SOLVER SUCCESS IS NOT ENOUGH: VERIFICATION-GROUNDED LLM AGENTS FOR DISTRIBUTION-GRID MODEL RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Updating a distribution-grid optimization model from a natural-language operational request is not a code-generation task alone. A single request may jointly change device states, parameters, constraints, objectives, and priorities, creating dependencies across interpretation, formulation, implementation, and execution. A candidate may execute and solve successfully while misrepresenting the request or producing a solution that violates nonlinear AC power-flow constraints. Existing evaluations of LLM agents do not jointly test such coupled modifications against both requirement fidelity and physical feasibility. We introduce a benchmark with 36 request types spanning parameter edits, formulation changes, and coupled state-and-formulation changes. To evaluate the same reconstruction tasks across different network structures, we instantiate all 36 request types under four operating conditions on three distribution networks, yielding 432 cross-network instances, each paired with executable requirement and physical checks. We also propose GridRecon, a verification-grounded multi-agent framework in which Planning, Formulation, Coding, and Review Agents exchange structured requirement specifications and mathematical change plans. Deterministic tools check requirement implementation, program execution, topology, AC power flow, and SOC relaxation quality, while a controller uses the resulting evidence to accept verified candidates, trigger bounded, targeted self-refinement for actionable failures, or reject unresolved cases. As a result, our GridRecon outperforms the strongest adapted baseline for each of four LLM backbones on IEEE 69 by 7.6–34.0 percentage points and passes 329 of 432 instances in the common cross-network evaluation on IEEE 33, IEEE 69, and case136ma.

1 INTRODUCTION

Distribution grids face changing renewable availability, device states, operating limits, and control objectives. Online optimization addresses decisions under time-varying conditions (Zhou et al., 2018), while changes to the decision problem itself require expertise in electrical engineering, mathematical formulation, and programming (Yang et al., 2026). As shown in Figure 1, we study the resulting *distribution-grid model reconstruction* task: given an existing optimization program, its current operating state, and a natural-language operational request, the goal is to produce an updated program and an acceptable and effective operating solution.

The first challenge is the coupled scope of the requested change. A single request may jointly alter device availability, numerical parameters, constraints, objectives, and priorities, creating dependencies across interpretation, formulation, implementation, and execution. Repository-level modifications already involve dependencies across code locations (Jimenez et al., 2024); optimization models additionally encode device scope, boundary conditions, and objective priorities. Taking a device out of service affects both its availability and output bounds, while introducing an objective must preserve existing priorities. An omission can therefore change the problem without causing an execution error. We name this corresponding acceptance condition **requirement fidelity**: the reconstructed state and program must preserve the requested entities, values, relationships, and priorities.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

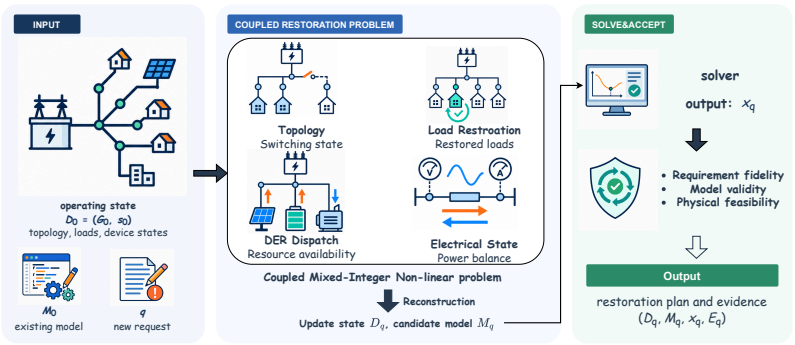


Figure 1: Distribution-grid model reconstruction. A request changes an existing model and operating state. A candidate is accepted only after independent checks of requirement fidelity, model validity, and nonlinear physical feasibility. Solver success alone is insufficient.

Requirement fidelity alone is insufficient because discrete switching decisions interact with nonlinear power flows. Second-order cone relaxations make branch-flow optimization tractable (Farivar & Low, 2013), but their exactness depends on the network and operating conditions (Gan et al., 2015). A candidate can execute successfully and satisfy its relaxed formulation yet violate voltage or current limits under AC power flow. Conversely, a physically feasible operating point does not establish that the request was implemented. This motivates a second acceptance condition, **physical feasibility**. Program execution and solver success are therefore necessary but insufficient for distribution-grid model reconstruction.

Tool-using language agents combine reasoning with execution feedback (Yao et al., 2023; Wang et al., 2024), and optimization-modeling systems generate and debug mathematical programs or modify existing code for what-if analysis (Ahmaditeshnizi et al., 2024; Li et al., 2023). Power-system applications extend these capabilities to coordinated dispatch modeling (Yang et al., 2026) and professional evaluation of parameter adjustments and structural extensions to optimal power flow (Shen et al., 2026). Specialized roles and feedback coordination further support complex workflows (Hong et al., 2024; Xiao et al., 2024; Li et al., 2026; Zhou et al., 2026). However, existing evaluations do not jointly test whether agents can handle coupled state-and-formulation changes, faithfully implement the request, and produce solutions that remain feasible under nonlinear grid physics. The difficulty of this reconstruction problem and the reliability of proposed agent frameworks therefore remain insufficiently measured.

In this paper, we first introduce a benchmark that makes these requirements measurable. It contains 36 request types: eight parameter-edit tasks, twelve formulation-change tasks, and sixteen coupled-change tasks. Four operating conditions vary initial topology, loading, and renewable availability, and every instance is paired with executable checks for requirement fidelity and physical feasibility. To compare the same reconstruction tasks across network structures, all 36 request types are instantiated under all four conditions on IEEE 33, IEEE 69, and case136ma, yielding 432 cross-network instances. The benchmark thus varies request complexity, operating condition, and network structure while retaining a common acceptance contract.

Reliable revision requires more than iteration alone: intrinsic self-correction without external feedback can degrade otherwise valid reasoning (Huang et al., 2024), whereas validation-in-the-loop power modeling demonstrates the value of solver and task checks (Hu et al., 2025). We propose **GridRecon**, a verification-grounded multi-agent framework that coordinates Planning, Formulation, Coding, and Review Agents through structured requirement specifications and mathematical change plans. Deterministic tools check realized state changes, program execution, topology, AC power flow, and SOC relaxation quality. A controller uses the resulting evidence to accept verified candidates, route actionable discrepancies to bounded, targeted self-refinement, or reject unresolved failures. Refinement is therefore evidence-triggered, issue-specific, and bounded rather than unconstrained self-correction. Across four LLM backbones on IEEE 69, GridRecon achieves the highest end-to-end success for every backbone, exceeding the strongest adapted baseline by 7.6–34.0 percentage points. With DeepSeek V4 Flash, it passes 329 of 432 instances in the common cross-network evaluation. Our main contributions are summarized as follows,

1. We formulate distribution-grid model reconstruction as the joint adaptation of an existing optimization program and operating state, and introduce a benchmark spanning 36 request types, four operating conditions, and a 432-instance cross-network evaluation with executable requirement and physical checks.
2. We propose GridRecon, a verification-grounded multi-agent framework that preserves cross-stage dependencies through structured handoffs and uses localized executable evidence for bounded, targeted self-refinement.
3. We establish an end-to-end evaluation contract separating requirement fidelity, numerical solvability, and physical feasibility, and demonstrate GridRecon's effectiveness across four LLM backbones and three distribution networks.

2 Related Work

Tool use and agent coordination. Combining language reasoning with external execution enables agents to advance complex tasks through environmental feedback. ReAct interleaves reasoning and actions, CodeAct represents tool operations as executable code, and SWE-agent develops interaction interfaces for repository-level program modification (Yao et al., 2023; Wang et al., 2024; Yang et al., 2024). As tasks span analysis, design, and implementation, research has increasingly explored coordination among specialized roles. MetaGPT organizes information exchange through standardized workflows, while ChatDev uses multi-role dialogue for software development (Hong et al., 2024; Qian et al., 2024). AgentFlow advances this direction through in-the-flow optimization, training a planning module within multi-turn interactions to adapt to evolving tool feedback (Li et al., 2026). Together, these studies highlight the importance of interaction interfaces, specialization, and feedback organization. GridRecon applies these principles to distribution-grid model reconstruction, coordinating formulation and implementation through structured requirements and using independent electrical calculations to inform candidate acceptance and revision.

LLM-driven optimization modeling. Automated optimization modeling connects natural-language requirements with mathematical formulations and solver programs. OptiMUS uses a modular workflow for mixed-integer linear programming formulation, code generation, and debugging, while Chain-of-Experts addresses complex operations research problems through dynamic expert selection and iterative reflection (Ahmaditeshnizi et al., 2024; Xiao et al., 2024). For existing models, OptiGuide translates what-if questions into optimization program modifications and generates answers from execution results (Li et al., 2023). More recently, EngiAgent targets open-ended engineering problems with a fully connected coordinator that routes feedback among analysis, modeling, verification, solving, and evaluation roles (Zhou et al., 2026). This line of work progressively extends the focus from executable program generation to formulation quality and engineering feasibility. We study the reconstruction of existing distribution-grid models under changing operational requirements, jointly examining how requirements are expressed in the model and whether candidate solutions are electrically feasible.

Power-system modeling and physical validation. Professional power-system modeling requires natural-language modifications to preserve both the intended optimization semantics and the electrical relationships of the network. Yang et al. (2026) coordinate information extraction, problem formulation, and code implementation for active distribution network dispatch, while Hu et al. (2025) combine LLM generation with validation and iterative correction for unit commitment. ProOPF benchmarks parameter adjustments and structural extensions to a canonical optimal power flow model, enabling systematic evaluation of domain-specific modeling capabilities (Shen et al., 2026). These efforts establish the value of specialized formulation and validation, but do not jointly examine coupled changes to operating states and model structure under both requirement and physical checks. In parallel, power-system optimization theory shows that the exactness of convex branch-flow relaxations depends on network and operating conditions (Farivar & Low, 2013; Gan et al., 2015). A solver-successful relaxed model can therefore still produce an operating point that violates the original nonlinear AC constraints. GridRecon connects these lines of work by integrating requirement and model-validity checks, topology and AC power-flow validation, and relaxation-quality diagnostics into agent coordination. The resulting evidence is used not only to assess final candidates, but also to localize failures and guide targeted revision.

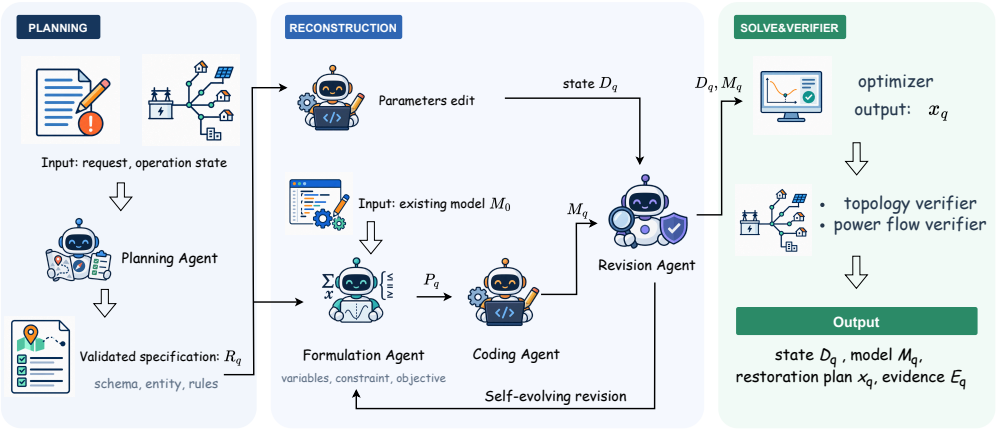


Figure 2: Overview of GridRecon. Planning grounds request q into a validated specification $\mathcal{R}_q$; parameter edits update $\mathcal{D}_q$ through the snapshot editor, while model extensions are translated into plan $\mathcal{P}_q$ and implemented as $\mathcal{M}_q$. Review compares the request, state edits, formulation plan, and implementation evidence, routing supported discrepancies to bounded revision. The reviewed candidate is then solved and independently checked for topology and nonlinear AC feasibility, yielding $(\mathcal{D}_q, \mathcal{M}_q, x_q)$ and validation evidence E_q.

3 METHOD

3.1 PROBLEM SETUP AND ACCEPTANCE CONTRACT

GridRecon organizes distribution-grid model reconstruction around two linked questions: how to preserve an operational request through mathematical formulation and implementation, and how to determine whether the resulting solution is electrically acceptable. Given a reference optimization model $\mathcal{M}_0$, its operating state $\mathcal{D}_0$, and a request q, the framework produces a candidate model $\mathcal{M}_q$, an updated state $\mathcal{D}_q$, a computed solution x_q, and verification evidence E_q. Our implementation uses single-period distribution-grid restoration with a mixed-integer second-order cone program (MISOCP); the complete formulation is given in Appendix H.

Let $C_q = (\mathcal{D}_q, \mathcal{M}_q, x_q)$. A successful reconstruction must satisfy requirement fidelity, numerical solvability, and nonlinear physical feasibility. With corresponding indicators V_{req}, V_{num}, and V_{phys}, the acceptance contract is

$$\text{Accept}(C_q) = V_{\text{req}}(C_q) \wedge V_{\text{num}}(C_q) \wedge V_{\text{phys}}(C_q). \tag{1}$$

Here, V_{req} checks the requested entities, values, relationships, and priorities in the realized state, model, and solution; V_{num} requires executable code and a valid solver result; and V_{phys} requires the mandatory topology and nonlinear AC checks. Thus, solver success establishes only one part of acceptance.

3.2 GRIDRECON ARCHITECTURE

A request such as changing renewable availability or adding a lower-priority objective involves several distinct decisions: identifying the affected devices, determining the mathematical consequences, implementing them, and checking the outcome. Resolving these decisions through repeated free-form descriptions can obscure dependencies and propagate omissions. We therefore organize four agent roles around explicit intermediate outputs (Figure 2). Planning establishes the requested changes; Formulation specifies their mathematical meaning; Coding implements them; Review compares the candidate with the request and observed implementation evidence. Each downstream stage retains access to the original request, providing a reference when intermediate interpretations disagree.

Three groups of tools support these roles: *knowledge and specification tools* provide modeling examples and validate structured requirements; *execution and testing tools* apply operating-state changes,

Table 1: Agent handoffs make decisions inspectable before subsequent implementation or revision.

Role	Decision to make explicit	Information passed onward
Planning	Modification scope and value sources	Grounded operations and priorities
Formulation	Objective order and constraint dependencies	Mathematical changes and expected behavior
Coding	Realization of the intended changes	Candidate code and execution records
Review	Whether a correction is justified	Findings with supporting evidence

support code editing, and run optimization programs; *electrical validation tools* examine topology, AC power flow, and SOC relaxation quality. A controller coordinates tool calls, revision budgets, and checkpoints. Agents interpret requirements and propose changes; tools provide repeatable calculations and inspection records that ground revision and final acceptance decisions.

3.3 Requirement-Preserving Model Reconstruction

An operational request often leaves dependencies implicit: increasing the current load requires a numerical reference, while a generator outage affects both availability and dispatch bounds. The Planning Agent records affected entities, operations, value sources, device scope, and objective priorities in a structured specification $\mathcal{R}_q$. Schema, entity, and consistency checks expose invalid fields, unresolved references, and conflicting operations before execution. The validated specification also determines the reconstruction route: supported operating-state edits are applied through existing tools, whereas requests involving new variables, objectives, or constraints proceed to formulation and coding. Explicit operation types and numerical sources distinguish, for example, a relative increase from an absolute assignment and an individual-device outage from a class-wide modification.

The Formulation Agent maps $\mathcal{R}_q$ to a mathematical plan $\mathcal{P}_q$ that specifies changed expressions, retained relationships, expected behavior, and targeted tests. Its context includes the reference formulation and retrieved modeling examples together with their applicability conditions. Separating formulation from coding makes objective order and constraint dependencies inspectable before implementation. For a request that increases load by 20% and appends a procurement-cost objective, $\mathcal{R}_q$ records the relative state change, while $\mathcal{P}_q$ defines the new expression and preserves the existing higher-priority objectives. This distinction matters because a weighted term and a separate priority level can produce different decisions even when both programs solve successfully.

The Coding Agent implements $\mathcal{P}_q$ using the original request, interface documentation, source navigation, and targeted tests. The candidate is executed in an isolated workspace, with state edits, source differences, test results, and execution records retained together. These artifacts connect mathematical intent to concrete program behavior and can expose missing quantities, incorrect field operations, or inconsistencies between declared and realized changes. Review therefore examines an actual candidate rather than relying on the code generator's description, while the original request remains available to detect requirements omitted by an earlier stage.

3.4 Verification-Grounded Targeted Self-Refinement

Successful execution establishes that a candidate can be evaluated, but a solvable program can still omit a requested objective or produce an electrically invalid operating state. GridRecon revises a candidate only when a check produces an actionable and localized discrepancy. Requirement-schema, entity, or rule failures return the invalid field and reason to Planning, while Coding self-check failures expose the failed command, test, or interface within the restricted workspace. Review compares q, $\mathcal{R}_q$, $\mathcal{P}_q$, realized state edits, source context, and directed-test records. A finding must identify the relevant requirement, expected behavior, observed implementation, direct evidence, and responsible downstream stage. For example, a requested bus-specific coefficient implemented as a shared constant supports a Coding revision; a preference for an alternative but mathematically equivalent implementation does not.

The controller selects an action

$$a_q \in \{\text{accept}, \text{revise}(s, f), \text{reject}\}, \qquad s \in \{\text{Planning}, \text{Formulation}, \text{Coding}\}, \tag{2}$$

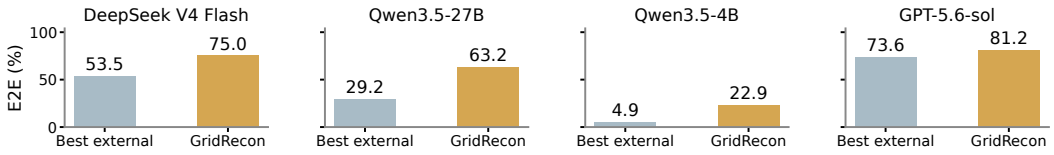


Figure 3: E2E success across backbones. Each pair compares GridRecon with the highest-performing external framework in Table 2.

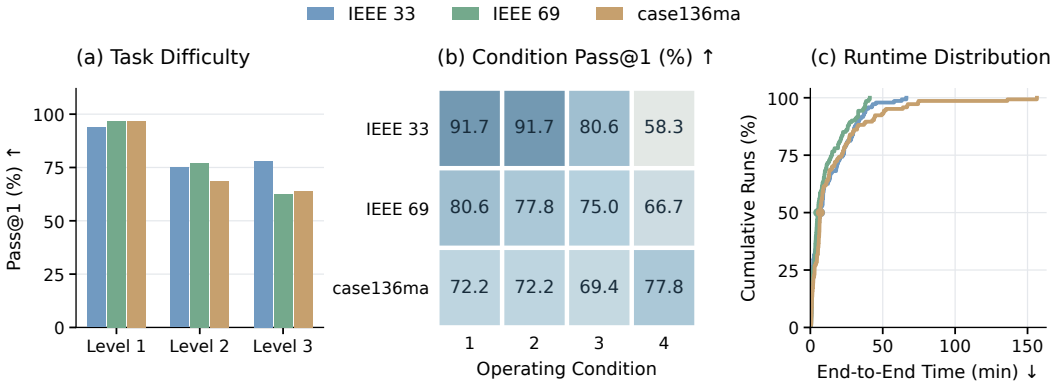


Figure 4: Cross-network evaluation of GridRecon with DeepSeek V4 Flash (432 instances). (a) Pass@1 by difficulty level, with 32/48/64 instances at Levels 1–3. (b) Pass@1 by initial condition, with 36 instances per cell. (c) Cumulative distribution of recorded end-to-end runtimes; dots mark medians. Overall success is 116/144, 108/144, and 105/144 for IEEE 33, IEEE 69, and case136ma, respectively.

where f is an evidence-supported finding and s is the stage responsible for resolving it. A revision reruns the affected stage and all dependent checks. The controller records handled findings and enforces revision and elapsed-time budgets; repeated findings, unsupported diagnoses, or exhausted budgets terminate with a recorded rejection. Self-refinement is therefore *targeted* because both the discrepancy and revision stage are explicit, and *bounded* because revision requires new evidence and an available budget.

After implementation review, the candidate is executed and solved to obtain x_q. Deterministic validation checks requested state and solution relations, fault isolation, connectivity, radiality, AC convergence, bus voltages, and line loading. SOC diagnostics examine the power–current relationship relaxed by the MISOCP, whose exactness depends on network and operating conditions (Farivar & Low, 2013; Gan et al., 2015); SOC quality remains a diagnostic distinct from the mandatory topology and AC checks. Physical-validation outcomes do not by themselves authorize speculative LLM edits: without a localized formulation or implementation conflict, they support acceptance or rejection rather than unconstrained revision. The final output retains $(\mathcal{D}_q, \mathcal{M}_q, x_q)$ together with E_q, making the basis for the decision inspectable. Role instructions, structured schemas, and execution controls are documented in Appendix C, and validation criteria in Appendix B.5.

4 Experiments

4.1 Evaluation Protocol

Implementation. Experiments are orchestrated on a workstation with an Intel Core i7-12650H CPU (10 cores, 16 threads) and 32 GB of RAM. GridRecon uses LangGraph 1.0.7 (LangChain, Inc., n.d.) to orchestrate agent interactions and persist execution state. The Coding Agent invokes the OpenCode CLI (OpenCode, n.d.) to edit and execute candidate programs. Gurobi 13.0.2 (Gurobi Optimization, LLC, 2026) solves the mixed-integer second-order cone programs, and pandapower 2.14.11 (Thurner et al., 2018) performs independent AC power-flow validation. All compared

Table 2: Experimental results on distribution-grid optimization reformulation across four LLM backbones. Phys. is the percentage of all instances passing solver, radial-topology, and AC power-flow checks jointly. Req. is the number of passed mandatory requirement checks divided by the total number of recorded mandatory requirement checks, pooled across instances (micro-average). E2E Pass@1 (gray, primary metric) is the percentage of all instances satisfying requirement implementation, program execution, optimization solving, and every mandatory validation check in one framework run, including its allowed revisions. Level 1–3 columns report E2E rates within parameter-edit, formulation-change, and coupled-change tasks. All rates are percentages; SOC relaxation quality is assessed separately. Time (seconds), Tokens (thousands of input plus output tokens), and Cost (USD, summed over model calls using input, cached-input, and output prices) are averaged over normally terminated runs with complete records, including unsuccessful runs; success rates use all instances. Prices and check criteria appear in Appendix B. Red/blue mark best/second-best results within each backbone.

	Performance (%)			Task Pass Rate (%)			Efficiency		
Method	Phys. ↑	Req. ↑	E2E ↑	Level 1 ↑	Level 2 ↑	Level 3 ↑	Time (s) ↓	Tokens (k) ↓	Cost ($) ↓
DeepSeek V4 Flash (0731)									
OpenCode	70.1	70.8	52.1	56.2	56.2	46.9	454	1343.1	0.0394
OptiGuide	72.9	77.3	39.6	43.8	47.9	31.2	**321**	92.5	0.0264
Chain-of-Experts	56.2	68.2	45.1	37.5	54.2	42.2	939	**58.0**	**0.0237**
EngiAgent	66.0	77.0	53.5	65.6	62.5	40.6	941	137.5	0.0697
GridRecon	**89.6**	**93.5**	**75.0**	**96.9**	**77.1**	**62.5**	611	817.6	0.0474
Qwen3.5-27B									
OpenCode	36.1	49.9	25.0	31.2	29.2	18.8	475	1247.9	0.1128
OptiGuide	77.8	71.0	29.2	46.9	39.6	12.5	**412**	123.4	0.0238
Chain-of-Experts	31.2	47.9	14.6	31.2	12.5	7.8	449	**74.6**	**0.0199**
EngiAgent	11.8	24.4	6.2	18.8	4.2	1.6	628	112.8	0.0307
GridRecon	**89.6**	**91.6**	**63.2**	**87.5**	**72.9**	**43.8**	837	777.1	0.0802
Qwen3.5-4B									
OpenCode	**68.8**	23.8	0.7	0.0	2.1	0.0	**248**	156.2	/
OptiGuide	17.4	27.8	4.9	6.2	**10.4**	0.0	263	275.0	/
Chain-of-Experts	6.9	13.8	3.5	3.1	6.2	1.6	385	**98.5**	/
EngiAgent	0.7	2.3	0.7	3.1	0.0	0.0	323	115.9	/
GridRecon	27.8	**57.8**	**22.9**	**65.6**	6.2	**14.1**	369	129.4	/
GPT-5.6-sol									
OpenCode	76.4	81.6	66.0	75.0	72.9	56.2	345	506.9	0.4751
OptiGuide	85.4	88.5	73.6	78.1	83.3	64.1	**230**	106.8	0.5415
Chain-of-Experts	64.6	75.9	54.9	62.5	62.5	45.3	294	**60.8**	**0.4158**
EngiAgent	50.7	65.6	42.4	43.8	60.4	28.1	664	148.6	0.9926
GridRecon	**88.9**	**93.6**	**81.2**	**96.9**	**87.5**	**68.8**	692	475.8	1.1742

frameworks use the same optimization and physical-validation tools. The benchmark comprises 36 single-snapshot request types: eight parameter-edit tasks (Level 1), twelve formulation-change tasks (Level 2), and sixteen coupled-change tasks (Level 3). They cover operating limits, restoration priorities, flexible and voltage-dependent loads, piecewise costs, generator commitment, and coordinated voltage/reactive control. Four conditions vary initial topology, loading, and renewable availability (Appendix A). Framework comparisons with all four backbones and ablations with DeepSeek V4 Flash and Qwen3.5-27B use 144 IEEE 69 instances per variant, distributed as 32/48/64 across Levels 1–3. The cross-network evaluation uses all 36 request types under four conditions on each of

three networks (432 instances). Methods share requirements, initial states, reference code, solver, and evaluation interface.

4.2 Framework Comparisons Across Backbones

Coordination Improves Completion Across Backbones. GridRecon achieves the highest E2E success in all four backbone groups (Table 2). With DeepSeek, it completes 108/144 instances versus 77/144 for EngiAgent, a gain of 21.5 percentage points. Its Level 3 success reaches 62.5%, versus the strongest external result of 46.9%. Gains over the strongest external framework are 34.0, 18.1, and 7.6 points with Qwen3.5-27B, Qwen3.5-4B, and GPT-5.6-sol. The advantage thus extends across backbone settings and coupled modeling changes.

Electrical Feasibility Does Not Establish Task Completion. With Qwen3.5-4B, OpenCode achieves 68.8% physical feasibility but 0.7% E2E success; GridRecon achieves 27.8% and 22.9%, respectively. A physically valid operating point can coexist with an unfulfilled modeling request. With DeepSeek, the feasibility-to-completion gap is 33.3 points for OptiGuide and 14.6 for GridRecon. GridRecon also attains the highest requirement accuracy in every backbone group. The joint metrics distinguish electrical validity from faithful implementation.

Structured Collaboration Balances Completion and Computational Effort. With DeepSeek, GridRecon improves completion while taking 611 seconds versus EngiAgent's 941 seconds and using 39.1% fewer tokens than OpenCode. Under Qwen3.5-27B, it uses 37.7% fewer tokens than OpenCode, with longer runtime (837 versus 475 seconds). With GPT-5.6-sol, it completes 11 more instances than OptiGuide at 692 versus 230 seconds and estimated costs of $1.1742 versus $0.5415.

4.3 Network Size and Operating Conditions

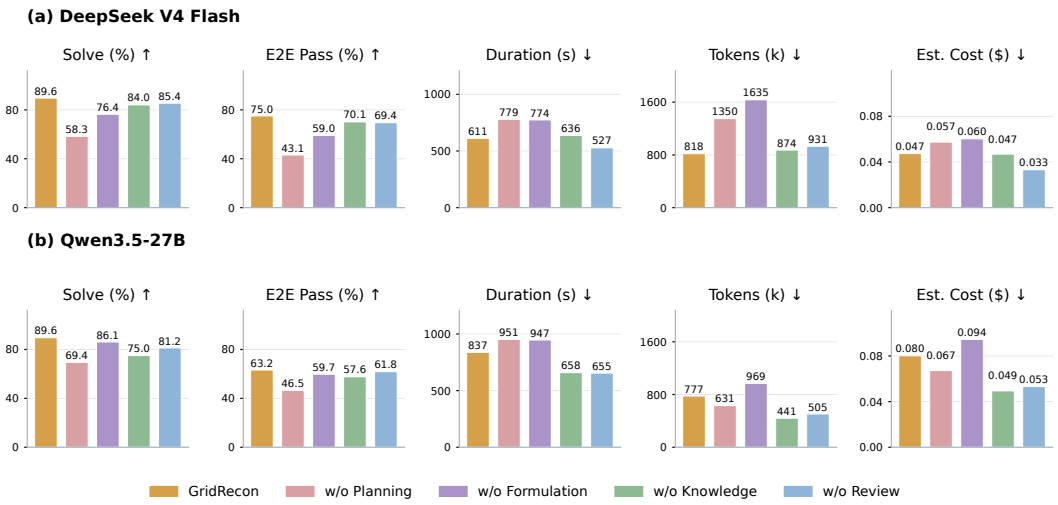
Cross-Network Success Reflects Both Task and Operating Context. GridRecon completes 329/432 cross-network instances, with overall success between 72.9% and 80.6% (Figure 4). Level 3 success varies from 78.1% on IEEE 33 to 62.5% on IEEE 69 and 64.1% on case136ma. While IEEE 33 achieves 91.7% success under Conditions 1–2, its performance falls to 80.6% and 58.3% under Conditions 3–4. This indicates that network size alone does not determine task difficulty. Median runtimes are 7.08, 5.09, and 6.88 minutes on IEEE 33, IEEE 69, and case136ma, respectively. Appendix G reports exact totals.

4.4 Component Contributions

Planning and Formulation Guide Different Modifications. Planning has its largest measured contribution on parameter-edit tasks. Removing it reduces Level 1 success from 96.9% to 15.6% with DeepSeek and from 87.5% to 6.2% with Qwen3.5-27B. This ablation removes specialized requirement interpretation and changes the downstream route, so the result measures their combined contribution. Formulation provides a different benefit: without it, DeepSeek success falls from 77.1% to 56.2% at Level 2 and from 62.5% to 43.8% at Level 3, while mean tokens increase from 817.6k to 1,635.4k. Explicit mathematical guidance is therefore particularly useful for these structural changes. With Qwen3.5-27B, removing Formulation changes overall success from 63.2% to 59.7%; Level 3 decreases to 37.5%, whereas Level 2 increases to 75.0%. Its aggregate benefit is supported across both backbones, with different task-level responses.

Retrieval Supports Completion Across Levels. Knowledge retrieval raises completion from 101 to 108 instances with DeepSeek and from 83 to 91 with Qwen3.5-27B. Both backbones gain across all three levels, with the largest change at Level 2 (8.3 percentage points). This pattern supports the value of reusable modeling guidance for formulation changes. Its resource effect differs across backbones: retrieval reduces DeepSeek mean tokens from 873.5k to 817.6k, while Qwen3.5-27B uses 777.1k with retrieval and 441.2k without it. The common benefit is improved completion, rather than a uniform reduction in context consumption.

Review Adds a Final Opportunity for Correction. Review raises DeepSeek completion from 100/144 to 108/144, with gains of 9.4, 4.2, and 4.7 points across Levels 1–3. With Qwen3.5-27B,



Method	Level 1 Pass@1	Level 1 Δ	Level 2 Pass@1	Level 2 Δ	Level 3 Pass@1	Level 3 Δ
DeepSeek V4 Flash						
GridRecon	96.9	–	77.1	–	62.5	–
w/o Planning Agent	15.6	-81.2	54.2	-22.9	48.4	-14.1
w/o Formulation Agent	93.8	-3.1	56.2	-20.8	43.8	-18.8
w/o Knowledge Retrieval	93.8	-3.1	68.8	-8.3	59.4	-3.1
w/o Review Agent	87.5	-9.4	72.9	-4.2	57.8	-4.7
Qwen3.5-27B						
GridRecon	87.5	–	72.9	–	43.8	–
w/o Planning Agent	6.2	-81.2	72.9	+0.0	46.9	+3.1
w/o Formulation Agent	81.2	-6.2	75.0	+2.1	37.5	-6.2
w/o Knowledge Retrieval	84.4	-3.1	64.6	-8.3	39.1	-4.7
w/o Review Agent	87.5	+0.0	68.8	-4.2	43.8	+0.0

Figure 5: Ablations on IEEE 69 with (a) DeepSeek V4 Flash and (b) Qwen3.5-27B, using 144 instances per variant. Solve is the percentage returning a valid numerical solution. Efficiency metrics are per-instance means. The table reports level-specific Pass@1 (%) and changes from GridRecon (percentage points).

it raises completion from 89/144 to 91/144; the aggregate gain is concentrated at Level 2, while Level 1 and Level 3 totals remain unchanged. Review adds an opportunity to inspect and correct candidate implementations, as illustrated by the recorded case in Appendix F. Its net contribution depends on the backbone: mean runtime rises from 527 to 611 seconds with DeepSeek and from 655 to 837 seconds with Qwen3.5-27B. The ablation measures the net effect of review and its correction loop, while individual traces establish which changes were actually repaired.

5 Conclusion

In this paper, we present GridRecon, a verification-grounded multi-agent framework for distribution-grid model reconstruction, and a 36-request benchmark spanning parameter edits, formulation changes, and coupled state-and-formulation changes. Across four LLM backbones on IEEE 69, GridRecon improves end-to-end success over the strongest adapted baseline by 7.6–34.0 percentage points and passes 329 of 432 instances in a common three-network evaluation. Component ablations show complementary contributions from requirement planning, formulation guidance, knowledge retrieval, and implementation review, supporting the use of localized verification evidence for bounded, targeted self-refinement. We hope the benchmark advances reliable LLM agents for faithful and physically feasible power-system modeling.

AI USE DISCLOSURE

Generative AI tools were used in both system operation and research development. As components of the experimental systems, large language models supported requirement interpretation, mathematical formulation, code generation, and result review in GridRecon and the comparison frameworks. The specific models, tool interfaces, and evaluation settings are described in the main text and appendices.

During research development, we used OpenAI Codex and other generative AI tools to assist in discussing and refining the framework design, benchmark tasks, and experimental plans; writing and debugging system code, baseline adaptations, reference implementations for benchmark tasks, and evaluation code; and organizing experimental records, computing summary statistics, and analyzing failure cases. During manuscript preparation, these tools assisted with literature search and organization, content structuring, initial drafting, translation between Chinese and English, language revision, and the preparation of figures and tables.

The authors determined the research objectives, final method, and evaluation protocol, and reviewed the AI-assisted content adopted in this work. Code and reference implementations were checked using software tests, optimization runs, and physical validation. Experimental statistics were checked against execution records, and references were verified against original sources. The authors take full responsibility for the final text, code, data, results, and conclusions.

REFERENCES

Ali Ahmaditeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 577–596. PMLR, 2024. URL https://proceedings.mlr.press/v235/ahmaditeshnizi24a.html.

Mesut E. Baran and Felix F. Wu. Optimal capacitor placement on radial distribution systems. *IEEE Transactions on Power Delivery*, 4(1):725–734, 1989a. doi: 10.1109/61.19265.

Mesut E. Baran and Felix F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, 4(2):1401–1407, 1989b. doi: 10.1109/61.25627.

Masoud Farivar and Steven H. Low. Branch flow model: Relaxations and convexification—Part I. *IEEE Transactions on Power Systems*, 28(3):2554–2564, 2013. doi: 10.1109/TPWRS.2013.2255317.

Lingwen Gan, Na Li, Ufuk Topcu, and Steven H. Low. Exact convex relaxation of optimal power flow in radial networks. *IEEE Transactions on Automatic Control*, 60(1):72–87, 2015. doi: 10.1109/TAC.2014.2332712.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL https://www.gurobi.com.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Yunkai Hu, Tianqiao Zhao, and Meng Yue. From natural language to solver-ready power system optimization: An LLM-assisted, validation-in-the-loop framework. *arXiv preprint arXiv:2508.08147*, 2025. doi: 10.48550/arXiv.2508.08147. URL https://arxiv.org/abs/2508.08147.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

IEEE Task Force on Load Representation for Dynamic Performance. Load representation for dynamic performance analysis. *IEEE Transactions on Power Systems*, 8(2):472–482, 1993. doi: 10.1109/59.260837.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

LangChain, Inc. LangGraph documentation, n.d. URL https://docs.langchain.com/oss/python/langgraph/overview. Accessed September 26, 2026.

Beibin Li, Konstantina Mellou, Bo Zhang, Jeevan Pathuri, and Ishai Menache. Large language models for supply chain optimization. *arXiv preprint arXiv:2307.03875*, 2023. doi: 10.48550/arXiv.2307.03875. URL https://arxiv.org/abs/2307.03875.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Y. Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/52c4a736cd63b3c7bf52011a1f66f3f6-Abstract-Conference.html.

José Roberto Sanches Mantovani, Fernando Casari, and Rubén Augusto Romero. Reconfiguração de sistemas de distribuição radiais utilizando o critério de queda de tensão. *SBA Controle & Automação*, 11(3):150–159, 2000. URL https://repositorio.unesp.br/items/07d77f2e-4225-47a3-bd00-d83d37d9a2fd.

OpenCode. OpenCode CLI documentation, n.d. URL https://opencode.ai/docs/cli/. Accessed September 26, 2026.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Chao Shen, Zihan Guo, Xu Wan, Zhenghao Yang, Yifan Zhang, Wengi Huang, Jie Song, Zongyan Zhang, and Mingyang Sun. ProOPF: Benchmarking and improving LLMs for professional-grade power systems optimization modeling. *arXiv preprint arXiv:2602.03070*, 2026. doi: 10.48550/arXiv.2602.03070. URL https://arxiv.org/abs/2602.03070.

Joshua A. Taylor and Franz S. Hover. Convex models of distribution system reconfiguration. *IEEE Transactions on Power Systems*, 27(3):1407–1413, 2012. doi: 10.1109/TPWRS.2012.2184307.

Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. pandapower—an open-source Python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, 2018. doi: 10.1109/TPWRS.2018.2829021.

Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58 (2):303–315, 2010. doi: 10.1287/opre.1090.0721.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 50208–50232. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

Ying Wang, Yin Xu, Jiaxu Li, Jinghan He, and Xiaojun Wang. On the radiality constraints for distribution system restoration and reconfiguration problems. *IEEE Transactions on Power Systems*, 35(4):3294–3296, 2020. doi: 10.1109/TPWRS.2020.2991356.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *International Conference on Learning Representations*, pp. 48519–48537, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/d45ee77826332c100a1e15f7765b99ff-Abstract-Conference.html.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Xu Yang, Chenhui Lin, Yue Yang, Qi Wang, Haotian Liu, Haizhou Hua, and Wenchuan Wu. Large language model powered automated modeling and optimization of active distribution network dispatch problems. *IEEE Transactions on Smart Grid*, 17(2):952–965, 2026. doi: 10.1109/TSG.2025.3621438.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xinyang Zhou, Emiliano Dall'Anese, Lijun Chen, and Andrea Simonetto. An incentive-based online optimization framework for distribution grids. *IEEE Transactions on Automatic Control*, 63(7):2019–2031, 2018. doi: 10.1109/TAC.2017.2760284.

Xiyuan Zhou, Ruixi Zou, Xinlei Wang, Yuheng Cheng, Yan Xu, Junhua Zhao, and Jinjin Gu. EngiAgent: Fully connected coordination of LLM agents for solving open-ended engineering problems with feasible solutions. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/66645.

Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26(1):12–19, 2011. doi: 10.1109/TPWRS.2010.2051168.