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

Jinyin Bai , Wei Zhu * , Xiangchen Wang , Shiluo Guo , Zongzhe Nie , Tianjin Ni , Jinji Zhou , Kaiyang Kou , Lingxin Xu , Yihao Zhong

Posted Date: 23 April 2026

doi: 10.20944/preprints202604.1634.v1

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

 doi:10.20944/preprints202604.1634.v1

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] National University of Defense Technology, Changsha 410073, China
[2] Information Support Force Engineering University, Wuhan 430035, China
[3] Beijing Satellite Navigation Center, Beijing 100006, China
* 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 often produces fluent yet operationally infeasible plans due to missing prerequisites, stage-order conflicts, and budget violations. To address this challenge, we propose ICG-Restore, an intent-constrained, graph-enhanced LLM planning framework with rule-consistent minimal-edit repair. The framework compiles natural-language requests, structured network observations, and operational rules into a task-intent object, retrieves task-relevant local context from a heterogeneous scenario graph and a restoration knowledge graph, generates stage-wise restoration candidates, and repairs infeasible plans through bounded edits that preserve the original planning backbone. Feasible candidates are then evaluated and ranked by a safety-aware agent executor in an abstract restoration action space. Experiments on three topology scales, four restoration tasks, and five environmental evolution modes demonstrate that ICG-Restore consistently improves executability, critical-target coverage, and overall recovery quality. Compared with Direct-LLM, it improves CSR, WCTC@5, and CRS by 1.99%, 38.87%, and 24.56%, respectively.

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, disabling 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

 doi:10.20944/preprints202604.1634.v1

in a highly structured form [11]. In real disaster settings, however, decision makers usually face heterogeneous, uneven, and partially unstructured information. Such inputs may include natural-language restoration requests and field reports, structured topological observations, service dependency relations, scheduling rules, and a continuously evolving operating environment [12–14]. How to transform such mixed inputs into high-level restoration plans that are executable, interpretable, and directly consumable by downstream systems remains insufficiently addressed.

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, but to produce a structured task package that can be validated, repaired, evaluated, and directly consumed by downstream systems. More concretely, given a natural-language recovery request, structured network observations, and operational rules, the planner must answer the following questions: what should be restored, what should be restored first, which objects should be prioritized, how should stage-wise actions be organized under limited budgets and time windows, and how should the resulting high-level plan be exported as a structured task package for downstream processing?

To this end, we develop the ICG-Restore framework, which unifies task-intent modeling, graph-enhanced context retrieval, stage-wise candidate generation, rule-consistent repair, and agent-based execution assessment into one closed 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, from which it jointly retrieves local structural context relevant to the current task. Conditioned on this context, an LLM generates stage-wise candidate restoration plans. These candidates are subsequently checked for structural completeness, rule consistency, and causal validity; partially infeasible plans are repaired by a minimal-edit repairer; and finally, all feasible candidates are evaluated and ranked by a safety-aware agent executor operating over abstract restoration primitives.

The main contributions of this work are as follows:

- We propose a rule-consistent minimal-edit repair framework for LLM-based planning in high-constraint post-disaster emergency communication recovery.
- We introduce a task-intent object that compiles natural-language restoration demands, structured observations, and rule boundaries into a stable intermediate representation.
- We design a joint retrieval mechanism over a heterogeneous scenario graph and a restoration knowledge graph to provide task-relevant local structural context for stage-wise plan generation.

 doi:10.20944/preprints202604.1634.v1

- We build a safety-aware agent-based execution environment that supports executable validation, quality comparison, and ranking of candidate restoration task packages under a unified abstraction layer.

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 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, path restoration, resource allocation, or service migration [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

LLMs have increasingly been applied to complex decision tasks because of their strong performance in multi-step reasoning, task decomposition, and structured generation [33–35]. 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 [36,37]. 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.

However, for highly constrained restoration planning, graph enhancement alone does not guarantee executability [38]. Even when graph-enhanced retrieval helps the model choose better

 doi:10.20944/preprints202604.1634.v1

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 [39,40]. The idea of rule-consistent minimal-edit repair emphasizes that repair should not merely produce a new feasible plan; it should also preserve as much as possible of the original phase logic, resource commitments, and coordination relationships [41,42].

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 differs in three key ways. First, the object of study is high-level restoration planning, not low-level network control. Second, the output is a structured task package, not a natural-language restoration narrative. Third, the proposed contribution is not merely graph enhancement, but a complete framework combining task-intent constraints, graph-enhanced stage-wise generation, rule-consistent minimal-edit repair, and safety-aware execution evaluation. It is this closed loop that transforms LLM outputs from "plans that sound reasonable" into "task packages that a system can actually consume".

Accordingly, while the framework is developed for emergency communications, it can also be viewed more broadly as a graph-enhanced LLM planning paradigm for high-constraint complex tasks. The methodology has implications for agent planning, structured decision making, and executability repair in other domains as well.

2. 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. After earthquakes, floods, typhoons, wildfires, and other disruptive events, decision makers do not face a static and fully specified optimization model. Instead, they must reason over an evolving mixture of heterogeneous inputs that differ in source, granularity, and reliability. These inputs include natural-language restoration requests, structured topology observations, critical-service priorities, limited resource budgets, stage-specific time windows, and various operational rule boundaries. The goal of high-level restoration planning is to transform these mixed inputs into stage-wise, structured, executable, and evaluable restoration task packages, rather than leaving the result at the level of textual explanation.

Compared with traditional path restoration or resource allocation problems, the problem considered here has three salient characteristics. The first is stage-wise progression: post-disaster recovery is rarely completed in one step. It typically proceeds through stages such as situation assessment, localized recovery, priority-service assurance, and stabilization, each with different target objects, resource requirements, and success criteria. The second is cross-layer coupling: communication recovery concerns not only physical connectivity but also core services, regional

 doi:10.20944/preprints202604.1634.v1

access, business dependencies, and channel resources. The third is interpretability and downstream usability: the output plan must be suitable both for human review and for machine consumption by agent executors or downstream schedulers.

To clarify the scope of the study, we make three boundary assumptions. First, the planner operates only at the level of abstract restoration primitives and does not produce any device-level commands, protocol configurations, or real operational procedures. Second, its role is to output high-level restoration task packages, not to replace downstream executors. Third, the paper evaluates the executability, structural soundness, and recovery benefit of the high-level planning framework itself, rather than the engineering details of low-level communication systems.

3.2. State Representation and Heterogeneous Scenario Graph

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

$$z_t = \langle G_t, b_t, I_t \rangle \tag{1}$$

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

$$G_t = (V_t, E_t, T_V, T_E, X_t, W_t) \tag{2}$$

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.

$$I_t = \langle g_t, O_t, B_t, T_t, E_t, A_t \rangle \tag{3}$$

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.

 doi:10.20944/preprints202604.1634.v1

$$\Pi_t = \{\phi_1, \phi_2, \ldots, \phi_{\mathrm{H}}\} \tag{4}$$

The h-th stage block is defined as

$$\phi_{\mathrm{h}} = \langle T_{\mathrm{h}}, P_{\mathrm{h}}, A_{\mathrm{h}}, R_{\mathrm{h}}, W_{\mathrm{h}}, \hat{e}_{\mathrm{h}}, F_{\mathrm{h}} \rangle \tag{5}$$

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, $\hat{e}_{\mathrm{h}}$ 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.

$$\Omega_{\mathrm{h}} = \langle T_{\mathrm{h}}, P_{\mathrm{h}}, C_{\mathrm{h}}, B_{\mathrm{h}}, W_{\mathrm{h}} \rangle \tag{6}$$

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

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

where $E(\Pi_t,m)$ denotes the restoration utility of plan Π_t under environmental evolution mode m, $C(\Pi_t)$ the resource cost, $R(\Pi_t)$ the operational risk, and $Q(\Pi_t)$ the robustness term. The restoration utility is not limited to link recovery; it jointly reflects critical-path reachability, service continuity, regional coverage improvement, and reduced coordination delay. Resource cost includes both stage-level resource consumption and the additional cost introduced by executability-preserving repair. Operational risk characterizes the vulnerability of the plan under evolving conditions.

Robustness is defined as

$$\mathrm{Q}(\Pi_{\mathrm{t}}) = \mu(E) - \eta\sigma(E) \tag{8}$$

which means that an effective plan should achieve not only high expected restoration benefit but also low performance variance across environmental evolution modes.

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

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

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.

 doi:10.20944/preprints202604.1634.v1

2. 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.

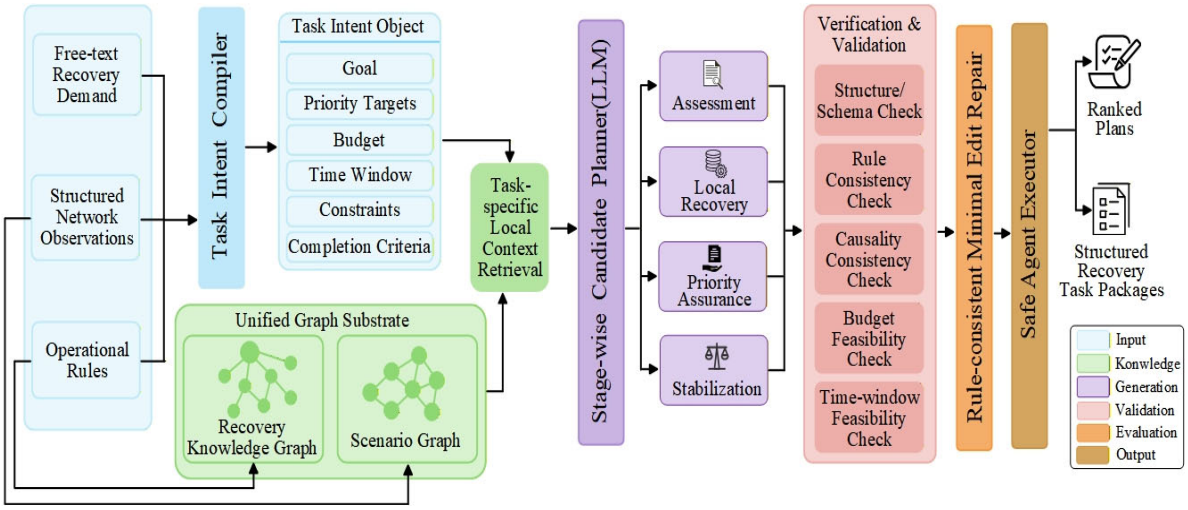

Figure 1. Overall architecture of ICG-Restore.

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.

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.

 doi:10.20944/preprints202604.1634.v1

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.

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.

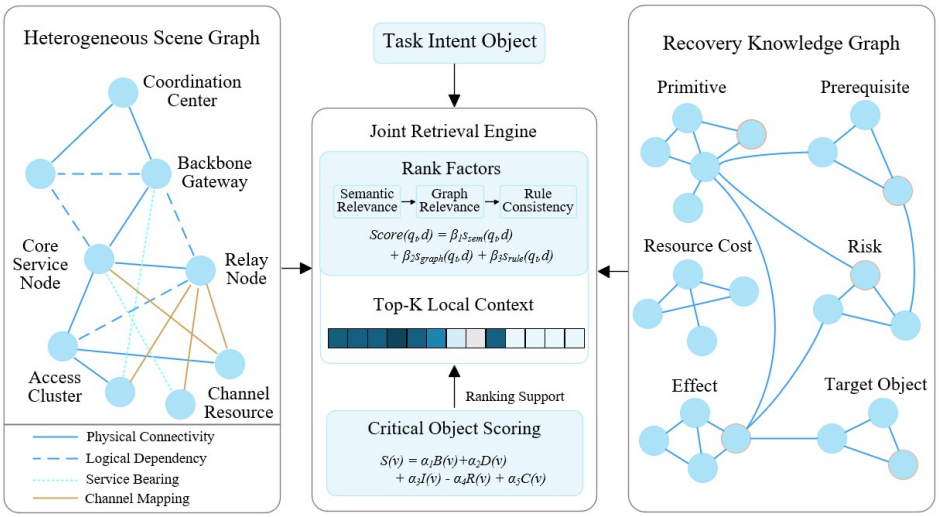

Figure 2. Unified graph substrate and task-relevant local structural context retrieval.

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.

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

 doi:10.20944/preprints202604.1634.v1

context that is most relevant to the current task and still acceptable under the budget and rule boundaries.

On this basis, we define a structural criticality score for object v.

$$S(v) = \alpha_1 B(v) + \alpha_2 D(v) + \alpha_3 I(v) - \alpha_4 R(v) + \alpha_5 C(v) \tag{10}$$

where $B(v)$ is betweenness centrality, $D(v)$ is service dependency strength, $I(v)$ is business importance, $R(v)$ is redundancy, and $C(v)$ is cross-layer coupling strength. This score is not used to replace the planner. Rather, it supports the construction of structural reference sets, retrieval ranking, and later structural-alignment evaluation.

The joint retrieval score is defined as

$$Score(q_t, d) = \beta_1 s_{sem}(q_t, d) + \beta_2 s_{graph}(q_t, d) + \beta_3 s_{rule}(q_t, d) \tag{11}$$

where s_{sem} measures semantic similarity, s_{graph} graph adjacency and path relevance, and s_{rule} consistency with budget, targets, and time windows. The top-K_r items are then selected to form the local context K_t for stage-wise 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

$$\phi_h = LLM(I_t, G_t^{local}, K_t, \phi_{1:h-1}) \tag{12}$$

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.

 doi:10.20944/preprints202604.1634.v1

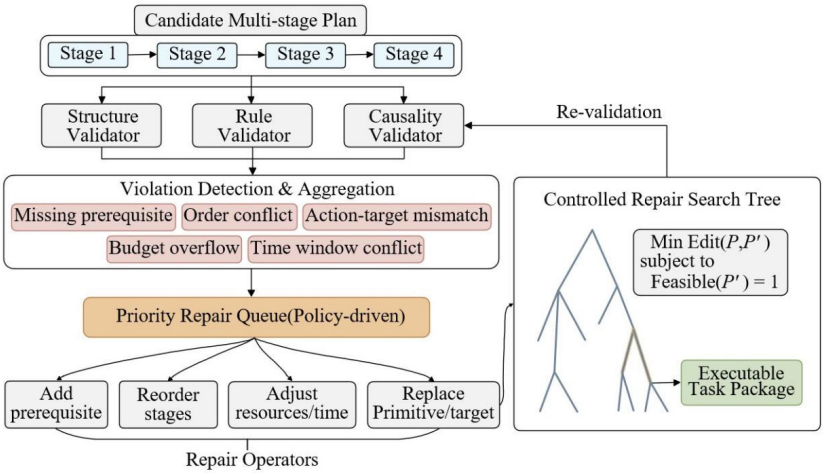

Figure 3. Illustration of rule-consistent minimal-edit repair.

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 causal 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.

Formally, the repair objective is

$$\Pi^{*}_{repair} = arg\min_{\Pi'} \text{Edit}(\Pi, \Pi') \text{ s.t. Feasible}(\Pi') = 1 \tag{13}$$

where the edit cost is defined as

$$\text{Edit}(\Pi, \Pi') = \lambda_{ins}N_{ins} + \lambda_{ord}N_{ord} + \lambda_{res}\Delta B + \lambda_{time}\Delta W + \lambda_{rep}N_{rep} \tag{14}$$

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;

 doi:10.20944/preprints202604.1634.v1

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.

In short, the repair module is not an unconstrained "second generator". It is a finite edit system centered on explicit violation types. Its purpose is not to find any new feasible plan, but to turn a candidate that is semantically reasonable yet structurally invalid into a restoration task package that is structurally executable, rule-compliant, and causally coherent, while preserving as much of the original stage backbone and resource commitment as possible.

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.

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 → validate → locally repair → 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.

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.

```
It ← CompileIntent(Dt, Ot, Rt)
(Gt, Kt) ← JointRetrieve(It, G, K)
Cfeas ← empty
for i = 1 to N do
    Π^(0) ← StagewiseGenerate(MLLM, It, Gt, Kt)
    (Π, V(Π), m) ← MinimalEditRepair(Π^(0), It, {Vstr, Vrule, Vcau}, M, τ)
    if V(Π) = empty then
        Cfeas ← Cfeas ∪ {Π}
    end if
end for
if Cfeas = empty  then
    return (empty, empty)
```

 doi:10.20944/preprints202604.1634.v1

13: end if
14: $S \leftarrow$ BatchEvaluateAndScore(E, C_{feas})
15: (Π^*, A_k) $\leftarrow$ RankAndSelect(C_{feas}, S, k)
16: return (Π^*, A_k)

Algorithm 2. Minimal-edit 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.
$\Pi \leftarrow \Pi^{(0)}$
$m \leftarrow 0$
$V(\Pi) \leftarrow$ ValidateAll(Π, V_{str}, V_{rule}, V_{cau})
while $V(\Pi) \neq$ empty do
    if $m \geq M$ then
        break
    end if
    if Deviation(Π, $\Pi^{(0)}$) > τ then
        break
    end if
    $e \leftarrow$ SelectEditByPriority($V(\Pi)$)
    $\Pi \leftarrow$ ApplyLocalEdit(Π, e)
    $m \leftarrow m + 1$
    $V(\Pi) \leftarrow$ ValidateAll(Π, V_{str}, V_{rule}, V_{cau})
end while
return (Π, $V(\Pi)$, m)

2. Experimental Platform and Design

5.1. Experimental Objectives and Research Questions

This section systematically evaluates whether ICG-Restore can stably generate restoration task packages that are executable, structurally coherent, and beneficial under complex constraints, dynamic environments, and heterogeneous inputs. 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?

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 test the method under different scales and structural complexities, we construct three benchmark emergency communication topologies: EC-S, EC-M, and EC-L, as summarized in Table 1. All three topologies share the same multi-layer heterogeneous semantic design and include

 doi:10.20944/preprints202604.1634.v1

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 channel scarcity.

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

Table 1. Benchmark topology scales and structural composition.

Topology	Coordination Centers	Backbone Gateways	Core Service Nodes	Relay Nodes	Frontier Access Nodes	Channel-Resource Nodes	Total Nodes	Structural Characterist
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

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

To evaluate robustness under dynamic conditions, each test instance is assigned one of five environmental evolution modes, shown in Table 2. M1: stable recovery after initial damage, with no additional failures; M2: secondary failures in local components during Stages 2–3; M3: congestion escalation caused by increasing service demand and intensified channel conflict; M4: delayed observation, where structured observations lag by 1–2 stages and are corrupted by noise; and M5: regional isolation drift, where local reachability fluctuates and coordination delay increases.

These modes are not intended to replicate low-level protocol behavior. Instead, they abstract the most common forms of uncertainty faced by high-level restoration planners.

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

 doi:10.20944/preprints202604.1634.v1

channel reallocation mitigates conflict and congestion. This shared execution environment provides a common basis for fair comparison.

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

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 [43], 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 [10]. 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 [44] directly feeds natural-language requirements and observations into an LLM and asks it to output a stage plan. PS-LLM (Plan-and-Solve) [18] first generates an explicit subtask plan and then solves it. GraphRAG-LLM [34] 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.

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

 doi:10.20944/preprints202604.1634.v1

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.

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	0.2
Language	Python 3.11.15	max_tokens	3072
cuda	13.1	text_top_k	8

5.6. Evaluation Metrics

We evaluate the proposed method from four aspects: executability, alignment with critical targets, integrated restoration effectiveness, and computational efficiency.

First, we use the Constraint Satisfaction Rate (CSR) to measure the proportion of final plans that satisfy all constraints.

$$\mathrm{CSR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\Pi_i \; is \; feasible\right] \tag{15}$$

where 1[·] 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, we use the Weighted Critical Target Coverage (WCTC@k) to evaluate how well the selected targets align with the graph-enhanced structural reference set.

$$\mathrm{WCTC@k} = \frac{1}{N}\sum_{i=1}^{N} \frac{\sum_{v \in P_i \cap T_i^{(k)}} S_i(v)}{\sum_{v \in T_i} S_i(v)} \tag{16}$$

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 truly important restoration objects.

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

$$\mathrm{CRS} = \alpha R_{path} + \beta R_{serv} + \gamma R_{cov} - \delta C - \rho Risk + \eta Rob \tag{17}$$

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 results of all methods on the three topology scales, while Figure 4 provides a visual summary. Overall, ICG-Restore achieves the highest WCTC@5 and CRS across all three topology scales and the highest CSR among the LLM-based baselines. This indicates that its advantage is not confined to a single metric, but extends simultaneously to critical-target selection, stage organization, and integrated restoration utility.

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

 doi:10.20944/preprints202604.1634.v1

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±0.00174	0.2574±0.0134	1.000	0.1023±0.00141	0.2125±0.0242	1.000	0.0823±0.0105	0.1742±0.0358
Direct-LLM	0.9311±0.00304	0.2829±0.00218	0.4503±0.00629	0.9212±0.00246	0.3096±0.00108	0.3307±0.00413	0.9011±0.00262	0.1576±0.0121	0.2428±0.0220
PS-LLM	0.9478±0.00211	0.3248±0.00256	0.4859±0.00410	0.9156±0.00247	0.3447±0.00216	0.3564±0.00338	0.9032±0.00305	0.1840±0.0107	0.2945±0.0217
GraphRAG-LLM	0.9412±0.00319	0.3669±0.00227	0.5220±0.00256	0.9297±0.00234	0.3398±0.00208	0.3733±0.00341	0.9111±0.00255	0.2068±0.0134	0.2752±0.0309
ICG-Restore	0.9536±0.00213	0.4069±0.00151	0.5514±0.00249	0.9345±0.00284	0.3814±0.00188	0.4112±0.00251	0.9202±0.00303	0.2534±0.0144	0.3126±0.0234

Averaged over the three topologies, ICG-Restore obtains 0.9361 CSR, 0.3472 WCTC@5, and 0.4251 CRS. Relative to Direct-LLM, these values improve 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 resource competition intensifies, the value of task-intent constraints and minimal-edit repair becomes increasingly important. In other words, the more complex the topology, the closer pure retrieval-enhanced generation gets to its ceiling, whereas the full loop of graph enhancement + rule-consistent repair continues to pay dividends.

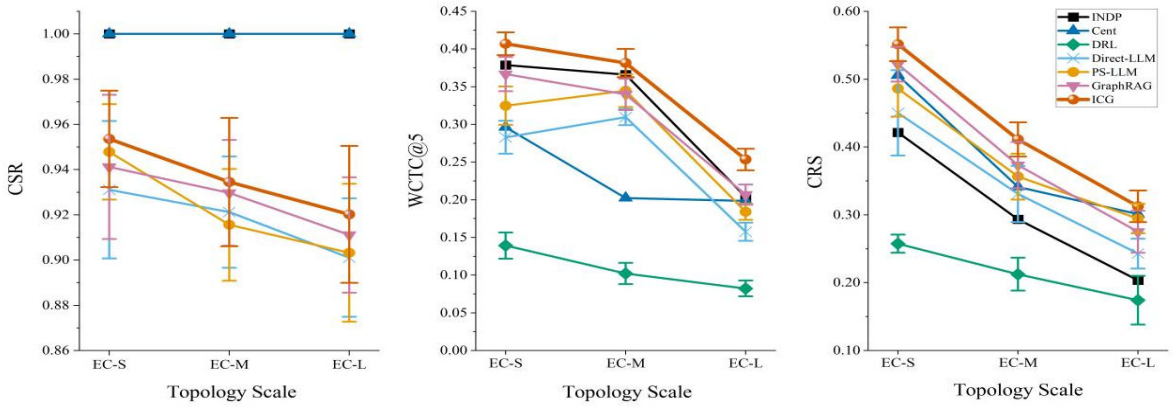

Figure 4. Main experimental results across three topology scales (EC-S, EC-M, and EC-L). Panels report CSR, WCTC@5, and CRS, respectively. Error bars denote mean ± standard deviation over five random seeds.

 doi:10.20944/preprints202604.1634.v1

Deterministic baselines use zero-variance placeholders for visual consistency. The proposed ICG-Restore is highlighted in orange.

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 narrows the executability gap between open-ended LLM planning and classical closed-form restoration methods, while outperforming them in critical-target coverage and integrated recovery quality. GraphRAG-LLM already demonstrates that graph enhancement is useful, but ICG-Restore further shows that graph retrieval alone is not enough. The decisive factor is the closed loop of task-intent constraints → graph-enhanced stage-wise generation → rule-consistent repair → unified execution evaluation.

5.8. Ablation Study

To examine the contribution of each core module, we perform ablations on the most challenging topology, EC-L. Three variants are considered: w/o Intent Compilation, w/o Graph Retrieval, and Repair → Regenerate, in which minimal-edit repair is replaced by "regenerate after validation failure". Exact results are given in Table 5, and relative shifts are visualized in Figure 5.

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

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

 doi:10.20944/preprints202604.1634.v1

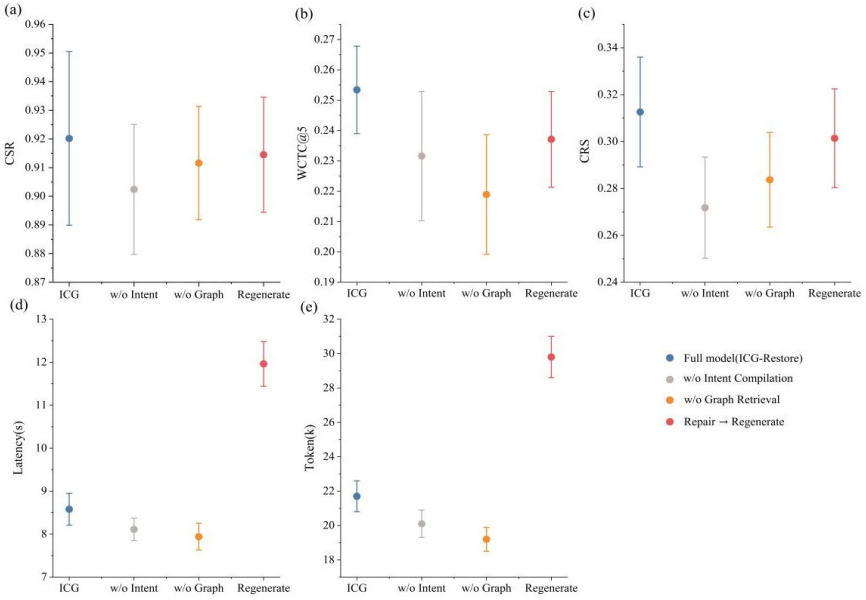

Figure 5. Ablation results on EC-L. Each panel reports one metric. Error bars denote mean ± standard deviation over five random seeds. The dashed horizontal line indicates the corresponding full-model performance.

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 21.7k to 29.8k, 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 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 Robustness

To examine whether the framework's benefit depends on a particular LLM backbone, we further test four backbone models: gpt-oss:20b, qwen3:14b, gemma3:27b, and mistral-small3.1:24b. The results are reported in Table 6.

Table 6. Cross-backbone robustness results (mean over all topology–task–mode–instance settings).

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

 doi:10.20944/preprints202604.1634.v1

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.

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 19.7k), 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.

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	Stage 2: Core service migration and priority-service assurance

 doi:10.20944/preprints202604.1634.v1

		"critical-link recovery" stage	
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 repair, the plan becomes a coherent recovery sequence: situation assessment and critical-link recovery → core service migration and priority-service assurance → local coverage reinforcement and channel coordination → stabilization and fallback monitoring. Importantly, this result is not a full rewrite, but a low-disturbance correction that preserves the original service-priority backbone while restoring structural executability, rule compliance, and causal consistency.

The repaired task package passes structural, rule-based, and causal validation and then enters the common agent-based execution evaluation. Compared with the unrepaired candidate, it achieves better service continuity and regional coverage while reducing coordination delay and operational risk. This suggests that minimal-edit repair improves not only formal feasibility, but also execution quality.

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.

Overall, this case confirms that the effectiveness of ICG-Restore arises from the closed-loop interaction of task-intent compilation, graph-enhanced retrieval, stage-wise generation, minimal-edit repair, and agent-based execution assessment. This integrated process enables the framework to move beyond plans that are merely plausible in language and toward task packages that are executable, evaluable, and directly usable by downstream systems.

6. Discussion

6.1. Why the Method Works

The effectiveness of ICG-Restore does not stem from a single isolated trick. Rather, it arises from an information loop formed by multiple mutually reinforcing modules. Task-intent compilation compresses free-text requirements into a stable structured constraint object and reduces input noise and semantic drift at the source. Graph-enhanced retrieval then narrows the open-ended knowledge space into task-relevant local structural context, so the LLM no longer plans "from impression" but instead organizes stage-wise actions around critical objects, dependency relations, and feasible restoration primitives. Minimal-edit repair converts semantically plausible yet structurally invalid candidates into executable plans while preserving as much of the original backbone as possible, thereby avoiding the instability of repeated full regeneration. Finally, the safety-aware agent executor provides a common external criterion rooted in actual restoration utility rather than textual fluency.

In other words, the key advantage of the method is not that it makes the LLM "write a more plan-like paragraph". The real contribution is that it enables the LLM to generate task packages that downstream systems can actually use, under explicit constraints, structural support, and executability feedback.

6.2. Relation to Existing Paradigms

 doi:10.20944/preprints202604.1634.v1

Compared with classical symbolic and hierarchical planning, ICG-Restore inherits the emphasis on executability and interpretability, but it does not require all inputs to be fully formalized from the start. It is specifically designed to handle the transition from mixed text-and-structure input to machine-consumable structured task packages, which makes it better suited to realistic high-level restoration settings.

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 the plan-repair literature, our contribution lies in integrating the idea ofrule-consistent minimal-edit repair into a graph-enhanced LLM planning framework and making repair a first-class component rather than a post hoc patch. In this framework, repair becomes the bridge from semantic plausibility to structural reliability.

6.3. Limitations and Threats to Validity

Despite the encouraging results, several limitations remain.

First, the experiments are conducted on self-constructed abstract topologies and a safety-aware agent executor, rather than on real communication engineering systems. The conclusions therefore support the effectiveness of the method at the high-level planning layer, not direct replacement of real operational workflows.

Second, the structural reference set used in WCTC@5 ultimately depends on a designed structural scoring model. Although the scoring model is external to the planner, it is still a design choice rather than an absolute ground truth. WCTC@5 should therefore be interpreted as a relative structural-alignment metric, not the sole criterion of correctness.

Third, the absolute performance level still depends to some extent on the LLM backbone. Although the cross-backbone experiments show stable relative trends, different models differ in structured-output stability, long-horizon planning ability, and constraint adherence. The contribution of this paper should therefore be understood as a framework contribution, not a claim about one specific model's absolute capability.

Fourth, the current evaluation setting mainly studies offline high-level planning by a single planner. It does not yet cover more complex settings such as multi-operator collaboration, human–AI co-decision, or continuous online replanning.

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

 doi:10.20944/preprints202604.1634.v1

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 high-constraint, high-uncertainty, and time-sensitive planning for post-disaster emergency communication service recovery. The central challenge is how to generate high-level restoration plans that are executable, interpretable, and evaluable from heterogeneous inputs, including free-text restoration requests, structured network observations, and operational rule boundaries.

To this end, we proposed ICG-Restore, a unified framework that integrates task-intent compilation, joint retrieval over a heterogeneous scenario graph and a restoration knowledge graph, stage-wise candidate generation, rule-consistent minimal-edit repair, and safety-aware agent-based execution evaluation. The framework establishes a complete planning chain from natural-language request input to structured task package output.

The empirical evidence from the main experiments, ablation study, cross-backbone robustness analysis, and case study consistently shows that ICG-Restore improves executability, structural alignment, and integrated restoration quality under complex constraints, while maintaining a reasonable balance between effectiveness and computational cost.

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 downstream systems can genuinely consume. This perspective may also be useful for related problems in critical infrastructure recovery, complex system scheduling, and multi-stage intelligent decision making.

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., G.S., Z.N. and T.N.; formal analysis, J.B., X.W. and G.S.; investigation, J.B., G.S., J.Z. and K.K.; resources, W.Z. and G.S.; 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 G.S. 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.

Acknowledgments: In this study, the authors used the large language model for the experiments reported in this manuscript. The authors take full responsibility for the content of this publication.

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

References

1. Matracia, M.; Saeed, N.; Kishk, M.A.; et al. Post-disaster communications: Enabling technologies, architectures, and open challenges. IEEE Open Journal of the Communications Society 2022, 3, 1177–1205. DOI: 10.1109/OJCOMS.2022.3192040.
2. Wang, Q.; Li, W.; Yu, Z.; et al. An overview of emergency communication networks. Remote Sensing 2023, 15, 1595. DOI: 10.3390/rs15061595.

 doi:10.20944/preprints202604.1634.v1

3. 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. DOI: 10.3390/electronics9091524.
4. Vittal, V.; Nair, N.; Rahmatian, F. Smart city energy technology in the face of emergency situations: Electric supply, electric transportation, and communication. IEEE Power and Energy Magazine 2022, 20, 16–25. DOI: 10.1109/MPE.2022.3184058.
5. Okeukwu-Ogbonnaya, A.; Amariucai, G.; Natarajan, B.; et al. Towards quantifying the communication aspect of resilience in disaster-prone communities. Scientific Reports 2024, 14, 8837. DOI: 10.1038/s41598-024-59192-3.
6. Sathurshan, M.; Saja, A.; Thamboo, J.; et al. Resilience of critical infrastructure systems: A systematic literature review of measurement frameworks. Infrastructures 2022, 7, 67. DOI: 10.3390/infrastructures7050067.
7. Wang, Y.; Zhao, O.; Zhang, L. Multiplex networks in resilience modeling of critical infrastructure systems: A systematic review. Reliability Engineering & System Safety 2024, 250, 110300. DOI: 10.1016/j.ress.2024.110300.
8. Tong, W.; Li, H.X.; Nasirzadeh, F.; et al. Resilience of interdependent infrastructure networks: Review and future directions. International Journal of Critical Infrastructure Protection 2025, 51, 100793. DOI: 10.1016/j.ijcip.2025.100793.
9. Huang, X.; Wang, N. Post-disaster restoration planning of interdependent infrastructure systems: A framework to balance social and economic impacts. Structural Safety 2024, 107, 102408. DOI: 10.1016/j.strusafe.2023.102408.
10. Liang, H.; Moya, B.; Chinesta, F.; et al. Resilience-based post-disaster recovery optimization for infrastructure systems via deep reinforcement learning. Reliability Engineering & System Safety 2026, 265, 111478. DOI: 10.1016/j.ress.2025.111478.
11. González, A.D.; Dueñas-Osorio, L.; Sánchez-Silva, M.; et al. The interdependent network design problem for optimal infrastructure system restoration. Computer-Aided Civil and Infrastructure Engineering 2016, 31, 334–350. DOI: 10.1111/mice.12171.
12. Bartolini, N.; Ciavarella, S.; et al. On critical service recovery after massive network failures. IEEE/ACM Transactions on Networking 2017, 25, 2235–2249. DOI: 10.1109/TNET.2017.2688330.
13. Arrigoni, V.; Prata, M.; Bartolini, N. Recovering critical service after large-scale failures with Bayesian network tomography. IEEE/ACM Transactions on Networking 2024, 32, 1–16. DOI: 10.1109/TNET.2024.3454478.
14. Basnayake, V.; Mabed, H.; Jayakody, D.N.K.; et al. Adaptive emergency call service for disaster management. Journal of Sensor and Actuator Networks 2022, 11, 83. DOI: 10.3390/jsan11040083.
15. Zhou, Y.; Luo, X.; et al. Survey on intelligent planning methods from a large language model perspective. Journal of System Simulation 2025, 37, 823–844. DOI: 10.16182/j.issn1004731x.joss.23-1468.
16. 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. DOI: 10.1109/ACCESS.2025.3552225.
17. Tantakoun, M.; Muise, C.; et al. 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), 2025.
18. Wang, L.; Xu, Z.; et al. 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), 2023.
19. Yao, S.; Yu, D.; Zhao, J.; et al. Tree of thoughts: Deliberate problem solving with large language models. In Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), 2023.
20. Yao, S.; Zhao, J.; et al. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), 2023.
21. Hao, S.; Gu, Y.; et al. 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), 2023.

 doi:10.20944/preprints202604.1634.v1

22. Töberg, J.-P.; Frese, U.; et al. Generation of robot manipulation plans using generative large language models. International Journal of Semantic Computing 2025, 19, 79–103. DOI: 10.1142/S1793351X25410041.
23. Almoghathawi, Y.; Barker, K.; Albert, L.A. Resilience-driven restoration model for interdependent infrastructure networks. Reliability Engineering & System Safety 2019, 185, 12–23. DOI: 10.1016/j.ress.2018.12.006.
24. Alkhaleel, B.A.; Liao, X.; et al. Model and solution method for mean-risk cost-based post-disruption restoration of interdependent critical infrastructure networks. Computers and Operations Research 2022, 144, 105812. DOI: 10.1016/j.cor.2022.105812.
25. Li, Y.; Zhang, Z.; et al. Joint optimization of workforce scheduling and routing for restoring a disrupted critical infrastructure. Reliability Engineering & System Safety 2019, 191, 106551. DOI: 10.1016/j.ress.2019.106551.
26. Zhang, Z.; Ji, Y.; et al. Dynamic emergency inspection routing and restoration scheduling to enhance the post-earthquake resilience of a highway–bridge network. Reliability Engineering & System Safety 2022, 220, 108282. DOI: 10.1016/j.ress.2021.108282.
27. López-Villegas, I.; Martínez-Rios, E.A.; Izquierdo-Reyes, J.; et al. A systematic literature review of emergency communications assisted by unnamed aerial vehicles. Ad Hoc Networks 2026, 182, 104063. DOI: 10.1016/j.adhoc.2025.104063.
28. Gu, X.; Zhang, G. A survey on UAV-assisted wireless communications: Recent advances and future trends. Computer Communications 2023, 208, 44–78. DOI: 10.1016/j.comcom.2023.05.013.
29. Luo, J.; Wang, Z.; Xia, M.; et al. Path planning for UAV communication networks: Related technologies, solutions, and opportunities. ACM Computing Surveys 2023, 55, 1–36. DOI: 10.1145/3560261.
30. Li, L.; Zhu, L.; Huang, F.; et al. Post-disaster emergency communications enhanced by drones and non-orthogonal multiple access: Three-dimensional deployment optimization and spectrum allocation. Drones 2024, 8, 63. DOI: 10.3390/drones8020063.
31. Zhang, S.; Shi, Y.; et al. Trajectory planning in UAV emergency networks with potential underlying D2D communication based on K-means. EURASIP Journal on Wireless Communications and Networking 2021, 2021, 10. DOI: 10.1186/s13638-021-01987-3.
32. Bai, J.; Zhu, W.; Liu, S.; Xu, L.; Wang, X. Path planning method for unmanned vehicles in complex off-road environments based on an improved A* algorithm. Sustainability 2025, 17, 4805. DOI: 10.3390/su17114805.
33. Procko, T.T.; Ochoa, J.; et al. Graph retrieval-augmented generation for large language models: A survey. In Proceedings of the 2024 Conference on AI, Science, Engineering, and Technology (AIxSET 2024), 2024.
34. Li, C.; Song, Y.; et al. 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), 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. Machine Learning and Knowledge Extraction 2026, 8, 75. DOI: 10.3390/make8030075.
36. Liu, B.; Fang, Y.; Xu, N.; et al. Large language models for knowledge graph embedding: A survey. Mathematics 2025, 13, 2244. DOI: 10.3390/math13142244.
37. You, Y.; Liu, Z.; Wen, X.; et al. Large language models meet graph neural networks: A perspective of graph mining. Mathematics 2025, 13, 1147. DOI: 10.3390/math13071147.
38. Zhang, D.; Liu, Y.; Zhao, J.; Xu, C. TransGoT: Structured graph-of-thoughts reasoning for machine translation with large language models. Big Data and Cognitive Computing 2026, 10, 70. DOI: 10.3390/bdcc10030070.
39. Babli, M.; Sapena, Ó.; Onaindia, E. Plan commitment: Replanning versus plan repair. Engineering Applications of Artificial Intelligence 2023, 123, 106275. DOI: 10.1016/j.engappai.2023.106275.
40. Zaidins, P.; Goldman, R.; et al. 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), 2025.
41. Jiang, X.; Dong, Y.; et al. Self-planning code generation with large language models. ACM Transactions on Software Engineering and Methodology 2024, 33, 1–30. DOI: 10.1145/3672456.

 doi:10.20944/preprints202604.1634.v1

42. Zhang, S.; Chen, Y.; et al. Planning with large language models for code generation. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), 2023.
43. Ulusan, A.; Ergun, Ö.; et al. Restoration of services in disrupted infrastructure systems: A network science approach. PLoS ONE 2018, 13, 1–28. DOI: 10.1371/journal.pone.0192272.
44. Olaiya, K.; Delnevo, G.; Lam, C.-T.; et al. From prompts to paths: Large language models for zero-shot planning in unmanned ground vehicle simulation. Drones 2025, 9, 875. DOI: 10.3390/drones9120875.