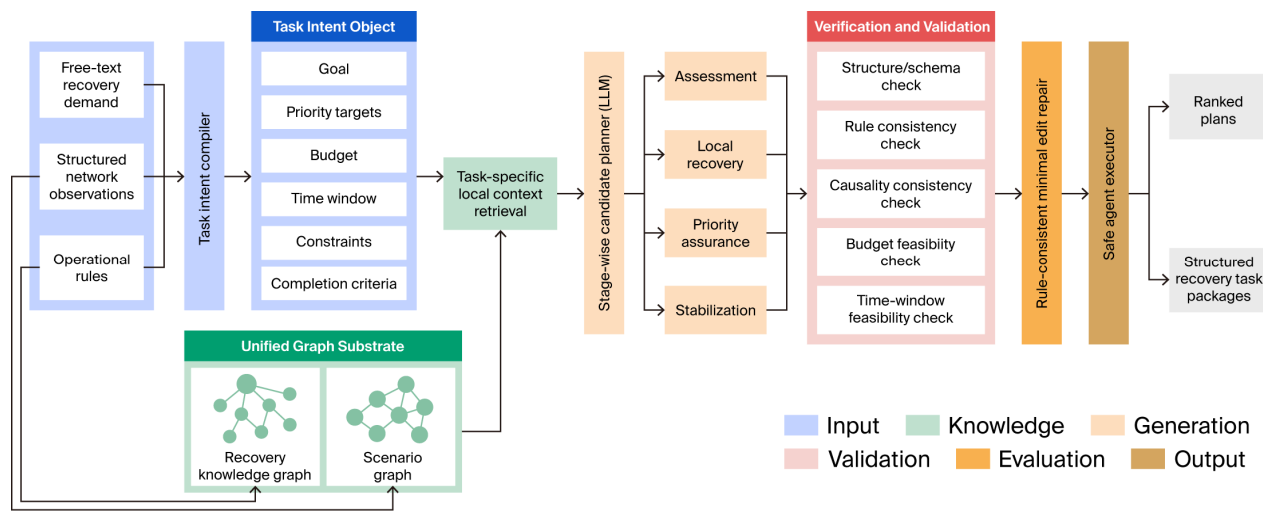


Figure 1. Overall architecture of ICG-Restore.

Each component plays a distinct role. Task intent defines what should be done and what must not be done. Graph retrieval determines which structural objects and dependencies should be considered. Stage-wise generation proposes how recovery might proceed. The repair module ensures whether the plan can actually be executed, and the agent executor evaluates whether the plan is worth adopting. It is precisely the coupling of these components into one coherent loop that turns a natural-language candidate into a downstream-consumable restoration task package.

4.2. Task-Intent Compilation

The first step in high-level restoration planning is not generation but making constraints explicit. In practice, restoration requests often arrive as a mixture of natural language, tabular observations, and partial situational cues. If these inputs are passed directly to an LLM without structured compilation, the model is prone to semantic drift, overly broad target selection, and rule omission during stage generation. To avoid this, we design a task-intent compiler that organizes the input into three categories: free-text restoration requests, structured situational observations, and operational rule boundaries.

The compiler outputs the fixed-slot intent object I_t , whose fields have clearly defined roles. The goal slot captures the overall direction of the restoration task; the prioritized-object slot identifies critical services, critical regions, or structurally important objects; the budget and time-window slots encode hard constraints; the service-priority and forbidden-primitives slots specify explicit operational boundaries; and the completion-criteria slot provides a stopping condition for downstream execution assessment. Only information that can be expressed within the schema is allowed to enter later modules. Inputs that cannot be normalized are treated as unreliable and are excluded from the decision chain.

The Task-Intent Compiler also includes a conflict-screening step before formal slot filling. In practice, natural-language restoration requests may contain ambiguous, incomplete, or even self-contradictory statements. The compiler therefore applies a deterministic precedence rule. Operational rules, including forbidden primitives, budget limits, and time-window constraints, have the highest priority. Structured observations are then used to verify factual states such as damaged objects, available resources, and service degrada-

tion. Natural-language requests are used to express restoration goals and preferences only when they do not conflict with these hard constraints. If a request contains incompatible priorities, impossible resource or time requirements, or primitives forbidden by the rule set, the conflicting element is either normalized into a conservative feasible slot or marked as unreliable and excluded from the decision chain. This design prevents contradictory human prompts from being silently converted into executable task packages and ensures that later generation and repair modules operate on rule-consistent intent slots.

To facilitate robust parsing, validation, and repair, we further require all candidate plans to be generated under a predefined JSON schema. Each stage must explicitly contain the following seven fields: goal, targets, primitives, resources, window, expected_effect, and fallback. Although this design imposes stronger generation constraints, it greatly improves the reliability of downstream parsing and repair, and is therefore a necessary step from free-form LLM output to machine-consumable task packages.

4.3. Unified Graph Substrate and Local Structural Context Retrieval

Task intent alone is not sufficient for high-quality restoration planning, because restoration objects are coupled through rich structural relations. We therefore build a unified graph substrate composed of a heterogeneous scenario graph and a restoration knowledge graph. The former captures what the current system state looks like; the latter captures which high-level restoration primitives are available and how they depend on one another.

As shown conceptually in Figure 2, the scenario graph represents coordination centers, backbone gateways, core service chains, relay nodes, frontier access clusters, and channel-resource nodes together with multiple relation types. The restoration knowledge graph organizes abstract restoration primitives, their preconditions, support relations, resource costs, and risk attributes. A joint retrieval module sits between them and combines semantic relevance, graph-structural relevance, and rule consistency to construct task-specific local structural context.

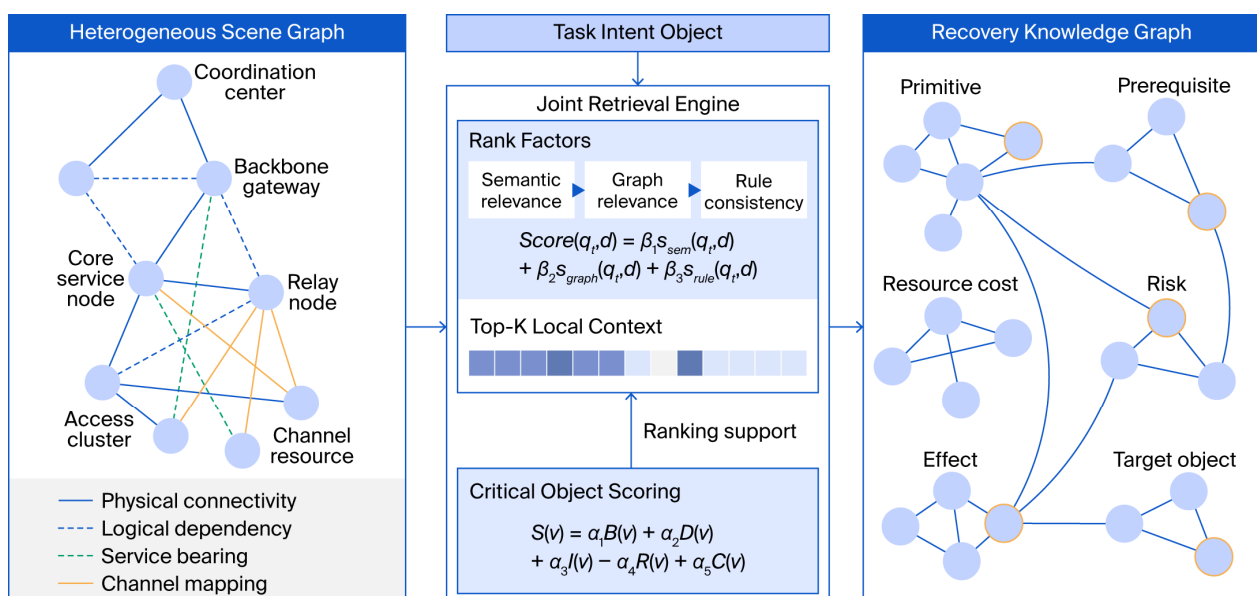


Figure 2. Unified graph substrate and task-relevant local structural context retrieval.

The scenario graph provides a unified entity–relation–state view, preventing the restoration problem from being reduced to a simple path-restoration problem. The restoration knowledge graph provides an operation–object–condition–effect organization of high-level recovery knowledge, preventing the LLM from freely hallucinating over an open-ended knowledge space. After joint retrieval, the LLM does not see all possible background

knowledge; it only sees the local planning context that is most relevant to the current task and still acceptable under the budget and rule boundaries.

To support local context construction, the scenario graph is used to identify structurally important objects relevant to the current task. In the implementation, object importance is estimated from normalized graph-side and service-side indicators, including topological centrality, service-dependency strength, business priority, redundancy, and cross-layer coupling. The resulting score serves only as an internal ranking signal for constructing structural reference sets and retrieval candidates, rather than as an additional optimization objective of the proposed framework.

The retrieval module further selects local context through a three-step process. First, candidate graph objects, dependency relations, and restoration primitives are collected from the neighborhoods of damaged or prioritized objects in the heterogeneous scenario graph and the restoration knowledge graph. Second, candidates that violate hard constraints, such as forbidden primitives or infeasible time-window conditions, are removed before ranking. Third, the remaining candidates are ranked by jointly considering semantic relevance to the task intent, graph proximity to prioritized or damaged objects, dependency relevance, and compatibility with budget and service-priority constraints. In the experiments, semantic relevance is computed using sentence-level embeddings, and the top ($K_r = 8$) local context units are retained for stage-wise LLM generation.

4.4. Stage-Wise Candidate Plan Generation

Once the task-intent object and local structural context are available, we do not generate the entire restoration plan in one shot. Instead, we adopt stage-wise progressive generation. The reason is simple: the most common failure in high-level recovery planning is not that an individual sentence sounds wrong, but that the stages become inconsistent in terms of targets, resources, preconditions, or time windows.

The generation of the h -th stage is written as

$$\phi_h = \text{LLM}(I_t, G_t^{\text{local}}, K_t, \phi_{1:h-1}) \quad (10)$$

meaning that the current stage depends not only on the task intent and local graph context, but also explicitly on previously generated stages. In this way, when the model generates stage h , it can reason about which critical objects have already been handled, which resources have already been consumed, and which preconditions have already been satisfied. This reduces redundant recovery actions, target drift, and budget loss of control.

Stage-wise generation also provides a stable operating unit for the repair module. Rather than editing the entire plan indiscriminately, the repairer can modify the goal, primitive bundle, resources, or order of one local stage block at a time. This is a major reason why minimal-edit repair is both more controllable and less disruptive than full regeneration.

4.5. Rule-Consistent Minimal-Edit Repair

Even with task-intent constraints, joint graph retrieval, and stage-wise generation, LLM-produced candidate plans can still contain several types of defects, such as missing preconditions, phase-order conflicts, action–target mismatches, budget overruns, and time-window violations. A straightforward response would be to discard any invalid candidate and ask the model to generate a new plan. In practice, however, this strategy has two drawbacks. First, regeneration destroys valid phase structure that is already present in the current candidate. Second, repeated regeneration increases inference cost without guaranteeing better plan quality. We therefore formulate repair not as a search for an entirely new plan, but as a minimal local feasibility transformation under explicit constraints.

Repair termination should not be confused with repair success. If the procedure reaches the maximum number of repair steps or the deviation boundary while residual violations remain, or if no applicable rule-consistent handler is available, the candidate is classified as a failed repair and is excluded from the validator-feasible candidate set.

The repair mechanism can directly address only violation types represented in the schema, scenario graph, rule base, and handler library. Unsupported primitives, invalid targets, action–target mismatches, missing prerequisites, and encoded budget or temporal conflicts may therefore be detected and rejected or repaired. Errors outside the encoded representations may remain undetected and require downstream engineering verification or human review.

Figure 3 conceptually shows the repair loop. A candidate plan is first passed through three validators: a structural validator, which checks field completeness and JSON validity; a rule validator, which checks budgets, forbidden primitives, and time windows; and a causality validator, which checks precedence relations and action–target compatibility. For each detected violation, the repairer applies a bounded local edit following a fixed priority order: fill missing preconditions first, adjust phase order second, adjust resources and time windows third, and replace primitives or targets only as a last resort. The repaired plan is then validated again, and the process repeats until all constraints are satisfied or a stopping condition is met.

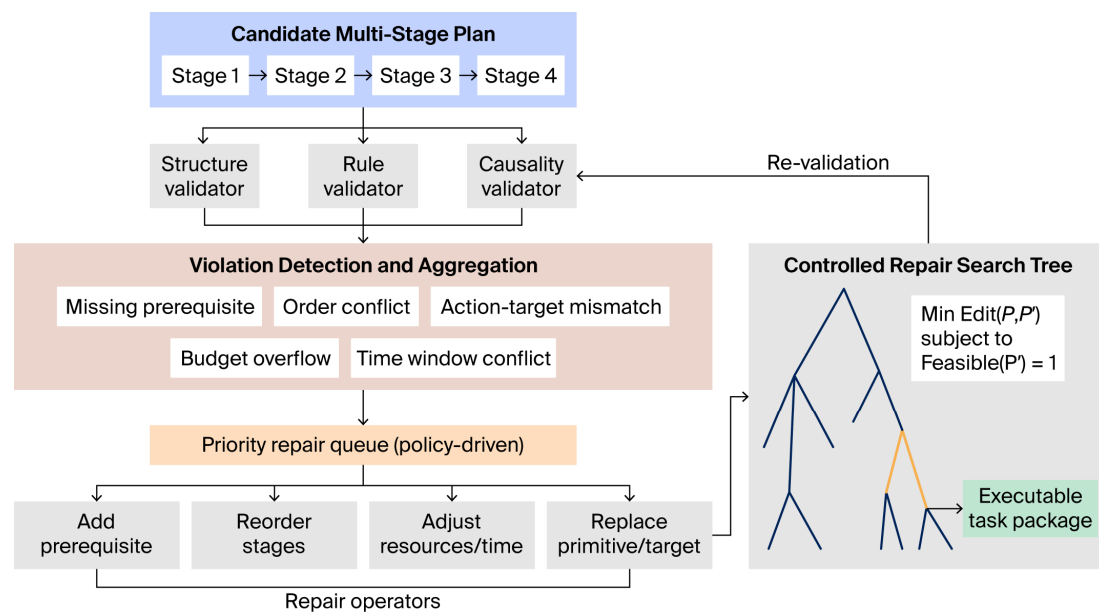


Figure 3. Illustration of the bounded local implementation of the minimal-edit repair principle.

The ideal minimal-edit repair target can be expressed as

$$\Pi_{repair}^* = \underset{\Pi'}{\operatorname{argmin}} \operatorname{Edit}(\Pi, \Pi') \text{ s.t. } \operatorname{Feasible}(\Pi') = 1 \quad (11)$$

where the edit cost is defined as

$$\operatorname{Edit}(\Pi, \Pi') = \lambda_{ins} N_{ins} + \lambda_{ord} N_{ord} + \lambda_{res} \Delta B + \lambda_{time} \Delta W + \lambda_{rep} N_{rep} \quad (12)$$

Here, N_{ins} is the number of inserted prerequisite stages, N_{ord} the number of ordering adjustments, ΔB the magnitude of budget adjustment, ΔW the magnitude of time-window adjustment, and N_{rep} the number of primitive or target replacements.

The priority rule has a clear motivation. In high-constraint restoration planning, many infeasible candidates are not globally misguided; they are locally flawed because they

omit a prerequisite, place one stage too early or too late, or bind resources improperly. If the repair module starts by replacing primitives or changing targets, it is likely to destroy the service-priority backbone of the plan and effectively degenerate into disguised regeneration. Prioritizing low-disturbance edits preserves the overall logic of the candidate while restoring executability.

Concretely, the repairer handles different violation types as follows. If a precondition is missing, it either inserts the required prerequisite stage or adds the supporting constraint field to the current stage. If a phase-order conflict is detected, the repairer reorders adjacent or local stages based on primitive dependency chains and structural support relations. If a budget overrun or time-window conflict occurs, the repairer shrinks resource allocations, adjusts stage windows, or postpones lower-priority tasks without breaking the core objective set. Only when these operations are insufficient does it replace a primitive or a target with a functionally similar but lower-cost, lower-risk, or more rule-consistent alternative.

Because repair itself can create new conflicts, we use an edit-and-revalidate loop. After each local edit, the revised plan is checked again by the structural, rule-based, and causal validators. The loop stops when one of the following conditions is met:

1. All validators return feasible;
2. The cumulative number of edit steps reaches a preset maximum M ;
3. Further editing would make the repaired plan deviate from the original candidate's objective set or phase backbone by more than a threshold τ .

Condition 2 controls repair cost, whereas condition 3 prevents over-repair, where a formally feasible result no longer preserves the original plan's core logic.

The repair module should be understood as a bounded, rule-guided local correction mechanism rather than an exact constrained optimizer or complete repair search. In this study, "minimal-edit" is used as a descriptive label for a low-disturbance repair principle: when a represented violation is detected, the implemented policy prioritizes an applicable correction intended to preserve the original priority targets, stage structure, and restoration primitives whenever possible. The term does not imply convergence to a validator-feasible plan, minimum edit distance, local or global optimality, an approximation guarantee, or completeness with respect to all possible repair sequences.

The repair procedure may produce one of three outcomes. First, the revised candidate may pass all structural, rule-based, and causal validators and is then classified as validator-feasible under the encoded high-level constraint model. Second, the procedure may reach the maximum repair-iteration or deviation boundary while residual violations remain. Third, it may stop because no applicable handler is available for a detected violation. The latter two outcomes are explicitly classified as repair failures and are excluded from the validator-feasible candidate set.

Repair success is conditional on the coverage of the encoded representations and handler library, but these operating conditions are not sufficient to guarantee success. A violation must be detected by the validators, an applicable handler must exist, and a suitable correction must remain within the iteration and deviation bounds. Even when these conditions hold, the fixed local policy may fail to reach a validator-feasible candidate that could potentially be reachable through another edit sequence. The framework therefore provides bounded failure reporting rather than a convergence, feasibility-completeness, optimality, or approximation guarantee.

4.6. Agent-Based Execution Evaluation and Plan Ranking

A repaired plan that passes all validators is not necessarily the best plan. To distinguish among multiple feasible candidates, we introduce an evaluation module consisting of a lightweight value estimator and a safety-aware agent executor. The former quickly filters

out clearly weak candidates; the latter evaluates the remaining plans in a unified abstract state space.

The executor maintains state variables such as node availability, residual link capacity, service integrity, channel availability, regional coverage, target confidence, and average coordination delay. Different restoration primitives induce different state transitions: wide-area assessment increases target confidence; backup-path activation improves the reachability of critical paths; portable relay deployment enhances local coverage; service migration strengthens core service continuity; and channel reallocation alleviates local conflicts and congestion.

For each feasible candidate plan, the executor performs repeated abstract simulations under a given environmental evolution mode or its belief distribution. A composite recovery score is then computed. Final ranking considers not only restoration benefit, but also repair perturbation and operational cost. This ensures that the preferred output is not simply the plan with the most polished wording, but the one that performs best under a common execution standard.

The executor is used as a common abstract evaluation environment rather than as a high-fidelity communication-network simulator. Accordingly, the resulting scores represent benchmark-estimated utility under the declared state-transition and action abstractions. They support controlled relative comparison among plans evaluated under identical assumptions.

4.7. Algorithm Summary and Pseudocode

For reproducibility, the overall ICG-Restore workflow is summarized in algorithmic form below. Unlike end-to-end generation pipelines, ICG-Restore follows a controlled loop of generate $\rightarrow$ validate $\rightarrow$ locally repair $\rightarrow$ uniformly evaluate. Its goal is not to write a plausible restoration narrative, but to generate a structured task package that is executable, evaluable, and usable by downstream systems such as Algorithm 1 and Algorithm 2.

Algorithm 1. Overall planning, repair, and ranking procedure of ICG-Restore

Input: Natural-language restoration request D_t ; structured observation O_t ; rule set R_t ; heterogeneous scenario graph G ; restoration knowledge graph K ; LLM planner M_{LLM} .

Output: Preferred plan Π^* ; alternative plan set A_k .

```

1:  $I_t \leftarrow \text{CompileIntent}(D_t, O_t, R_t)$ 
2:  $(G_t, K_t) \leftarrow \text{JointRetrieve}(I_t, G, K)$ 
3:  $C_{feas} \leftarrow \text{empty}$ 
4: for  $i = 1$  to  $N$  do
5:    $\Pi^{(0)} \leftarrow \text{StagewiseGenerate}(M_{LLM}, I_t, G_t, K_t)$ 
6:    $(\Pi, V(\Pi), m) \leftarrow \text{MinimalEditRepair}(\Pi^{(0)}, I_t, \{V_{str}, V_{rule}, V_{cau}\}, M, \tau)$ 
7:   if  $V(\Pi) = \text{empty}$  then
8:      $C_{feas} \leftarrow C_{feas} \cup \{\Pi\}$ 
9:   end if
10: end for
11: if  $C_{feas} = \text{empty}$  then
12:   return (empty, empty)
13: end if
14:  $S \leftarrow \text{BatchEvaluateAndScore}(E, C_{feas})$ 
15:  $(\Pi^*, A_k) \leftarrow \text{RankAndSelect}(C_{feas}, S, k)$ 
16: return  $(\Pi^*, A_k)$ 

```

Algorithm 2. Bounded local repair with iterative revalidation

Input: Initial candidate plan $\Pi^{(0)}$; task-intent object I_t ; validator set $\{V_{str}, V_{rule}, V_{cau}\}$; maximum repair steps M ; deviation threshold τ .

Output: Repaired plan Π ; final violation set $V(\Pi)$; cumulative edit count m .

```

1:  $\Pi \leftarrow \Pi^{(0)}$ 
2:  $m \leftarrow 0$ 
3:  $V(\Pi) \leftarrow \text{ValidateAll}(\Pi, V_{str}, V_{rule}, V_{cau})$ 
4: while  $V(\Pi) \neq \text{empty}$  do
5:   if  $m \geq M$  then
6:     break
7:   end if
8:   if  $\text{Deviation}(\Pi, \Pi^{(0)}) > \tau$  then
9:     break
10:  end if
11:   $e \leftarrow \text{SelectEditByPriority}(V(\Pi))$ 
12:   $\Pi \leftarrow \text{ApplyLocalEdit}(\Pi, e)$ 
13:   $m \leftarrow m + 1$ 
14:   $V(\Pi) \leftarrow \text{ValidateAll}(\Pi, V_{str}, V_{rule}, V_{cau})$ 
15: end while
16: return  $(\Pi, V(\Pi), m)$ 

```

5. Experimental Platform and Design

5.1. Experimental Objectives and Research Questions

This section evaluates ICG-Restore within the controlled high-level planning benchmark. The assessment concerns validator-level feasibility, supplementary structural alignment, benchmark-estimated utility, empirical cross-mode stability, and computational overhead. Five research questions are considered:

RQ1: Compared with classical restoration optimization, network-science heuristics, learning-based sequential restoration, and existing LLM planning approaches, can ICG-Restore consistently improve constraint satisfaction and overall restoration quality?

RQ2: Does graph enhancement improve target selection and stage organization so that the generated plans align more closely with structurally critical objects in the scenario graph?

RQ3: Compared with “regenerate after validation failure”, can rule-consistent minimal-edit repair convert infeasible candidates into executable plans with smaller disturbance and lower cost?

RQ4: Under dynamic conditions such as secondary failure, congestion escalation, delayed observation, and regional isolation drift, does the proposed framework remain stable and robust?

RQ5: Can the proposed method strike a reasonable balance among plan quality, repair cost, and planning latency, rather than improving quality only by spending more inference budget?

These five research questions are essential because they correspond to the main validity dimensions of the proposed framework. RQ1 evaluates whether ICG-Restore provides an overall advantage over optimization-based, heuristic, learning-based, and LLM-based baselines. RQ2 isolates the role of graph-enhanced retrieval in selecting structurally important recovery targets and organizing stage-wise plans. RQ3 examines whether rule-consistent minimal-edit repair is more effective and less disruptive than simple regeneration after validation failure. RQ4 tests whether the framework remains stable under realistic

forms of environmental evolution, such as secondary failures, congestion escalation, delayed observation, and regional isolation drift. RQ5 evaluates whether the improvement in recovery quality is achieved with acceptable latency and token cost. Together, these questions ensure that the evaluation covers not only final performance, but also module contribution, robustness, and practical computational overhead.

To avoid mistaking incidental instance-level fluctuations for real methodological differences, the empirical design follows a four-part evidence chain: overall performance, module contribution, cross-backbone robustness, and mechanistic case analysis.

5.2. Benchmark Topologies and Task Construction

To compare the methods under different scales and structural complexities, we construct three controlled abstract emergency communication topologies, denoted EC-S, EC-M, and EC-L, as summarized in Table 1. All three topologies share the same multi-layer heterogeneous semantic design and contain abstract coordination centers, backbone gateways, core service nodes, relay nodes, frontier access nodes, and channel-resource nodes. They differ in node scale, redundancy, cross-layer coupling, and modeled channel scarcity.

Table 1. Benchmark topology scales and structural characteristics.

Topology	Coordination Centers	Backbone Gateways	Core Service Nodes	Relay Nodes	Frontier Access Nodes	Channel-Resource Nodes	Total Nodes	Structural Characteristics
EC-S	2	4	4	4	8	2	24	Small scale; suitable for basic feasibility validation
EC-M	2	6	8	8	12	4	40	Medium scale; dependency chains are more complex
EC-L	3	8	12	12	20	5	60	Large scale; redundancy and coupling are substantially stronger

EC-S contains 24 nodes and is mainly used to verify basic feasibility. EC-M contains 40 nodes, with more dependency chains and more candidate restoration paths. EC-L contains 60 nodes and represents the most challenging setting, where cross-layer coupling, local redundancy, and resource conflict become simultaneously prominent.

Across these topologies, we define four classes of restoration tasks to cover the most typical high-level planning needs in post-disaster emergency communication recovery: backbone service path restoration, which tests the ability to identify and organize critical connectivity paths; backup-path activation and continuity assurance, which evaluates how effectively backup resources are used after primary-path failure; core service restoration, which examines whether the planner can organize stages around key service chains and priority services; and regional access-cluster coverage restoration, which assesses the coordination of relay deployment, channel management, and local coverage reinforcement.

For every combination of topology $\times$ task $\times$ environmental mode, 20 test instances are constructed, and each method is run with five random seeds. Thus, for each topology, one method is evaluated on $4 \times 5 \times 20 = 400$ test instances per seed; across all three topologies, each method completes 6000 planning–execution evaluations in total.

5.3. Environmental Evolution Modes and the Safety-Aware Agent Executor

To examine empirical cross-mode stability under controlled dynamic conditions, each test instance is assigned one of five predefined environmental evolution modes, as shown in Table 2. M1 represents stable recovery after initial damage; M2 introduces secondary failures in local components during Stages 2–3; M3 represents congestion escalation caused

by increasing modeled demand and channel conflict; M4 introduces delayed and noisy structured observations; and M5 represents regional-isolation drift through fluctuations in modeled reachability and coordination delay.

Table 2. Environmental evolution modes.

Mode	Name	Dynamic Change	Main Capability Tested
M1	Stable Recovery	No additional damage after the initial disruption; demand gradually declines	Basic planning quality
M2	Secondary Failure	Local secondary failures appear during Stages 2–3	Stage robustness and fallback design
M3	Congestion Escalation	Service load increases and channel conflicts intensify	Channel coordination and priority-service assurance
M4	Delayed Observation	Structured observations lag by 1–2 stages and include noise	Robust planning under partial observation
M5	Regional Isolation Drift	Local reachability fluctuates and coordination delay increases	Coverage restoration and coordination stability

These modes provide abstract benchmark perturbations for comparing relative planning behavior. Performance consistency across these modes is therefore reported as empirical cross-mode stability rather than formal robustness.

All methods are evaluated through the same safety-aware agent executor. The executor operates only on abstract restoration primitives and never accesses device-level commands or real operational procedures. Its maintained state variables include node availability, residual link capacity, service integrity, channel availability, regional coverage, target confidence, and average coordination delay. Different primitives update these variables according to predefined templates: wide-area assessment increases target confidence; backup-path activation improves critical-path reachability; portable relay deployment increases local coverage; service migration improves core service continuity; and channel reallocation mitigates conflict and congestion. This shared execution environment provides a common basis for fair comparison.

5.4. Baselines

To evaluate ICG-Restore comprehensively, we compare it against representative methods from four directions: classical restoration optimization, network-science heuristics, learning-based sequential restoration, and LLM planning.

For classical optimization, we use the Interdependent Network Design Problem (INDP) [11] as a representative method. INDP solves structured restoration under budget and operational constraints and serves as a strong reference for hard-constrained recovery planning.

For heuristic restoration, we use Cent-Restore [44], which schedules restoration based on the importance ranking of service-network components. It serves as a representative network-science baseline and tests whether structural centrality alone is sufficient for high-level recovery planning.

For learning-based restoration, we use DRL-Restore, a DDQN-based sequential restoration method. This baseline models post-disaster restoration as a sequential decision process and allows us to compare learned recovery sequences with our constrained graph-enhanced planning framework.

For LLM baselines, we include three representative variants. Direct-LLM [45] directly feeds natural-language requirements and observations into an LLM and asks it to output a stage plan. PS-LLM (Plan-and-Solve) first generates an explicit subtask plan and then

solves it. GraphRAG-LLM enhances generation with graph-structured retrieval, allowing us to isolate the contribution of graph retrieval alone.

Together, these baselines cover four major paradigms—structured optimization, heuristic restoration, learning-based restoration, and open-ended LLM planning—and provide a broad foundation for answering the five research questions.

5.5. Implementation Details and Fairness Settings

Unless otherwise stated in the cross-backbone experiments, all main experiments use gpt-oss:20b as the default backbone. All LLM-based methods—Direct-LLM, PS-LLM, GraphRAG-LLM, and ICG-Restore—share the same output JSON schema, the same set of abstract restoration primitives, the same budget/time-window encoding, the same candidate-plan parser, and the same safety-aware agent executor. The only differences are whether task-intent compilation, joint graph retrieval, and rule-consistent minimal-edit repair are used. The retrieval size, maximum repair iterations, and deviation threshold are treated as fixed operating parameters rather than learned quantities. The retrieval size controls context breadth, the iteration bound limits repair cost, and the deviation threshold constrains departure from the original planning intent.

Unless otherwise stated in the cross-backbone experiments, all main experiments use gpt-oss:20b as the default backbone. All LLM-based methods—Direct-LLM, PS-LLM, GraphRAG-LLM, and ICG-Restore—share the same output JSON schema, abstract restoration-primitive catalog, budget and time-window encoding, candidate parser, validators, and abstract executor. The methodological differences concern the use of task-intent compilation, joint graph retrieval, and bounded local repair. The retrieval size K_r , maximum repair-iteration count MMM , and deviation threshold τ are fixed operating parameters rather than learned quantities. The present study does not provide a quantitative sensitivity analysis of these parameters or a quantitative estimate of repair-failure rates; consequently, no claim of parameter robustness is made.

To ensure input parity, all methods are built from the same canonical structured instance, including topology state, damaged objects, service priorities, resource budgets, time windows, and forbidden primitives. For LLM-based methods, the natural-language request is generated from this canonical instance and is provided together with the same serialized structured fields. Therefore, the natural-language request does not introduce additional task information; it only reflects the practical interface through which human operators may express restoration goals. Traditional optimization, heuristic, and learning-based baselines directly consume the canonical structured instance because they do not require natural-language input. All outputs are converted into the same stage-wise task-package schema and evaluated by the same safety-aware executor.

To keep comparisons with traditional methods as fair as possible, INDP, Cent-Restore, and DRL-Restore are given the same structured state, constraints, and resource information as provided by the experimental platform. Their outputs are uniformly mapped to the same stage-task-package format before being sent to the common agent executor. In other words, we do not compare “who writes better natural language”; we compare which method can recover more critical objects with more appropriate stage logic and achieve a higher integrated restoration benefit under a shared execution standard. For reproducibility, the default retrieval size is set to $K_r = 8$. The graph neighborhood radius is two hops around damaged or prioritized objects. The candidate context units include object descriptions, typed relations, dependency paths, and restoration-primitive descriptions. All LLM-based methods using graph retrieval share the same retrieval configuration.

The main experiments use gpt-oss:20b as the default backbone, while the cross-backbone analysis additionally evaluates models from the Qwen, Gemma, and Mistral families. The experimental environment and major parameters are shown in Table 3.

Table 3. Experimental environment and major parameters.

Environment	Specification	Parameter	Value
Operating System	Microsoft Windows 11	Seed	5
CPU	AMD Ryzen 9 9950X3D	LLM	gpt-oss:20b
GPU	NVIDIA GeForce RTX 5090 D	LLM temperature	
Language	Python 3.11.15	Max tokens	3072
CUDA	13.1	Text top_k	8

5.6. Evaluation Metrics

We report four benchmark dimensions: validator-level feasibility, supplementary structural alignment, benchmark-estimated recovery utility, and computational efficiency.

First, the Constraint Satisfaction Rate (CSR) measures the proportion of final task packages that pass all encoded structural, rule-based, causal, budget, and time-window validators.

$$\text{CSR} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[\Pi_i \text{ is feasible}] \quad (13)$$

where $\mathbf{1}[\cdot]$ is the indicator function. A plan is deemed feasible only if it simultaneously satisfies rule consistency, causal consistency, budget feasibility, and time-window feasibility. Higher CSR indicates a stronger ability to output executable restoration task packages.

Second, Weighted Critical Target Coverage at k , reported as WCTC@5 in the experiments, is used as a supplementary, benchmark-specific structural-alignment diagnostic. For each benchmark instance, the reference set is constructed from normalized topology-side and service-side importance indicators defined by the experimental platform. WCTC@5 measures the weighted overlap between the targets selected by a plan and this fixed benchmark reference set.

$$\text{WCTC@}k = \frac{1}{N} \sum_{i=1}^N \frac{\sum_{v \in P_i \cap T_i^{(k)}} S_i(v)}{\sum_{v \in T_i} S_i(v)} \quad (14)$$

where P_i is the set of targets selected by the plan, $T_i^{(k)}$ is the top- k structural reference set derived from the structural score, and $S_i(v)$ is the structural criticality score of object v . Higher WCTC@ k indicates better coverage of the predefined structural reference set.

Third, we adopt the Composite Recovery Score (CRS) to measure the overall quality of plan execution.

$$\text{CRS} = \alpha R_{\text{path}} + \beta R_{\text{serv}} + \gamma R_{\text{cov}} - \delta C - \rho \text{Risk} + \eta \text{Rob} \quad (15)$$

where R_{path} , R_{serv} , and R_{cov} denote critical-path recovery, core-service restoration, and regional-coverage improvement, respectively; C is resource cost; Risk is operational risk; and Rob is cross-mode robustness. All subterms are normalized. A higher CRS indicates a better balance among restoration benefit, cost control, and robustness.

Finally, for the ablation and cross-backbone experiments, we also report average planning latency and total token consumption. Since traditional optimizers and LLMs differ substantially in computational paradigm, these efficiency metrics are mainly compared within the LLM family.

5.7. Main Results and Analysis

Table 4 reports the quantitative benchmark results on the three controlled topology scales, and Figure 4 provides a visual summary. ICG-Restore achieves the highest CRS across all three topology scales and the highest CSR among the evaluated LLM-based baselines. It also obtains the highest WCTC@5 values, which are reported as a supplementary benchmark-specific structural-alignment result. These findings indicate improved validator-level feasibility and higher abstract executor-estimated utility under the current benchmark.

Table 4. Main results of different methods on three topology scales.

Method	EC-S CSR	EC-S WCTC@5	EC-S CRS	EC-M CSR	EC-M WCTC@5	EC-M CRS	EC-L CSR	EC-L WCTC@5	EC-L CRS
INDP	1.000	0.3789	0.4213	1.000	0.3661	0.2934	1.000	0.2035	0.2712
Cent-Restore	1.000	0.2963	0.5057	1.000	0.2023	0.3412	1.000	0.1984	0.3012
DRL-Restore	1.000	0.1392 ± 0.0174	0.2574 ± 0.0134	1.000	0.1023 ± 0.0141	0.2125 ± 0.0242	1.000	0.0823 ± 0.0105	0.1742 ± 0.0358
Direct-LLM	0.9311 ± 0.0304	0.2829 ± 0.0218	0.4503 ± 0.0629	0.9212 ± 0.0246	0.3096 ± 0.0108	0.3307 ± 0.0413	0.9011 ± 0.0262	0.1576 ± 0.0121	0.2428 ± 0.0220
PS-LLM	0.9478 ± 0.0211	0.3248 ± 0.0256	0.4859 ± 0.0410	0.9156 ± 0.0247	0.3447 ± 0.0216	0.3564 ± 0.0338	0.9032 ± 0.0305	0.1840 ± 0.0107	0.2945 ± 0.0217
GraphRAG-LLM	0.9412 ± 0.0319	0.3669 ± 0.0227	0.5220 ± 0.0256	0.9297 ± 0.0234	0.3398 ± 0.0208	0.3733 ± 0.0341	0.9111 ± 0.0255	0.2068 ± 0.0134	0.2752 ± 0.0309
ICG-Restore	0.9536 ± 0.0213	0.4069 ± 0.0151	0.5514 ± 0.0249	0.9345 ± 0.0284	0.3814 ± 0.0188	0.4112 ± 0.0251	0.9202 ± 0.0303	0.2534 ± 0.0144	0.3126 ± 0.0234

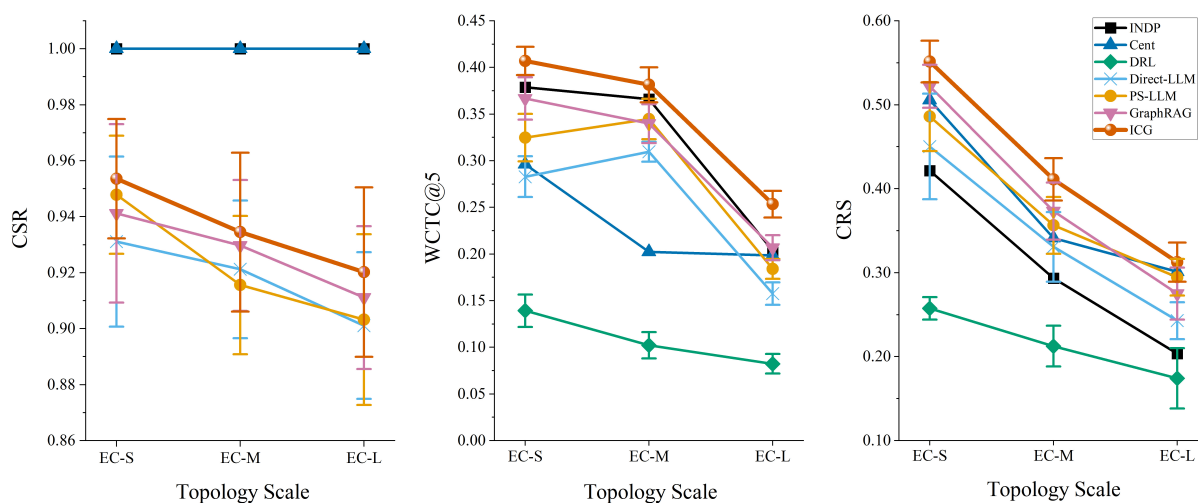


Figure 4. Main experimental results across three topology scales (EC-S, EC-M, and EC-L). Panels report CSR, WCTC@5, and CRS, respectively. Error bars denote mean $\pm$ standard deviation over five random seeds. Deterministic baselines use zero-variance placeholders for visual consistency. The proposed ICG-Restore is highlighted in orange.

Averaged over the three topology scales reported in Table 4, ICG-Restore obtains 0.9361 CSR, 0.3472 WCTC@5, and 0.4251 CRS. The corresponding averaged results of Direct-LLM are 0.9178 CSR, 0.2500 WCTC@5, and 0.3413 CRS. Therefore, relative to Direct-LLM, ICG-Restore improves CSR, WCTC@5, and CRS by 1.99%, 38.87%, and 24.56%, respectively. Relative to PS-LLM, the improvements are 1.51%, 22.05%, and 12.17%. Relative to GraphRAG-LLM, the gains are 0.95%, 14.03%, and 8.94%. These results indicate that direct generation, plan-then-solve prompting, or graph retrieval alone is not sufficient for stable high-constraint restoration planning. What matters is the closed-loop synergy among task-intent constraints, stage-wise generation, rule-consistent repair, and execution evaluation.

Figure 4 further suggests that the performance gap among LLM-based methods is relatively modest on CSR, but much larger on WCTC@5 and CRS. This implies that once all methods are forced to output structured plans under a common schema, the main bottleneck is no longer whether the model can produce a syntactically well-formed plan; instead, the bottleneck is whether it can organize recovery around the right critical objects, in the right stage order, under the right resource allocation. This is exactly where ICG-Restore offers the clearest advantage.

The benefit of ICG-Restore is more pronounced on larger topologies. On the most difficult setting, EC-L, ICG-Restore reaches 0.2534 in WCTC@5, exceeding 0.2068 for GraphRAG-LLM and 0.1840 for PS-LLM. Its CRS on EC-L is 0.3126, exceeding 0.2752 for GraphRAG-LLM and 0.2428 for Direct-LLM. Relative to GraphRAG-LLM, ICG-Restore improves WCTC@5 and CRS on EC-L by 22.53% and 13.59%, respectively. This suggests that as topology size grows, cross-layer dependencies deepen, and as resource competition intensifies, the value of task-intent constraints and minimal-edit repair becomes increasingly important. In other words, the more complex the topology is, the closer pure retrieval-enhanced generation gets to its ceiling, whereas the full loop of graph enhancement and rule-consistent repair continues to pay dividends.

One additional point deserves emphasis: feasibility and restoration utility do not always rise together. INDP, Cent-Restore, and DRL-Restore all achieve CSR = 1.000 on all three topologies, yet none surpasses ICG-Restore in CRS. Thus, being feasible by construction does not mean that a method restores the most valuable targets, nor that it yields the highest utility under the common executor. INDP achieves relatively strong WCTC@5 on EC-S and EC-M but lower CRS, indicating that it is good at recovering structurally critical objects under hard constraints but less effective at balancing service continuity and coverage restoration. Cent-Restore attains a relatively high CRS on EC-S (0.5057), showing that simple centrality heuristics can work surprisingly well on small topologies, but its WCTC@5 drops sharply on EC-M and EC-L, which suggests that local centrality alone cannot capture cross-layer dependencies, business priorities, or inter-stage causal structure. DRL-Restore remains feasible throughout, but its WCTC@5 and CRS are consistently weak, implying that reward-driven recovery sequences tend to become conservative strategies that are executable in form yet insufficiently focused on critical targets.

Overall, ICG-Restore reduces the validator-level feasibility gap between open-ended LLM planning and methods that produce constraint-compliant outputs by construction, while obtaining higher benchmark-estimated CRS under the common abstract executor. The supplementary WCTC@5 results indicate stronger alignment with the benchmark structural reference set.

Both the graph-retrieval process and WCTC@k rely, in different ways, on graph-related information. We therefore acknowledge that the metric is not conceptually independent of all structural assumptions in the framework. For this reason, WCTC@k is interpreted together with CSR, CRS, efficiency results, and ablation evidence rather than being used as the sole basis for evaluating the method.

5.8. Ablation Study

The ablation results provide benchmark-level evidence relevant to RQ2, RQ3, and RQ5. Task-intent compilation improves the preservation of encoded constraint boundaries and stage goals. Graph retrieval improves the supplementary WCTC@5 structural-alignment diagnostic and benchmark-estimated CRS. The bounded local repair component increases the proportion of candidates that pass the encoded validators and reduces latency and token use relative to full regeneration in the evaluated setting. The ablation results of ICG-Restore on EC-L are shown in Table 5.

Table 5. Ablation results of ICG-Restore on EC-L.

Variant	CSR	WCTC@5	CRS	Latency(s)	Token(k)
ICG-Restore	0.9202 $\pm$ 0.0303	0.2534 $\pm$ 0.0144	0.3126 $\pm$ 0.0234	8.58 $\pm$ 0.37	21.7 $\pm$ 0.9
w/o Intent Compilation	0.9024 $\pm$ 0.0227	0.2316 $\pm$ 0.0213	0.2718 $\pm$ 0.0216	8.11 $\pm$ 0.26	20.1 $\pm$ 0.8
w/o Graph Retrieval	0.9116 $\pm$ 0.0198	0.2189 $\pm$ 0.0197	0.2837 $\pm$ 0.0202	7.94 $\pm$ 0.31	19.2 $\pm$ 0.7
Repair $\rightarrow$ Regenerate	0.9145 $\pm$ 0.0201	0.2371 $\pm$ 0.0158	0.3014 $\pm$ 0.0211	11.96 $\pm$ 0.52	29.8 $\pm$ 1.2

Removing task-intent compilation reduces CSR, WCTC@5, and CRS to 0.9024, 0.2316, and 0.2718, respectively. Relative to the full model, the declines are 1.93%, 8.60%, and 13.05%. This confirms that the task-intent object is not a superficial summary but a critical intermediate layer that stabilizes constraints and stage objectives. Without it, the model is more likely to drift in target scope, miss priority objects, or weaken completion criteria, leading to simultaneous deterioration in structural alignment and execution quality. The ablation results on EC-L. Each panel reports one metric are shown in Figure 5.

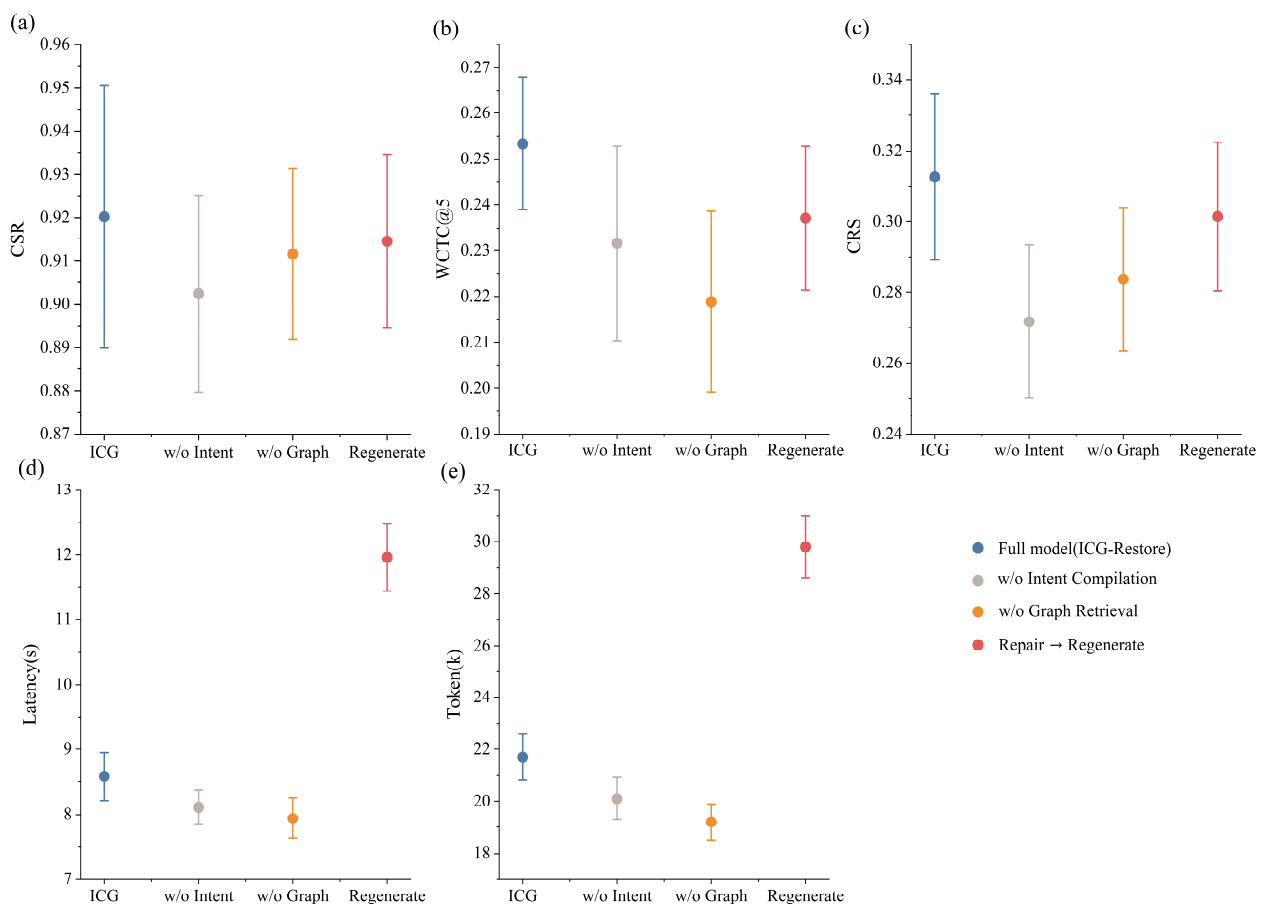


Figure 5. Ablation results on EC-L. Each panel reports one metric. (a) CSR metric, (b) WCTC@5 metric, (c) CRS metric, (d) Latency metric, (e) Token metric. Error bars denote mean $\pm$ standard deviation over five random seeds. The dashed horizontal line indicates the corresponding full-model performance.

Removing graph retrieval causes only a modest drop in CSR, from 0.9202 to 0.9116, but WCTC@5 drops sharply from 0.2534 to 0.2189, and CRS falls to 0.2837. Relative to the full model, the declines in WCTC@5 and CRS reach 13.61% and 9.25%, respectively. This indicates that graph enhancement mainly improves planning quality, especially the selection of critical objects and the structural rationality of stage organization, rather than merely increasing formal feasibility.

Replacing minimal-edit repair with regeneration after validation failure causes only moderate decreases in CSR, WCTC@5, and CRS, but planning latency rises from 8.58 s to 11.96 s, and token consumption increases from 21.7 k to 29.8 k, corresponding to increases of 39.39% and 37.33%, respectively. This is an important finding: in high-constraint restoration planning, regeneration is not the safer option it might appear to be. It often incurs a large computational penalty without delivering commensurate quality gains, and it is more likely to destroy the original phase backbone of the plan. By contrast, rule-consistent minimal-edit repair preserves useful structure while achieving a better balance between quality and cost.

The latency and token results also indicate that the closed-loop framework introduces a moderate but controlled overhead. On EC-L, ICG-Restore requires 8.58 s and 21.7 k tokens, whereas the variant without graph retrieval requires 7.94 s and 19.2 k tokens. This means that the additional graph-supported and repair-aware planning loop increases the cost by 0.64 s and 2.5 k tokens in this setting. However, replacing minimal-edit repair with regeneration increases latency to 11.96 s and token consumption to 29.8 k. Therefore, the proposed repair strategy is more efficient than full regeneration, reducing latency and token consumption by approximately 28.26% and 27.18%, respectively, compared with the Repair → Regenerate variant. These results suggest that the closed-loop design improves planning reliability without incurring an uncontrolled computational burden on the largest tested topology.

The ablation results therefore answer RQ2, RQ3, and RQ5 directly. Task-intent compilation stabilizes constraint boundaries and stage goals; graph retrieval improves critical-target coverage and structural plan quality; and minimal-edit repair is the key mechanism that turns semantically plausible yet structurally infeasible candidates into high-quality executable task packages. These modules are not interchangeable. Their gains are complementary.

5.9. Cross-Backbone Consistency

We examine whether the relative effect of the proposed workflow is observed across the four evaluated backbones: gpt-oss:20b and models from the Qwen, Gemma, and Mistral families. All backbones use the same input schema, prompt organization, parser, encoded constraint validators, and abstract executor. This experiment is intended to assess empirical consistency within the evaluated model set. The Cross-backbone consistency results are shown in Table 6.

Table 6. Cross-backbone consistency results within the evaluated model set.

Backbone	Direct-LLM CSR	ICG CSR	ICG WCTC@5	ICG CRS	Latency (s)	Token (k)
gpt-oss:20b	0.9178	0.9361	0.3472	0.4251	7.82	21.6
qwen3:14b	0.8927	0.9172	0.3185	0.3897	6.41	19.7
gemma3:27b	0.9231	0.9303	0.3390	0.4164	9.08	22.5
mistral-small3.1:24b	0.9143	0.9259	0.3218	0.4285	8.36	21.3

ICG-Restore improves executability over Direct-LLM on all four backbones. The CSR gains are 1.99% on gpt-oss:20b, 2.74% on qwen3:14b, 0.78% on gemma3:27b, and 1.27% on mistral-small3.1:24b. This shows that the improvement does not depend on a particular model's output style; it arises from the framework design itself.

The cross-backbone results further show that the runtime and token overhead remain within a practical range for high-level planning. Across the tested backbones, the average latency ranges from 6.41 s to 9.08 s, and token consumption ranges from 19.7 k to 22.5 k. This indicates that different backbones mainly shift the quality–efficiency trade-off,

while the framework-level overhead remains controlled. Since ICG-Restore operates at the high-level planning layer rather than the real-time device-control layer, this level of latency is acceptable for offline or operator-reviewed restoration planning, although further optimization would be required for continuous online replanning.

At the same time, the backbones exhibit stable but distinct quality–efficiency trade-offs. Under ICG-Restore, gpt-oss:20b achieves the highest WCTC@5 (0.3472), suggesting the strongest performance in critical-target selection and stage organization. mistral-small3.1:24b achieves the highest CRS (0.4285), indicating slightly better integrated recovery utility. qwen3:14b yields the lowest latency and token consumption (6.41 s and 19.7 k), though with lower WCTC@5 and CRS. gemma3:27b incurs the highest cost. Thus, different backbones mainly shift the position of the quality–efficiency frontier, rather than changing whether the framework works at all.

In short, the advantage of ICG-Restore is framework-level, not a by-product of one particularly strong LLM backbone.

5.10. Case Study: From an Infeasible Candidate to an Executable Restoration Task Package

To further illustrate the source of the performance gains, we analyze one representative instance from the main test set. The case is drawn from the EC-M topology, under the core service restoration task and the M2 (secondary failure) mode. Unlike the aggregate results in Sections 5.7–5.9, this subsection focuses on how ICG-Restore transforms heterogeneous inputs into an executable restoration task package in a concrete planning episode.

The input consists of a free-text restoration request, structured network observations, and operational rule boundaries. The task requires the planner to prioritize emergency command and medical-data return services under limited budget and a strict time window, while avoiding high-risk restoration primitives and retaining fallback capability. The damaged scenario shows a disrupted critical service chain between a core service node and a backbone gateway, degraded local access-cluster coverage, and additional constraints imposed by channel competition and coordination delay. In such a setting, direct LLM generation tends to produce plans with incomplete constraints and unstable phase organization. By contrast, ICG-Restore first compiles the heterogeneous input into a unified task-intent object, explicitly specifying the restoration goal, priority objects, budget, time window, forbidden primitives, and completion criteria.

The framework then performs joint retrieval of task-relevant local structural context. The heterogeneous scenario graph identifies the damaged subgraph centered on the backbone gateway–core service chain–damaged access cluster, while the restoration knowledge graph provides relevant high-level primitives and dependency patterns, including backup-path activation, service migration, portable relay deployment, channel reallocation, and stabilization. By combining semantic relevance, graph-structural relevance, and rule consistency, the retrieval module supplies a compact structural context for stage-wise planning.

Conditioned on the task intent and retrieved context, the LLM generates an initial candidate plan. Although semantically plausible, the candidate remains infeasible in two respects. First, it schedules service migration and coverage reinforcement before restoring the connectivity required to support them, leading to missing preconditions and causal-order violations. Second, it places relay deployment and high-cost channel adjustment in the same stage, causing budget and time-window conflicts. Rather than discarding the candidate, ICG-Restore applies minimal-edit repair, which preserves the original planning backbone while correcting local infeasibilities. The Violation diagnosis and minimal-edit repair trajectory are shown in Table 7.

Table 7. Violation diagnosis and minimal-edit repair trajectory in the representative case.

Original Candidate Stage	Main Problem	Repair Operation	Repaired Stage Package
Service migration to backup service node	Missing prerequisite of critical-link reachability; causal-order violation	Postpone the operation and insert a preceding “critical-link recovery” stage	Stage 2: Core service migration and priority-service assurance
Portable relay deployment to damaged access cluster	Misaligned with service priority; overly coupled with channel constraints	Bind it with channel reallocation and postpone it	Stage 3: Local coverage reinforcement and channel coordination
Backup-path activation	Scheduled too late to support later recovery	Move it forward and merge it with situation assessment	Stage 1: Situation assessment and critical-link recovery
Stabilization stage	Missing fallback and secondary-failure monitoring	Add explicit fallback and revalidation logic	Stage 4: Stabilization and fallback monitoring

After bounded local repair, the candidate becomes a coherent high-level sequence comprising situation assessment and critical-link recovery, core-service migration and priority-service assurance, local-coverage reinforcement and channel coordination, and stabilization and fallback monitoring. The revised candidate passes the encoded structural, rule-based, and causal validators. Most priority-service targets and restoration primitives are retained, indicating a low-disturbance local correction relative to the original candidate. This observation is descriptive and does not establish that the repair has minimum edit distance or is optimal among all feasible alternatives.

The repaired task package passes the encoded validators and is then evaluated by the common abstract executor. Relative to the unrepaired candidate, it receives higher modeled service-continuity and regional-coverage values and lower modeled coordination delay and operational risk. These differences indicate improved benchmark-estimated utility under the abstract executor.

This case also explains the ablation result of Repair → Regenerate in Table 5. Full regeneration increases token cost and planning latency, yet does not yield proportional quality gains. The key advantage of ICG-Restore therefore lies not in repeatedly rewriting candidate plans, but in preserving useful structure and converting it, with minimal disturbance, into an executable restoration task package.

To make the repair process verifiable, the exact original and repaired JSON-form plans of this case are provided in the Supplementary Materials. The original candidate contains a causal-order violation because service migration is scheduled before backup-path activation, and it also couples relay deployment with channel adjustment under an overly tight resource envelope. The repaired plan corrects these defects through local stage reordering, prerequisite insertion, resource adjustment, and fallback-field completion. Importantly, the priority service targets and most restoration primitives are preserved. Therefore, the repair is a bounded local transformation rather than a full regeneration.

Overall, the case illustrates how task-intent compilation, graph-enhanced retrieval, stage-wise generation, validation, bounded local repair, and abstract executor-based assessment interact within the framework. The resulting task package is validator-feasible under the encoded high-level constraint model and machine-consumable through the specified schema.

5.11. Statistical Analysis

To strengthen the empirical interpretation of the results, we further conduct statistical analysis for the LLM-based methods. ICG-Restore is compared with Direct-LLM, PS-LLM, and GraphRAG-LLM using paired Wilcoxon signed-rank tests and bootstrap 95% confidence intervals with 10,000 resamples.

The results show that ICG-Restore achieves the highest CSR, WCTC@5, and CRS among the LLM-based methods across all three topology scales. Specifically, ICG-Restore obtains CSR values of 0.9536, 0.9345, and 0.9202 on EC-S, EC-M, and EC-L, respectively. Its WCTC@5 values are 0.4069, 0.3814, and 0.2534, and its CRS values are 0.5514, 0.4112, and 0.3126. Averaged over the three topology scales, ICG-Restore obtains 0.9361 CSR, 0.3472 WCTC@5, and 0.4251 CRS. The statistical analysis further supports the robustness of these improvements over the LLM-based baselines under the current high-level planning benchmark.

6. Discussion

6.1. Why the Method Works

The benchmark performance of ICG-Restore arises from the interaction of several modules rather than from a single component. Task-intent compilation converts heterogeneous requirements into an explicit structured constraint object. Graph-enhanced retrieval narrows the context to task-relevant objects, relations, and restoration primitives. Stage-wise generation organizes candidate actions into auditable task blocks. When represented violations are detected, the bounded local repair policy attempts low-disturbance corrections while preserving the candidate backbone where possible; it does not guarantee that a validator-feasible repair will be found. Finally, the safety-aware abstract executor provides a common basis for estimating utility and empirical cross-mode stability under the benchmark assumptions.

The contribution is therefore not that the LLM produces a more persuasive restoration narrative. Rather, the framework provides a controlled interface through which LLM-generated candidates can be represented in a fixed schema, checked against encoded high-level constraints, subjected to bounded local correction, and compared under a common abstract executor. The resulting task packages are machine-consumable for subsequent processing, but further engineering translation, simulation, safety review, and human authorization would be required before any operational use.

6.2. Relation to Existing Paradigms

Compared with classical symbolic and hierarchical planning, ICG-Restore retains an emphasis on explicit structure, constraints, and interpretability while allowing mixed natural-language and structured inputs. Its outputs are not claimed to be physically executable plans; they are high-level task packages whose feasibility is evaluated with respect to the encoded validator model.

Compared with existing LLM + RAG or GraphRAG approaches, the present work makes executability the central design goal rather than topical relevance or textual consistency. In emergency communication recovery, a high-level plan that is structurally invalid is of little practical use, no matter how fluent it sounds. The emphasis here is therefore on passing the full chain of structure → rules → causality → execution, not merely on producing text that appears reasonable.

Compared with plan repair approaches, the contribution lies in incorporating a rule-guided bounded local repair policy as an explicit component of the graph-enhanced LLM planning workflow. The term “minimal-edit” describes the policy’s preference for less disruptive corrections and preservation of the original stage backbone.

6.3. Limitations and Threats to Validity

Despite the encouraging benchmark results, several limitations and threats to validity should be acknowledged. First, the experiments are conducted on controlled, self-constructed abstract topologies using a safety-aware abstract executor rather than opera-

tional communication networks, field disaster-response data, or high-fidelity engineering simulators. The benchmark topologies provide a common and reproducible environment for comparing relative planning behavior under identical assumptions, but they are not digital replicas of a particular disaster-response network. Consequently, the results demonstrate neither physical-network executability nor deployment readiness. Validation using field outage records, externally sourced network topologies, protocol- or engineering-level simulators, and operational expert review remains necessary.

Second, WCTC@5 is a supplementary, benchmark-specific structural-alignment metric derived from an author-designed structural scoring model. Although the reference scores are computed separately from the planner and are not directly exposed during generation, the retrieval mechanism and the metric both use structural information concerning connectivity, dependencies, and service importance. This conceptual overlap may favor plans that follow related structural principles.

Third, cross-backbone consistency is demonstrated only for the four evaluated backbones: gpt-oss:20b and the reported models from the Qwen, Gemma, and Mistral families. Different backbones continue to vary in structured-output stability, long-horizon planning, instruction following, and constraint adherence.

Finally, the present evaluation focuses primarily on offline high-level planning by a single planner and does not fully characterize all possible error and interaction modes. In particular, it does not quantify the frequency of individual hallucination categories, residual semantic errors, or cases in which bounded repair cannot produce a feasible plan. The validators reduce errors that are explicitly represented in the task schema, scenario graph, action catalog, and rule model, but they cannot guarantee the detection of every context-dependent or semantically inappropriate action. The current setting also excludes multi-operator coordination, human–AI co-decision, continuously changing observations, and online replanning during execution. More systematic parameter-sensitivity, failure-rate, out-of-distribution, and human-in-the-loop evaluations are therefore required before the framework can be considered sufficiently characterized for safety-critical operational use.

6.4. Practical Implications

From an application perspective, the proposed framework offers three practical implications. First, in highly constrained and strongly coupled restoration settings, LLMs are better used as candidate generators than as final decision makers. Second, graph structure and rule-consistent repair are not optional add-ons; they are necessary components for transforming semantically plausible plans into executable restoration task packages. Third, a practically useful high-level restoration planning system should provide a stable intermediate representation that can be reviewed by human operators and further consumed by downstream optimizers, schedulers, or multi-agent control modules.

6.5. Future Work

Several directions merit further investigation. First, the minimal-edit repairer could be extended to handle more complex cross-stage conflicts, such as long-horizon resource coupling, persistent time-window interactions, and multi-object coordination failures. Second, the structured task packages produced by ICG-Restore could be more tightly integrated with downstream multi-agent execution systems, thereby forming a closed loop of high-level planning, low-level execution, and feedback correction. Third, the framework should be tested in more realistic business-oriented simulation environments and combined with human-in-the-loop review, so as to improve practical usability and trustworthiness.

7. Conclusions

This paper addressed the problem of generating structured high-level plans for post-disaster emergency communication service recovery from heterogeneous inputs, including natural-language restoration requests, structured network observations, and encoded operational rules. The central objective was to produce task packages that are interpretable, machine-consumable, and validator-feasible under an explicitly encoded high-level constraint model.

To this end, we proposed ICG-Restore, which integrates task-intent compilation, joint retrieval over a heterogeneous scenario graph and a restoration knowledge graph, stage-wise candidate generation, explicit validation, bounded local repair, and safety-aware abstract executor-based evaluation. Under the controlled abstract benchmark, ICG-Restore achieves higher validator-level CSR and benchmark-estimated CRS than the evaluated LLM-based baselines. The supplementary WCTC@5 results indicate stronger alignment with the benchmark-defined structural reference set, while the environmental-mode results indicate empirical cross-mode stability across the five predefined modes. Cross-backbone results provide evidence of consistency within the four evaluated backbone families.

The significance of this work does not lie in making an LLM generate a restoration narrative that merely sounds more plausible. Rather, it lies in providing a controllable, repairable, and evaluable high-level planning framework that enables LLMs to produce structured task packages that can be consumed by downstream abstract executors or schedulers at the high-level planning layer. This perspective may also be useful for related problems in critical infrastructure recovery, complex system scheduling, and multi-stage intelligent decision making. At the same time, the current evidence should be understood as controlled benchmark validation of high-level planning feasibility, not as proof of immediate deployment readiness in real disaster-response communication systems.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/ai7080294/s1>, Edit_statistics_table.tex; Original_and_repaired_plans.json.

Author Contributions: Conceptualization, J.B. and W.Z.; methodology, J.B., W.Z. and X.W.; software, J.B., X.W., Z.N. and T.N.; validation, J.B., X.W., S.G., Z.N. and T.N.; formal analysis, J.B., X.W. and S.G.; investigation, J.B., S.G., J.Z. and K.K.; resources, W.Z. and S.G.; data curation, J.B., J.Z., K.K., L.X. and Y.Z.; writing—original draft preparation, J.B.; writing—review and editing, all authors; visualization, J.B. and X.W.; supervision, W.Z.; project administration, W.Z.; funding acquisition, W.Z. and S.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Social Science Fund (Grant No. 2025-SKJJ-D-048).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code and related implementation materials supporting the findings of this study are publicly available at <https://github.com/BaiBai2002/ICG-Restore> (accessed on 22 April 2026).

Acknowledgments: During the preparation of this study, the authors used [gpt, gpt-oss:20b] for the purposes of Experiment. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Matracia, M.; Saeed, N.; Kishk, M.A.; Alouini, M.-S. Post-disaster communications: Enabling technologies, architectures, and open challenges. *IEEE Open J. Commun. Soc.* **2022**, *3*, 1177–1205. [\[CrossRef\]](#)
- Wang, Q.; Li, W.; Yu, Z.; Abbasi, Q.; Imran, M.; Ansari, S.; Sambo, Y.; Wu, L.; Li, Q.; Zhu, T. An overview of emergency communication networks. *Remote Sens.* **2023**, *15*, 1595. [\[CrossRef\]](#)
- Bordel Sánchez, B.; Alcarria, R.; Robles, T. Managing wireless communications for emergency situations in urban environments through cyber-physical systems and 5G technologies. *Electronics* **2020**, *9*, 1524. [\[CrossRef\]](#)
- Vittal, V.; Nair, N.; Rahmatian, F. Smart city energy technology in the face of emergency situations: Electric supply, electric transportation, and communication. *IEEE Power Energy Mag.* **2022**, *20*, 16–25. [\[CrossRef\]](#)
- Okeukwu-Ogbonnaya, A.; Amariuca, G.; Natarajan, B.; Kim, H.J. Towards quantifying the communication aspect of resilience in disaster-prone communities. *Sci. Rep.* **2024**, *14*, 8837. [\[CrossRef\]](#) [\[PubMed\]](#)
- Sathurshan, M.; Saja, A.; Thamboo, J.; Haraguchi, M.; Navaratnam, S. Resilience of critical infrastructure systems: A systematic literature review of measurement frameworks. *Infrastructures* **2022**, *7*, 67. [\[CrossRef\]](#)
- Wang, Y.; Zhao, O.; Zhang, L. Multiplex networks in resilience modeling of critical infrastructure systems: A systematic review. *Reliab. Eng. Syst. Saf.* **2024**, *250*, 110300. [\[CrossRef\]](#)
- Tong, W.; Li, H.X.; Nasirzadeh, F.; Yao, F.; Ji, Y. Resilience of interdependent infrastructure networks: Review and future directions. *Int. J. Crit. Infrastruct. Prot.* **2025**, *51*, 100793. [\[CrossRef\]](#)
- Huang, X.; Wang, N. Post-disaster restoration planning of interdependent infrastructure systems: A framework to balance social and economic impacts. *Struct. Saf.* **2024**, *107*, 102408. [\[CrossRef\]](#)
- Liang, H.; Moya, B.; Chinesta, F.; Chatzi, E. Resilience-based post-disaster recovery optimization for infrastructure systems via deep reinforcement learning. *Reliab. Eng. Syst. Saf.* **2026**, *265*, 111478. [\[CrossRef\]](#)
- González, A.D.; Dueñas-Osorio, L.; Sánchez-Silva, M.; Medaglia, A.L. The interdependent network design problem for optimal infrastructure system restoration. *Comput.-Aided Civ. Infrastruct. Eng.* **2016**, *31*, 334–350. [\[CrossRef\]](#)
- Bartolini, N.; Ciavarella, S.; La Porta, T.F.; Silvestri, S. On critical service recovery after massive network failures. *IEEE/ACM Trans. Netw.* **2017**, *25*, 2235–2249. [\[CrossRef\]](#)
- Arrigoni, V.; Prata, M.; Bartolini, N. Recovering critical service after large-scale failures with Bayesian network tomography. *IEEE/ACM Trans. Netw.* **2024**, *32*, 5216–5231. [\[CrossRef\]](#)
- Basnayake, V.; Mamed, H.; Jayakody, D.N.K.; Canalda, P.; Beko, M. Adaptive emergency call service for disaster management. *J. Sens. Actuator Netw.* **2022**, *11*, 83. [\[CrossRef\]](#)
- Zhou, Y.; Luo, X.; Gu, X.; Zhang, W. Survey on intelligent planning methods from a large language model perspective. *J. Syst. Simul.* **2025**, *37*, 823–844. [\[CrossRef\]](#)
- de Carvalho, G.P.; Sawanobori, T.; Horii, T. Data-driven motion planning: A survey on deep neural networks, reinforcement learning, and large language model approaches. *IEEE Access* **2025**, *13*, 52195–52245. [\[CrossRef\]](#)
- Tantakoun, M.; Muise, C.; Zhu, X. LLMs as planning formalizers: A survey for leveraging large language models to construct automated planning models. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025), Vienna, Austria, 27 July–1 August 2025.
- Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R.K.W.; Lim, E.P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023), Toronto, ON, Canada, 9–14 July 2023.
- Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. In Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023.
- Yao, S.; Zhao, J.; Yu, D.; Shafran, I.; Narasimhan, K.R.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, 1–5 May 2023.
- Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; Hu, Z. Reasoning with language models is planning with a world model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023), Singapore, 6–10 December 2023.
- Töberg, J.-P.; Frese, U.; Cimiano, P. Generation of robot manipulation plans using generative large language models. *Int. J. Semant. Comput.* **2025**, *19*, 79–103. [\[CrossRef\]](#)
- Almoghathawi, Y.; Barker, K.; Albert, L.A. Resilience-driven restoration model for interdependent infrastructure networks. *Reliab. Eng. Syst. Saf.* **2019**, *185*, 12–23. [\[CrossRef\]](#)
- Alkhaleel, B.A.; Liao, X.; Sullivan, K.M. Model and solution method for mean-risk cost-based post-disruption restoration of interdependent critical infrastructure networks. *Comput. Oper. Res.* **2022**, *144*, 105812. [\[CrossRef\]](#)
- Li, Y.; Zhang, Z.; Jia, C.; Li, X.; Zhu, Y. Joint optimization of workforce scheduling and routing for restoring a disrupted critical infrastructure. *Reliab. Eng. Syst. Saf.* **2019**, *191*, 106551. [\[CrossRef\]](#)

26. Zhang, Z.; Ji, Y.; Wei, H.-H. Dynamic emergency inspection routing and restoration scheduling to enhance the post-earthquake resilience of a highway–bridge network. *Reliab. Eng. Syst. Saf.* **2022**, *220*, 108282. [\[CrossRef\]](#)
27. Zeng, Y.; Zhang, R.; Lim, T.J. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Commun. Mag.* **2016**, *54*, 36–42. [\[CrossRef\]](#)
28. Gu, X.; Zhang, G. A survey on UAV-assisted wireless communications: Recent advances and future trends. *Comput. Commun.* **2023**, *208*, 44–78. [\[CrossRef\]](#)
29. Luo, J.; Wang, Z.; Xia, M.; Wu, L.; Tian, Y.; Chen, Y. Path planning for UAV communication networks: Related technologies, solutions, and opportunities. *ACM Comput. Surv.* **2023**, *55*, 1–36. [\[CrossRef\]](#)
30. Li, L.; Zhu, L.; Huang, F.; Wang, D.; Li, X.; Wu, T.; He, Y. Post-disaster emergency communications enhanced by drones and non-orthogonal multiple access: Three-dimensional deployment optimization and spectrum allocation. *Drones* **2024**, *8*, 63. [\[CrossRef\]](#)
31. Zhang, S.; Shi, Y.; Feng, T.; Gu, X. Trajectory planning in UAV emergency networks with potential underlying D2D communication based on K-means. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 10. [\[CrossRef\]](#)
32. Zhang, Z.; Wang, Y.; Luo, Y.; Zhang, H.; Zhang, X.; Ding, W. Iterative trajectory planning and resource allocation for UAV-assisted emergency communication with user dynamics. *Drones* **2024**, *8*, 149. [\[CrossRef\]](#)
33. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Online, 6–12 December 2020.
34. Li, C.; Song, Y.; Li, A. GRAG-ZRE: Graph retrieval-augmented generation for zero-shot relation extraction in domain-sensitive scenarios. In *Proceedings of the 21st International Conference on Intelligent Computing (ICIC 2025), Ningbo, China, 26–29 July 2025*; Springer Nature: Singapore, 2025.
35. Zhang, L.; Zhu, Q. ActivityRDI: A centralized solution framework for activity retrieval and detection intelligence based on knowledge graphs, large language models, and imbalanced learning. *Mach. Learn. Knowl. Extr.* **2026**, *8*, 75. [\[CrossRef\]](#)
36. Liu, B.; Fang, Y.; Xu, N.; Hou, S.; Li, X.; Li, Q. Large language models for knowledge graph embedding: A survey. *Mathematics* **2025**, *13*, 2244. [\[CrossRef\]](#)
37. You, Y.; Liu, Z.; Wen, X.; Zhang, Y.; Ai, W. Large language models meet graph neural networks: A perspective of graph mining. *Mathematics* **2025**, *13*, 1147. [\[CrossRef\]](#)
38. Bai, J.; Zhu, W.; Wang, X.; Kou, K.; Guo, S.; Liu, S.; Li, D.; Ni, T.; Zhou, J.; Zhong, Y. A hierarchical multi-agent reinforcement learning framework with high-level guidance from large language models. *Sci. Rep.* **2026**. [\[CrossRef\]](#) [\[PubMed\]](#)
39. Valmeekam, K.; Marquez, M.; Sreedharan, S.; Kambhampati, S. On the planning abilities of large language models: A critical investigation. In Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023.
40. Babli, M.; Sapena, Ó.; Onaindia, E. Plan commitment: Replanning versus plan repair. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106275. [\[CrossRef\]](#)
41. Zaidins, P.; Goldman, R.; Kuter, U.; Nau, D.; Roberts, M. HTN plan repair algorithms compared: Strengths and weaknesses of different methods. In Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS 2025), Melbourne, Australia, 9–14 November 2025.
42. Jiang, X.; Dong, Y.; Wang, L.; Fang, Z.; Shang, Q.; Li, G.; Jin, Z.; Jiao, W. Self-planning code generation with large language models. *ACM Trans. Softw. Eng. Methodol.* **2024**, *33*, 1–30. [\[CrossRef\]](#)
43. Zhang, S.; Chen, Z.; Shen, Y.; Ding, M.; Tenenbaum, J.B.; Gan, C. Planning with large language models for code generation. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, 1–5 May 2023.
44. Ulasan, A.; Ergun, Ö. Restoration of services in disrupted infrastructure systems: A network science approach. *PLoS ONE* **2018**, *13*, e0192272. [\[CrossRef\]](#) [\[PubMed\]](#)
45. Olaiya, K.; Delnevo, G.; Lam, C.-T.; Pau, G.; Salomoni, P. From prompts to paths: Large language models for zero-shot planning in unmanned ground vehicle simulation. *Drones* **2025**, *9*, 875. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.