REPAIR THE ACTION, PRESERVE THE INTENT: MINIMAL FEASIBILITY PROJECTION FOR POWER-GRID AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents can translate operator requests into power-grid actions, but an executable switching or redispatch command can violate connectivity, generator, voltage, thermal, or operational-goal constraints. Re-prompting until a simulator accepts a command supplies neither a minimality guarantee nor a boundary on how much of the request may be replaced. We formulate *intent-preserving feasibility repair*: search a declared finite mixed discrete–continuous neighborhood in lexicographic order of action type, target, support, magnitude, and finally operating cost. Our implementation, IPFR, validates semantic edit buckets with AC power flow, returns the lowest-cost action in the first feasible bucket, and explicitly refuses when the neighborhood contains no admissible repair. This yields soundness and lexicographic minimality relative to the declared neighborhood, without assigning commensurate weights to semantic and monetary quantities. On 120 held-out scenarios from IEEE 57- and 118-bus systems, IPFR finds an AC-feasible action in all 108 repairable cases and refuses all 12 infeasible controls. In 73 conditional, adversarial priority-conflict cases it preserves the proposed action type and target in every case, whereas cost-first feasibility projection does so in none. These cases test contract compliance, not field prevalence. IPFR uses 4.77 AC validations per repairable case versus 60.07 for an exhaustive lexicographic oracle and agrees with that oracle on every selected action. Euclidean and development-tuned scalar projections match its actions at ordinary units but require exhaustive search and lose unit invariance. The contribution is a verifiable boundary between correcting an unsafe agent and silently replacing what it was asked to do.

1 INTRODUCTION

Tool-using agents are beginning to inspect grid cases, call power-flow software, screen contingencies, and propose mitigations (Zhang et al., 2025; Mylonas et al., 2026; Trashchenkov, 2026; Maldonado et al., 2026). Their language interface is attractive: an operator can ask to "relieve line 15 by redispatching the southern pair," and an agent can translate that intent into a structured command. The translation is not a safety certificate. A syntactically valid command can exceed generator limits, island a bus, fail AC power flow, or solve a different operational problem than the one requested.

A common deployment pattern is to reject a failed command and ask the model again. This makes the simulator a binary critic but leaves the correction policy implicit. The next proposal can change a magnitude, generator pair, switching target, or even action family. A conventional projection is more systematic, yet "nearest" depends on a metric: one changed branch ID and one megawatt cannot be compared without an arbitrary conversion. Minimizing operating cost among feasible actions is more dangerous still; it can economically optimize away the requested intervention.

Safety layers and run-time assurance explicitly separate an unverified policy from a verified action (Pham et al., 2018; Dalal et al., 2018; Hobbs et al., 2023). Safe reinforcement learning has adapted projection and security correction to power systems (Su et al., 2025; Bi et al., 2024), while optimal power flow and corrective switching supply mature physical models (Zimmerman et al., 2011; Coffrin et al., 2018; Fisher et al., 2008). Our question is narrower and semantic: *when safety requires*

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.

editing an agent's mixed discrete–continuous action, which edits are permitted to take priority over which others?

We answer with a typed action grammar and a lexicographic repair contract. The contract first insists on physical feasibility; among feasible candidates it minimizes changes to action type, target/support, and normalized magnitude in declared order, then uses operating cost only as a final tie-breaker. IPFR searches semantic buckets rather than scalarizing incomparable quantities. Its guarantee is deliberately local: minimality holds in the published finite repair neighborhood, not over every possible grid action.

Our contributions are:

- a formal intent-preserving repair problem for switching and balanced redispatch, including an explicit NO-REPAIR outcome;
- IPFR, a solver-in-the-loop bucket search with soundness, lexicographic-minimality, non-override, and refusal properties; and
- a fully executable AC benchmark on four public IEEE systems. Held-out tests separate ordinary corruptions, conditional cost–intent conflicts, target errors, and infeasible controls rather than pooling them silently; an exhaustive lexicographic oracle verifies every returned action.

We do not claim that an LLM inferred the correct operator intent, that the finite neighborhood contains a globally optimal grid intervention, or that simulation alone makes an action deployable. The contribution is an inspectable correction boundary after an agent has emitted a structured plan.

2 RELATED WORK

Agentic power-system workflows. Grid-Agent combines language-model planning with numerical validation and rollback (Zhang et al., 2025). GridMind orchestrates AC optimal power flow, contingency analysis, and cascade tools with artifact provenance (Maldonado et al., 2026). New executable benchmarks expose tool contracts, hidden evaluators, and evidence-sensitive scores (Mylonas et al., 2026; Trashchenkov, 2026). These systems motivate a post-planning repair interface, but they do not specify when a validator may replace one requested actuator or operation type with another.

Safety filters and projections. OptLayer projects a neural action toward a constraint set (Pham et al., 2018); analytic safety layers correct continuous actions under local linear models (Dalal et al., 2018); run-time assurance switches from an unverified primary controller to a safety mechanism (Hobbs et al., 2023). Power-system safe-RL methods employ projection, shielding, Lyapunov constraints, or learned security margins (Su et al., 2025; Bi et al., 2024; Vu et al., 2021). They primarily ask whether the executed action is safe. IPFR additionally declares a hierarchy among semantically different safe replacements and exposes failure when none lies inside the authorized neighborhood.

Grid optimization and corrective actions. MATPOWER and PowerModels provide reproducible AC/DC network formulations (Zimmerman et al., 2011; Coffrin et al., 2018). Corrective switching and security-constrained OPF search feasible, low-cost interventions (Fisher et al., 2008; Hedman et al., 2010). That is appropriate when the optimizer owns the objective. Here an upstream agent has already selected an action type and target from a natural-language request. Operating cost is therefore a final tie-breaker, not permission to substitute an unrelated action.

2.1 THE INTEGRATION BOUNDARY

IPFR begins *after* language understanding. Let u be an operator utterance, P an arbitrary parser or agent, and $a_0 = P(u, x)$ its structured proposal in state x. Our contract does not certify that P interpreted u correctly. It certifies a narrower implication: if the authorized edit order and finite neighborhood accurately express what may be changed, the executed action is validator-feasible and lexicographically closest under that declared order. The wrapper should preserve three records: (u, a_0), the repair diff $a^\star - a_0$, and the physical witness $V_x(a^\star)$. Conflating these stages would let a physically valid repair hide an upstream semantic error.

This separation also defines appropriate evidence. An end-to-end LLM accuracy test would measure P, whereas our controlled corruptions test the repair contract conditional on a_0. Re-proposal below is consequently a budget-matched stochastic-search control, not a claim about the behavior of a particular language model. The deterministic language templates demonstrate the typed interface and are not counted as a natural-language benchmark.

3 Intent-Preserving Feasibility Repair

3.1 Typed Actions and the Validator

For a grid state x, an action is

$$\begin{aligned} &a = (\tau, b, \Delta p), \\ &\tau \in \{\textsc{Redispatch}, \textsc{Switch}, \textsc{Switch+Redispatch}\}. \end{aligned} \tag{1}$$

b is either a branch index or null; $\Delta p \in \mathbb{R}^{n_g}$ is a balanced generator change. The grammar requires a null branch for pure redispatch, zero Δp for pure switching, online generators only, and $\mathbf{1}^\top \Delta p = 0$.

The deterministic validator $V_x(a)$ applies a, checks grammar, generator and connectivity constraints, solves AC power flow, then checks

$$\begin{aligned} \underline{v} \le |v_i| \le \overline{v}, \quad |S_\ell| \le \overline{S}_\ell, \\ \underline{p}_g \le p_g \le \overline{p}_g, \quad \underline{q}_g \le q_g \le \overline{q}_g, \end{aligned} \tag{2}$$

and a scenario-specific goal such as reducing flow on a designated branch. It returns either an infeasibility reason or a witness containing voltages, flows, generation, goal progress, and operating-cost change $C_x(a)$.

3.2 A Lexicographic Edit Contract

Let a_0 be the agent proposal and $\mathcal{N}(a_0)$ a finite, declared neighborhood. Define its typed target as $T(a) = (b, \mathrm{supp}(\Delta p))$: a branch identifier (possibly null) and the set of commanded redispatch generators. The support cardinality separately measures how many actuators change. With generator ranges $r_g = \overline{p}_g - \underline{p}_g$, define

$$\begin{aligned} d_\tau(a, a_0) &= \mathbb{1}[\tau \neq \tau_0], \\ d_T(a, a_0) &= \mathbb{1}[T(a) \neq T(a_0)], \\ d_S(a, a_0) &= |\,\mathrm{supp}(\Delta p) \triangle \mathrm{supp}(\Delta p_0)|, \\ d_M(a, a_0) &= \sum_g \frac{|\Delta p_g - \Delta p_{0g}|}{\max(r_g, 10^{-6})}. \end{aligned} \tag{3}$$

d_T is a categorical authorization decision: it asks whether the commanded object changes at all. Conditional on that decision, d_S prefers replacing fewer actuators. For example, replacing one member of a generator pair and replacing both are both target edits ($d_T = 1$), but have $d_S = 2$ and 4. This avoids pretending that a branch identifier and a number of generators have commensurate distance. The repair objective is

$$\operatorname*{lexmin}_{a \in \mathcal{N}(a_0) : V_x(a) = 1} \big(d_\tau, d_T, d_S, d_M, C_x(a)\big). \tag{4}$$

This is not a weighted sum. Scaling dollars to cents or changing the units of Δp cannot promote operating cost above an action-type decision.

Why a scalar penalty is a different contract. For weights $w_i > 0$, a scalar objective $\sum_i w_i d_i$ reproduces a lexicographic ordering on a finite set only if every higher-priority weight dominates the maximum attainable sum of all lower-priority terms. Those bounds depend on neighborhood size, units, and cost range. A weight tuned on one case can therefore invert a priority after a currency or base-MVA change.

Require: proposal a_0, neighborhood $\mathcal{N}$, validator V_x
partition $\mathcal{N}$ into buckets $\{\mathcal{B}_s\}$ by semantic key
for s in increasing lexicographic order **do**
  $\mathcal{F}_s \leftarrow \emptyset$
  for $a \in \mathcal{B}_s$ **do**
    run $V_x(a)$; if feasible, add $(a, C_x(a))$ to $\mathcal{F}_s$
  end for
  if $\mathcal{F}_s \neq \emptyset$ **then**
    return least-cost action in $\mathcal{F}_s$ and its witness
  end if
end for
return NO-REPAIR and the complete failure ledger

Figure 1: IPFR. Candidate construction and the AC validator are independent of the upstream agent.

Proposition 1 (unit invariance). Applying any strictly increasing transformation to one coordinate of every candidate leaves the lexicographic minimizer unchanged, provided deterministic ties are preserved. A fixed weighted sum need not be invariant to positive rescaling of one coordinate. Lexicographic comparison is decided by the first unequal coordinate; for a weighted sum, a sufficiently large rescaling of a lower-priority nonzero difference can reverse the scalar order.

3.3 IPFR

IPFR groups candidates by the four-coordinate semantic key $s(a) = (d_\tau, d_T, d_S, d_M)$ and sorts distinct keys lexicographically. It validates every action in one bucket before advancing. At the first bucket containing any feasible action, it returns the member of minimum C_x. If all buckets fail, it emits NO-REPAIR with the tested neighborhood hash and validator reasons. Equal costs are broken by the canonical serialized action, making replay deterministic. This bucket form also permits parallel AC solves within a semantic priority.

Proposition 2 (soundness). If IPFR returns $a^\star$, then $a^\star \in \mathcal{N}(a_0)$ and $V_x(a^\star) = 1$.

Proposition 3 (declared lexicographic minimality). For finite $\mathcal{N}(a_0)$ and deterministic V_x, an IPFR return is a solution of Eq. (4).

Proof sketch. Candidates are partitioned by s. The first feasible bucket has no feasible predecessor, so its semantic key is minimal. All members of that bucket are validated and the least-cost feasible member is returned.

Corollary 1 (non-override). If a feasible candidate with the proposal's action type and target/support exists, IPFR cannot return a candidate that changes either.

Proposition 4 (honest refusal). IPFR returns NO-REPAIR iff no member of the declared finite neighborhood passes V_x.

3.4 CERTIFICATES AND REFUSAL LEDGERS

A successful return is accompanied by

$$\mathcal{Q} = (h_\mathcal{N}, h_V, a_0, a^\star, s^\star, w^\star), \tag{5}$$

where $h_\mathcal{N}$ hashes the canonical neighborhood, h_V identifies the validator configuration, $s^\star$ is the first feasible semantic key, and $w^\star$ is the AC witness. A replay checker can verify neighborhood membership, recompute $V_x(a^\star)$, confirm that every earlier bucket is infeasible, and confirm that no feasible member of $\mathcal{B}_{s^\star}$ has lower cost. The certificate is relative to the supplied neighborhood and validator; it is not a claim that no feasible grid action exists elsewhere.

A refusal record replaces $(a^\star, s^\star, w^\star)$ with one outcome per canonical candidate. Its strongest justified statement is therefore "no authorized candidate passed this validator." This distinction matters for operators: a local refusal should escalate planning or human review, not be interpreted as proof

that the physical system has no corrective action. The submitted implementation records the neighborhood hash and individual failure reasons so the boundary can be audited after execution.

4 Experimental Protocol

4.1 Cases, Scenarios, and Neighborhoods

We use public MATPOWER IEEE 14/30 systems for development (40 randomized states each) and IEEE 57/118 systems for held-out testing (60 each) (Zimmerman et al., 2011). Loads receive seed-fixed perturbations; each accepted base state passes the same AC validator. Synthetic emergency ratings place base-case branch utilization uniformly in $[0.25, 0.45]$ so the study tests repair semantics rather than reproducing field ratings. Voltage bounds are $[0.94, 1.06]$ p.u.; active-balance tolerance is 10^{-5} MW. Because the stock IEEE-57 solution is below 0.94 p.u., that case uses a documented 0.85–0.95 load multiplier; the others use 1.05–1.15. We compare methods within the same scenario, never interpret this network-specific normalization as a cross-network method effect.

Each operator-language template names an overload-relief or contingency-response intent and is compiled to Eq. (1). We then create controlled unsafe proposals. A redispatch neighborhood contains up to four ordered generator pairs, both orientations for alternative pairs, and magnitude multipliers $\{0, .25, .5, .75, 1, 1.25, 1.5, 1.75, 2\}$. A switching neighborhood contains the proposal branch and up to four alternatives; type-changing actions are also available but ordered last.

The 120 held-out scenarios comprise 73 *priority conflicts*, conditioned on the existence of both a feasible same-target repair and a cheaper feasible different-target repair; 34 *target corruptions*, whose proposed switch islands the graph; one ordinary magnitude corruption; and 12 *refusal controls*, whose required flow reduction is unattainable in the neighborhood. Conditional conflict frequency is not an estimate of operational prevalence. A generation-only failure on IEEE-57 prompted a documented base-state correction before any held-out method result existed; the full audit trail is supplied.

Protocol amendments. The development pilot initially made every redispatch case conditional on a cost–intent conflict. That condition generated only three IEEE-57 instances. Before any held-out method file existed, we registered two changes: retain an ordinary unsafe-magnitude case when the conflict condition is absent, and preserve the heterogeneous generator-voltage profile of IEEE-57/118 instead of replacing every setpoint by one value. The stock IEEE-57 case still violated the fixed lower-voltage bound under increased load, motivating its light-load range above. These changes affect scenario availability rather than a method outcome and are timestamped with failed-run traces.

Post-review method correction. An integrity review found that an earlier implementation counted generator support difference inside both target and support coordinates and stopped at the first feasible canonical action instead of completing its semantic bucket. Before the results reported here were generated, we applied the typed definition in Eq. (3), implemented complete-bucket cost tie-breaking, regenerated development weights, and reran both held-out networks from scratch. The dated correction and original outputs remain in the Code and Data Supplement.

4.2 Validator and Candidate Accounting

For each action, branch flow is the larger terminal apparent-power magnitude. Thermal loading divides it by the declared emergency rating. The active-power redispatch is balanced before the solve; the AC slack response and losses are then reflected in the post-solve generator outputs. $C_x(a)$ evaluates the case-file polynomial generation costs on those outputs and normalizes their difference by base cost. Thus the cost tie-breaker uses the physical post-action solution rather than only the requested setpoint change.

Generator-pair candidates exclude the slack generator when enough alternatives exist. Their order is seed-fixed; the proposal pair comes first and at most three alternatives are authorized. For switching, the proposed branch comes first and alternatives are ranked by base-state apparent flow with seed-fixed jitter for deterministic ties. Candidate canonicalization removes duplicate zero-magnitude and

Table 1: Held-out repairable cases ($n = 108$). Semantic success means AC feasible with original action type and target/support retained.

Method	Feasible (%)	Semantic (%)	AC calls
No repair	0.0	0.0	1.00
Clip	13.0	13.0	1.00
Re-propose	45.4	6.5	3.70
Cost-first	100.0	0.9	60.07
Euclidean	100.0	68.5	60.07
Weighted	100.0	68.5	60.07
Lex-oracle	100.0	68.5	60.07
IPFR	**100.0**	**68.5**	**4.77**

repeated actions. The resulting neighborhoods contain 31–77 unique actions in held-out testing. A validator call is one independent application of the complete grammar, connectivity, AC power-flow, limit, and goal checks to one canonical candidate; proposal validation is counted when the proposal appears in the ordered search.

The same cached candidate ledger is exposed to every exhaustive baseline. Caching is used only to compare methods and replay evidence: IPFR's reported call count is the number it would require without borrowing baseline results. The Re-propose control uses a seed-fixed random permutation and IPFR's call budget. It is a stochastic-search control, not an empirical model of a specific LLM's re-prompt distribution.

4.3 Baselines and Metrics

All methods receive identical candidates and cached AC outcomes. **No repair** executes the proposal. **Clip** clips generator bounds and restores declared balance. **Re-propose** samples candidates uniformly using the same validator-call budget as IPFR. **Cost-first** chooses the least-cost feasible candidate. **Euclidean** minimizes one-hot discrete plus normalized continuous distance. **Weighted** minimizes a scalar combination of the five edit coordinates; its weights are selected once from $3 \times 3 \times 3 \times 3 = 81$ settings on IEEE 14/30. **Lex-oracle** exhausts the same candidates and minimizes $(d_\tau, d_T, d_S, d_M, C_x)$; it verifies IPFR's selected action but cannot stop early. The no-refusal ablation returns the candidate with greatest goal progress even when none is feasible.

Primary metrics are feasible-repair coverage and joint action-type/target retention. Their conjunction is *semantic repair success*. We also report AC validator calls and correct refusal. Scenario bootstrap confidence intervals use 10,000 resamples. Semantic success is compared by exact McNemar tests and calls by paired two-sided Wilcoxon tests. Each predeclared outcome family contains one test, so the registered Holm adjustment is the identity. Seeds 44596–44600 and every accepted scenario manifest are included.

5 Results

Feasibility and intent. Table 1 shows that every exhaustive projection and IPFR repairs all 108 repairable proposals. This parity matters: preserving intent does not reduce feasibility within the declared neighborhoods. Exact target retention is possible in 74 cases and impossible in the 34 islanding-target corruptions. IPFR, Euclidean, and Weighted attain this empirical ceiling: 74/108, or 68.5% (bootstrap 95% CI 59.3–76.9). Cost-first retains the target in only 1/108. On the 73 conditional priority conflicts, IPFR retains type and target in 73/73 while Cost-first retains them in 0/73. The paired discordances are 73 versus 0, a paired risk difference of 67.6 percentage points over all repairable cases (bootstrap 95% CI 58.3–75.9; exact McNemar $p = 2.12 \times 10^{-22}$). This effect is conditional on the constructed stress test and is not an estimate of how often cost and intent conflict in service.

Verification effort. IPFR uses 4.77 validator calls per repairable case, compared with 60.07 for each exhaustive projection and Lex-oracle (Fig. 2). It selects exactly the Lex-oracle action in 108/108

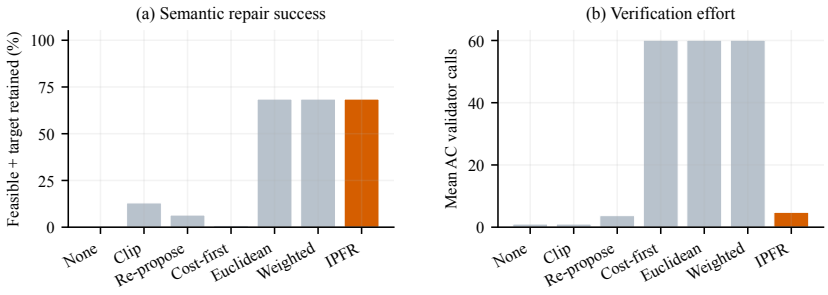


Figure 2: Semantic outcome and verification effort on repairable held-out cases. Bars are paired over the same finite candidate sets.

Table 2: Complete held-out candidate failure ledger. Percentages use all 6,309 infeasible candidates.

Failure reason	Count	Share (%)
Active-generator bound	4,817	76.4
Insufficient goal reduction	815	12.9
Voltage bound	298	4.7
Reactive-generator bound	149	2.4
Islanding	120	1.9
Thermal bound	110	1.7

repairable cases. The paired median saving is 61 calls (bootstrap 95% CI 61–62; two-sided Wilcoxon $p = 1.63 \times 10^{-19}$). This is an ordering benefit, not fewer physical checks per candidate: IPFR stops only after completing the first feasible semantic bucket.

Refusal. All 12 explicit controls contain no valid candidate. IPFR, Euclidean, Weighted, Cost-first, and Re-propose correctly return no action in 12/12. The no-refusal ablation always emits a command, but every such command fails the validator. Thus returning "the best available" is not equivalent to repair when the authorized neighborhood is empty.

What the validator rejects. The held-out ledger contains 7,171 unique candidate–scenario evaluations: 862 feasible and 6,309 infeasible. Table 2 shows that the benchmark is not driven by AC nonconvergence; most rejected actions exceed an active-generator bound, followed by insufficient task progress. Connectivity, voltage, reactive, and thermal checks still produce distinct counterexamples. Keeping these reasons is useful operationally: a bound violation may be repaired locally, while an exhausted goal ledger should trigger replanning.

[†]Correct refusal, rather than target retention.

Cost is secondary, not ignored. Among repairable cases, IPFR's mean normalized operating-cost change is $+0.735\%$ of base cost; Cost-first obtains -0.325% by changing the target in 107/108 cases. Euclidean and Weighted select exactly the same action and cost as IPFR in the tested units. IPFR does not claim minimum grid cost across semantic buckets. It claims the least cost after the authorized semantic edit has been fixed.

Across held-out networks. IPFR repairs all 54 non-control IEEE-57 and all 54 non-control IEEE-118 instances. Mean calls for conditional redispatch are 4.67 and 4.65, respectively; switch repair uses 5.00 on both. This near equality follows from the fixed neighborhood ordering, not equal wall time. The larger case is more expensive per AC solve, which motivates reporting calls separately from hardware-dependent seconds.

One decision, fully traced. The operator template asks to relieve monitored branch 7 while retaining generator pair $(1, 2)$ if a safe magnitude exists. The 87.95-MW proposal violates an active-

Table 3: Separated held-out strata. "Retain" is exact type+target retention; it is structurally impossible for an unsafe bridge target.

Stratum	Method	n	Retain (%)	Calls
Priority conflict	Cost-first	73	0.0	53.79
	Euclidean/Weighted	73	100.0	53.79
	IPFR	73	**100.0**	**4.66**
Unsafe bridge	Exhaustive projections	34	0.0	74.38
	IPFR	34	0.0	**5.00**
No repair exists	**IPFR refusal**	12	100.0†	56.92

Table 4: Auditable IEEE-57 trace (scenario test_case57_0000). Generator indices follow the supplied case file. ΔC is normalized change from base cost and R is monitored-flow reduction.

Choice	Pair	MW	d_T	ΔC	R
Proposal	$1\uparrow, 2\downarrow$	87.95	0	infeasible	–
Cost-first	$1\uparrow, 4\downarrow$	87.95	1	-0.814%	21.76%
IPFR / oracle	$1\uparrow, 2\downarrow$	43.97	0	$+0.900\%$	1.21%

generator bound. Of 66 authorized candidates, IPFR completes the first feasible semantic bucket in four AC calls and halves the same-pair redispatch to 43.97 MW. It exceeds the required 0.5% reduction, achieving 1.21%, with a $+0.900\%$ cost change. Cost-first checks all 66 candidates and saves cost by replacing generator 2 with generator 4; that action reduces the monitored flow more, but crosses the declared target boundary. Lex-oracle returns exactly the IPFR action. All values in Table 4 are extracted from the submitted candidate ledger.

Why not a tuned weighted sum? At the development-selected units, Weighted selects exactly the IPFR action in all repairable tests. This is a useful negative result: weighting can mimic the lexicographic rule on one scale. It does not encode the same contract. Multiplying the operating-cost coordinate by 10^4 reduces exact agreement to 53.7% and target retention from 68.5% to 59.3%, while IPFR is unchanged (Fig. 3). The theory predicts this unit invariance; the experiment illustrates when scalarization loses it.

6 Limitations and Scope

Minimality is relative to a small enumerated neighborhood, not the global mixed-integer AC feasible region. The benchmark uses one-step steady-state AC power flow, synthetic emergency ratings, public test cases, and controlled corruptions; it does not model transient stability, protection timing, uncertain topology, multi-period reserves, or human approval. Target-error cases intentionally make exact target retention impossible, so their zero retention is a task property, not evidence that another method should keep an unsafe target. The priority-conflict stratum is conditional by construction and cannot estimate field frequency. PYPOWER's NumPy-2 path does not robustly enforce reactive limits internally; we therefore widen scenario-local reactive reserves around the solved base point and check the declared limits externally with a 1-Mvar tolerance. Deployment would require a production solver, contingency security, dynamics, uncertainty sets, and operator-defined edit priorities.

Threats to validity. Internal validity rests on a deterministic PYPOWER validator and cached candidate table rather than an independent commercial solver. We mitigate selection errors with the exhaustive lexicographic oracle, exact action agreement, tests that distinguish target identity from support cardinality, and a complete post-review rerun. Construct validity is narrower: "intent" means equality under the published typed action fields, not human mental state. External validity is limited by synthetic ratings, small neighborhoods, and conditional stress construction. Statistical uncertainty therefore describes this benchmark population only. The refusal controls and impossible-target cases prevent a uniformly positive story, but do not substitute for field trials.

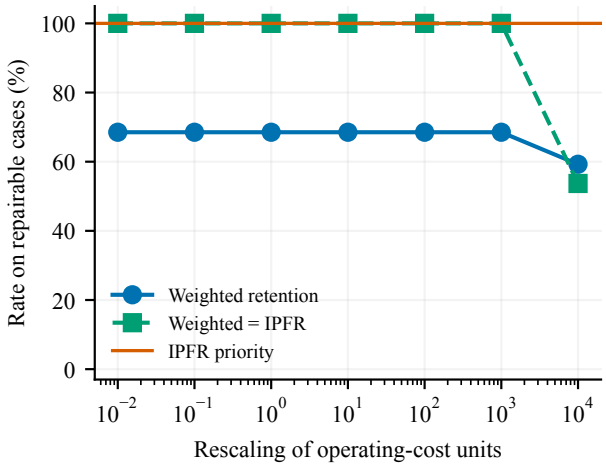


Figure 3: A tuned scalar objective depends on the arbitrary numerical scale of operating cost; the lexicographic priority is scale invariant.

7 Conclusion

An agent action can be physically invalid without granting a repair layer permission to replace its meaning. IPFR makes that permission explicit: validate a finite neighborhood in semantic order, minimize cost only inside the first feasible edit bucket, and refuse if no authorized repair exists. The result is not a smarter language model; it is a small, auditable boundary between language-mediated intent and grid physics.

Operational interface. A deployable wrapper should present four objects together: the original request, the parsed proposal, a human-readable edit diff, and the AC witness or refusal ledger. An operator can reject a repair whose declared priorities are formally respected but contextually inappropriate. This preserves human authority and post-incident traceability; IPFR is a guardrail, not an autonomous dispatch policy.

Reproducibility Statement

Data and code availability. All source code, scenario manifests, candidate-level outcomes, seeds, analysis scripts, and generated figures are included in the submitted Code and Data Supplement. Public grid cases are distributed with MATPOWER/PYPOWER. No external anonymous repository is required.

Ethics Statement

The study uses no human participants, personal data, or live infrastructure. It must not be interpreted as authorization for autonomous grid control.

Author Contributions and Disclosures

The anonymous author performed conceptualization, implementation, experiments, analysis, and writing. **Conflicts of interest.** None declared. **Funding.** No funding information is disclosed during anonymous review; the camera-ready version will state the applicable funding status.

AI Use Statement

Generative AI assisted with language editing and code drafting. OpenAI Codex also assisted with adapting the LaTeX formatting to the ICLR 2027 template and preparing submission metadata. The author checked the implementation, executed every reported experiment, inspected outputs, and remains responsible for all claims.

REFERENCES

Congbo Bi, Lipeng Zhu, Di Liu, and Chao Lu. Make safe decisions in power system: Safe reinforcement learning based pre-decision making for voltage stability emergency control. *arXiv preprint arXiv:2405.16485*, 2024.

Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. PowerModels.jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference*, pp. 1–8, 2018. doi: 10.23919/PSCC.2018.8442948.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Emily B. Fisher, Richard P. O'Neill, and Michael C. Ferris. Optimal transmission switching. *IEEE Transactions on Power Systems*, 23(3):1346–1355, 2008. doi: 10.1109/TPWRS.2008.922256.

Kory W. Hedman, Michael C. Ferris, Richard P. O'Neill, Emily B. Fisher, and Shmuel S. Oren. Co-optimization of generation unit commitment and transmission switching with n-1 reliability. *IEEE Transactions on Power Systems*, 25(2):1052–1063, 2010. doi: 10.1109/TPWRS.2009.2037232.

Kerianne L. Hobbs, Mark L. Mote, Matthew Abate, Samuel Coogan, and Eric Feron. Run time assurance for safety-critical systems: An introduction to safety filtering approaches for complex control systems. *IEEE Control Systems Magazine*, 43(2):28–65, 2023. doi: 10.1109/MCS.2023.3234380.

D. Adrian Maldonado, Jonghwan Kwon, Sunash B. Sharma, Hongwei Jin, Stefano Fenu, Emil Constantinescu, David Sehloff, Parfait Gasana, Duane Rudolph Verner, and Kibaek Kim. Gridmind: An agentic workflow for weather-driven power grid risk assessment. In *KDD Workshop on Scientific Agents and Large Language Models*, 2026.

Costas Mylonas, Magda Foti, Andrea Pomarico, Matheus Duarte, Qian Zhang, and Emmanouel Varvarigos. PowerAgentBench-SS: A benchmark for agentic ai in power system steady-state studies. *arXiv preprint arXiv:2606.18789*, 2026.

Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. OptLayer: Practical constrained optimization for deep reinforcement learning in the real world. In *IEEE International Conference on Robotics and Automation*, pp. 6236–6243, 2018. doi: 10.1109/ICRA.2018.8460547.

Tong Su, Tong Wu, Junbo Zhao, Anna Scaglione, and Le Xie. A review of safe reinforcement learning methods for modern power systems. *Proceedings of the IEEE*, 113(3):213–255, 2025. doi: 10.1109/JPROC.2025.3584656.

Sergei Trashchenkov. Power systems agent benchmark: Executable evaluation of ai agents in electric power engineering. *arXiv preprint arXiv:2606.20950*, 2026.

Thanh Long Vu, Sayak Mukherjee, Tim Yin, Renke Huang, Jie Tan, and Qiuhua Huang. Safe reinforcement learning for emergency load shedding of power systems. In *2021 IEEE Power & Energy Society General Meeting*, pp. 1–5, 2021. doi: 10.1109/PESGM46819.2021.9638007.

Yan Zhang, Ahmad Mohammad Saber, Amr Youssef, and Deepa Kundur. Grid-agent: An llm-powered multi-agent system for power grid control. *arXiv preprint arXiv:2508.05702*, 2025.

Ray D. Zimmerman, Carlos E. Murillo-Sanchez, and Robert J. 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.

A Additional Guarantee Details

Proof of Proposition 3. Because $\mathcal{N}(a_0)$ is finite, its candidates induce finitely many semantic keys $s_1 < \cdots < s_m$. Let j be the first index for which $\mathcal{B}_{s_j}$ contains a validator-feasible action. Complete validation of every preceding bucket proves that no feasible candidate has semantic key smaller than s_j. Complete validation of $\mathcal{B}_{s_j}$ and the final minimum-cost selection prove that no feasible candidate with key s_j has smaller C_x. Hence the returned action minimizes the full tuple in Eq. (4). Canonical serialization resolves any remaining equality, so repeated runs under deterministic V_x return the same action.

Non-override and refusal. If a feasible action retains type and target, it lies in a bucket with $(d_\tau, d_T) = (0, 0)$. Every type- or target-changing action has a strictly larger prefix and therefore cannot be returned first. This proves Corollary 1 independently of magnitude or cost. Conversely, IPFR reports NO-REPAIR only after validating every candidate; thus refusal is equivalent to emptiness of the declared feasible subset. Neither result says that a human chose the correct target or that $\mathcal{N}$ contains every useful control action. The guarantee is deliberately contractual rather than global.

What can falsify the implementation. Four observable failures would contradict the claims: a returned action fails V_x; a feasible earlier bucket exists; a cheaper feasible action exists in the returned bucket; or a refusal ledger contains a feasible candidate. Our exhaustive Lex-oracle checks the middle two conditions, candidate-level outcomes check the first and fourth, and target/support unit tests check that the implemented tuple matches Eq. (3).

Complexity and caching. With $K = |\mathcal{N}|$, worst-case validation cost is K AC solves, matching exhaustive projection. If the cumulative number of candidates through the first feasible bucket is k, IPFR uses exactly k solves. Results are cached by canonical action strings, so audit replay and baselines do not recompute a candidate. A bucket may be evaluated in parallel without changing the selected action; we use serial evaluation so validator calls remain directly interpretable.

These statements do not establish global AC optimality. They establish exactly what a repair layer can audit: the validator, authorized candidate set, edit order, and returned witness.

B Operational Use and Counterfactuals

When the original target is repairable. Suppose the proposal violates a magnitude bound but at least one smaller command on the same typed target is feasible. Corollary 1 prevents every different-target or different-type candidate from being returned, regardless of its operating cost. IPFR may still choose among multiple magnitudes in the first feasible bucket using C_x. This is the intended "repair, do not override" regime illustrated in Table 4.

When the original target is unsafe. For a proposed switching action on a bridge, no connected same-target action exists. The contract then advances to the smallest authorized target edit. A zero retained-target score is correct behavior in this stratum: insisting on the original branch would either island the graph or force refusal despite an authorized alternative. The ordered tuple makes this transition visible, whereas a scalar nearest-action score does not identify which semantic boundary was crossed.

When the neighborhood is empty of feasible actions. A no-refusal optimizer can rank infeasible candidates by violation or goal progress, but executing the top-ranked candidate changes the meaning of "repair." IPFR instead returns a complete local failure ledger. A controller can use the dominant reason to choose the next process: expand the candidate neighborhood, invoke security-constrained optimization, request a new agent plan, or escalate to an operator. It must not automatically weaken voltage, thermal, connectivity, or task constraints merely to produce an action.

When the upstream interpretation is wrong. If $P(u, x)$ names the wrong generator pair but that pair admits a safe magnitude, IPFR will faithfully preserve the wrong target. This is not a failure of Proposition 3; it is evidence that parser assurance and action repair solve different problems. A production interface should show the natural-language request, structured target, proposed magnitude,

repair diff, and witness on one screen. Human acceptance can then reject a semantically wrong but physically feasible result.

Recommended execution gates. We envision five explicit gates. First, schema validation rejects malformed or out-of-scope actions before numerical search. Second, the unmodified proposal is validated and executed only if policy permits direct action. Third, IPFR searches a versioned neighborhood whose edit order was approved independently of the current incident. Fourth, any changed action is presented with its certificate and human-readable diff; high-impact switching should require human authorization. Fifth, a refusal triggers escalation rather than constraint relaxation. Logs should bind the state snapshot, parser version, neighborhood hash, validator settings, selected action, and witness so that a later audit can reconstruct exactly what was authorized.

The benchmark implements the first three gates and the evidence object for the fourth; it does not implement control-room workflow or automatic execution. This boundary is central to the safety claim. IPFR is a deterministic post-planning mechanism that can be tested independently of model updates, but its priorities remain a policy choice for the grid operator.

Policy configuration and state freshness. The edit order should be versioned like a protection setting, not embedded in a prompt. Different operating modes may authorize different neighborhoods: redispatch-only during routine congestion, switching alternatives after contingency analysis, and no automatic type change for high-consequence events. Approval should specify candidate constructors, validator tolerances, goal definitions, and which semantic boundaries require human confirmation. Changing any of these invalidates an earlier certificate and requires a new hash.

Feasibility is also state-dependent. Between proposal, validation, approval, and execution, telemetry or topology may change. A practical certificate therefore needs a state-snapshot identifier and an expiration rule. If the state moves beyond the declared tolerance, the wrapper must revalidate the selected action rather than replay the old result. Bucket-level parallelism can reduce latency without changing the selected tuple, but partial evaluation of a bucket cannot justify the minimum-cost tie-break. These conditions are not additional guarantees proved by the benchmark; they are deployment requirements implied by the guarantee's assumptions.

What the study supports. The evidence supports four bounded conclusions: returned actions pass the declared static AC validator; the implementation matches exhaustive lexicographic selection on the held-out candidate sets; early stopping reduces the number of required validations; and scalar weights can lose the declared priority under a unit change. It does not establish superiority for language understanding, global corrective control, dynamic security, or naturally distributed field incidents. Stating this boundary is part of the proposed contract, not a retreat from it.

C Purpose and Contents

This supplement expands the proofs, validator, scenario generator, statistical analysis, and reproducibility details for the main paper. It is not needed to interpret the central claims: the complete problem, propositions, primary protocol, and results appear in the main paper. The separately uploaded Code and Data Supplement contains every file named below.

D Formal Definitions and Proofs

D.1 Candidate Set and Order

Let the structured proposal be $a_0 = (\tau_0, b_0, \Delta p_0)$ and let $\mathcal{N}(a_0)$ be the finite list returned by the public candidate constructor. Canonical JSON serialization removes duplicates. For a candidate a, write

$$s(a) = \big(d_\tau(a, a_0), d_T(a, a_0), d_S(a, a_0), d_M(a, a_0)\big).$$

The typed target is $T(a) = (b, \mathrm{supp}(\Delta p))$. $d_T = \mathbb{1}[T(a) \neq T(a_0)]$ records categorical target identity, while d_S is the support symmetric-difference cardinality. The former asks whether the authorized object changes; the latter refines how many actuators differ. The exact coordinates are in Eq. (3) of the main paper and in `src/ipfr/core.py`. Distinct candidates with the same $s(a)$

form one semantic bucket $\mathcal{B}_s$. The validator is a deterministic function $V_x(a) \in \{0, 1\}$ with a reason and numerical witness. When $V_x(a) = 1$, $C_x(a)$ is its normalized post-action generation-cost change.

D.2 ALGORITHM

Require: proposal a_0, finite neighborhood $\mathcal{N}(a_0)$, validator V_x
remove duplicate candidates by canonical serialization
partition candidates into $\mathcal{B}_s = \{a : s(a) = s\}$
for s in increasing lexicographic order **do**
  $\mathcal{F}_s \leftarrow \emptyset$
  for $a \in \mathcal{B}_s$ **do**
    $(q, w) \leftarrow V_x(a)$
    if $q = 1$ **then**
      add $(a, C_x(a), w)$ to $\mathcal{F}_s$
    end if
  end for
  if $\mathcal{F}_s \neq \emptyset$ **then**
    return member with minimum $(C_x(a), \mathrm{canon}(a))$
  end if
end for
return NO-REPAIR, neighborhood hash, and reason ledger

D.3 PROOFS

Proposition 1 (unit invariance). Let $u = (u_1, \ldots, u_m)$ and $v = (v_1, \ldots, v_m)$ be two objective vectors, and let j be their first unequal coordinate. For strictly increasing f_j, $u_j < v_j$ iff $f_j(u_j) < f_j(v_j)$; earlier coordinates remain equal. Thus applying independent strictly increasing maps to the coordinates preserves every pairwise lexicographic comparison and the minimizer set.

For non-invariance of a weighted sum, take $u = (0, M)$ and $v = (1, 0)$ with weights $(w_1, w_2) > 0$. Initially $w_2 M < w_1$ may select u. Rescaling the second coordinate by $\alpha > w_1/(w_2 M)$ selects v. Hence positive weights alone do not encode an absolute priority.

Proposition 2 (soundness). IPFR inserts a into $\mathcal{F}_s$ only after the common validator returns one. The returned action is selected only from a nonempty $\mathcal{F}_s$ and every bucket is a subset of $\mathcal{N}(a_0)$. Therefore the returned action is both a declared candidate and validator-feasible.

Proposition 3 (lexicographic minimality). Let $\hat{s}$ be the first key whose bucket contains a feasible action. Every key $s <_{\mathrm{lex}} \hat{s}$ has been completely validated and contains no feasible action. The returned action therefore has the smallest possible semantic key. IPFR validates the whole $\mathcal{B}_{\hat{s}}$ and minimizes C_x among its feasible members, with canonical serialization only breaking an exact cost tie. This is precisely the finite optimization stated in the main paper.

Corollary (non-override). If a feasible action preserves the proposal's type and target/support, its first two coordinates are zero. Every action that changes type or target/support has a positive first or second coordinate and cannot precede the preserving feasible bucket. IPFR therefore cannot return such an override.

Proposition 4 (honest refusal). If IPFR refuses, it has exhausted every finite bucket without observing a feasible action, hence none exists in $\mathcal{N}(a_0)$. Conversely, if none exists, every $\mathcal{F}_s$ is empty and the algorithm reaches NO-REPAIR.

E AC VALIDATION CONTRACT

The validator applies the following checks in fixed order:

1. action-grammar type, array length, branch-index, and online-generator checks;

2. declared redispatch balance to 10^{-5} MW;
3. pre-power-flow active generator limits and graph connectivity;
4. Newton AC power-flow convergence with tolerance 10^{-9} and at most 40 iterations;
5. voltage magnitudes in $[0.94, 1.06]$ p.u.;
6. declared reactive limits with a 1-Mvar numerical/model tolerance;
7. branch apparent-power ratios not exceeding one by more than 10^{-6};
8. scenario goal progress; and
9. post-power-flow active generator limits.

A passing witness stores cost, cost gap, goal-branch flow and reduction, maximum loading, voltage extrema, maximum active/reactive violation, and solved active generation. Failure stops at the first violated contract item, so its reason is deterministic and human-readable.

PYPOWER 5.1.19 under NumPy 2.3.5 does not reliably enforce reactive limits inside its Newton routine. The benchmark therefore declares scenario-local reactive reserves at least 60 Mvar around the accepted solved base point and checks those declared bounds externally. This workaround is a limitation, not a production recommendation.

F Scenario Construction

F.1 Base States

The four public MATPOWER/PYPOWER cases are IEEE 14, 30, 57, and 118. Positive loads receive clipped Gaussian multiplicative perturbations whose weighted mean is renormalized to one. Total generation is allocated within active limits to cover perturbed demand and the case's original loss ratio. For the larger grids, original heterogeneous voltage-setpoint profiles are retained under a common 0.98–1.02 multiplier. Every base state must itself converge and satisfy the declared voltage and active limits.

The stock IEEE-57 AC solution has minimum voltage approximately 0.936 p.u. Therefore IEEE-57 uses load multipliers uniformly in $[0.85, 0.95]$ while the other cases use $[1.05, 1.15]$. This keeps the voltage contract fixed. Each accepted branch receives a transparent synthetic emergency rating equal to its base apparent flow divided by a seed-fixed utilization sampled from $[0.25, 0.45]$.

F.2 Repair Neighborhood

For redispatch, the candidate list includes the proposal generator pair first, up to three alternatives, both directions for alternative pairs, and nine magnitude multipliers:

$$\{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2\}.$$

For switching, it includes the proposal branch plus up to four alternatives. Every proposal also receives type-changing candidates, so a cost-first method may select a feasible semantic override. All methods receive the same canonical list and cached validator outcomes.

F.3 Strata

Table 5: Final held-out scenario inventory.

Stratum	IEEE-57	IEEE-118	Total
Priority conflict, repairable	36	37	73
Unsafe switch target, repairable	18	16	34
Ordinary magnitude, repairable	0	1	1
Explicit refusal control	6	6	12
Total	60	60	120

A priority-conflict instance is accepted only when the proposal is unsafe, a same-type/same-target repair exists, another target has a feasible repair, and the latter has lower operating cost. This is a controlled stress stratum. The unsafe-switch proposal opens a true graph bridge; at least one non-bridge candidate is AC feasible. Refusal controls replace the goal threshold by 101% flow reduction after neighborhood construction, ensuring no candidate is admissible while preserving the same action grammar.

F.4 Audit Amendments

The initial development pilot showed that unconstrained redispatch generation did not create a cost–intent conflict. A timestamped amendment introduced the conditional stress criterion before held-out runs. The first held-out generation attempt then produced only 3/60 IEEE-57 scenarios. No method result file was written. Subsequent diagnostics found that forcing one common generator voltage setpoint destroyed the heterogeneous IEEE-57 profile and that increased load violated the fixed 0.94-p.u. base-state floor. The large-network normalizer and IEEE-57 load range were corrected before the complete held-out set existed. All amendment notes are in `audit/`.

Stage 3 review later identified a duplicated target/support distance and a first-feasible canonical stop that did not implement complete-bucket cost tie-breaking. The Stage 4 correction is explicitly post-held-out. We changed the method definition and implementation, added a target-versus-support unit test, regenerated development weights, and reran IEEE-57/118 from scratch. The dated note, original outputs, and corrected outputs are all included.

G Baselines and Development Selection

NoRepair and ClipBalance use one validator call. Reproposal samples a seed-fixed random permutation with the same call budget as IPFR. Euclidean, CostFirst, WeightedTuned, and LexOracle exhaust the candidate list. LexOracle minimizes the same five-coordinate tuple as IPFR and is used to verify action identity, not as an efficient method. WeightedTuned searches 81 combinations: type and target weights in $\{1, 10, 100\}$, support weight in $\{0.1, 1, 10\}$, magnitude weight fixed to one, and cost weight in $\{0.01, 0.1, 1\}$. Selection on 80 IEEE-14/30 development scenarios maximizes lexicographic agreement, then target retention, then minimizes mean cost. The selected vector is $(1, 1, 0.1, 1, 0.01)$.

H Complete Results

Table 6: All repairable held-out scenarios ($n = 108$).

Method	Feasible	Type+target	Mean calls
NoRepair	0.0%	100.0%†	1.00
ClipBalance	13.0%	100.0%†	1.00
Reproposal	45.4%	6.5%	3.70
CostFirst	100.0%	0.9%	60.07
Euclidean	100.0%	68.5%	60.07
WeightedTuned	100.0%	68.5%	60.07
LexOracle	100.0%	68.5%	60.07
IPFR	100.0%	68.5%	4.77

†The unmodified action retains its label but is unsafe; semantic repair success additionally requires feasibility.

IPFR's semantic repair success is 68.5% (scenario-bootstrap 95% CI 59.3–76.9). The 34 unsafe-bridge proposals cannot retain their exact target; IPFR attains the empirical feasible-retention ceiling of 74/108. On the 73 priority conflicts, IPFR succeeds semantically in 73 and CostFirst in zero. The paired success risk difference is 67.6 percentage points (bootstrap 95% CI 58.3–75.9), and the exact paired McNemar test gives $p = 2.12 \times 10^{-22}$. The median paired call saving against exhaustive feasibility projection is 61 (bootstrap 95% CI 61–62); a two-sided Wilcoxon signed-rank test gives

$p = 1.63 \times 10^{-19}$. Each registered outcome family has one paired test, so Holm correction leaves both values unchanged. LexOracle and IPFR select the identical action in 108/108 repairable cases.

Every exhaustive method and IPFR refuses all 12 controls. The no-refusal ablation emits an action in all 12, and all 12 fail validation. At ordinary units, WeightedTuned and Euclidean agree exactly with IPFR on all repairable cases. Rescaling the cost coordinate by 10^4 reduces WeightedTuned's exact agreement to 53.7% and target retention to 59.3%; IPFR is unchanged.

I Computing Environment and Reproduction

Experiments ran serially on an Apple M3 CPU (8 cores, 8 GB RAM) under macOS 15.5; no GPU or cloud instance was used. The environment records Python 3.12.13, NumPy 2.3.5, SciPy 1.16.1, pandas 2.2.3, PYPOWER 5.1.19, NetworkX 3.5, Matplotlib 3.10.5, and pytest 8.4.1. The metadata JSON also records the platform and elapsed time. The corrected Stage 4 development and held-out runs took 40.25 and 67.57 seconds, respectively, on this machine; these times are included for reproduction rather than cross-hardware comparison.

From the Code and Data Supplement root, first run the test suite; then execute `run_benchmark.py` on the development split with 40 cases per network, run `tune_weights.py`, execute the benchmark on the test split with 60 cases per network, and finally run `analyze_results.py`. Each script provides a command-line help message with the exact required paths. The supplied `README.md` gives complete paths and expected SHA-256 hashes. Regeneration does not require network access.

J Additional Limitations

The test is one-step and steady-state; AC feasibility is not transient or contingency security. Synthetic ratings make comparisons controlled but non-operational. Conditional priority conflicts demonstrate behavior when cost and intent disagree, not how often they disagree in a control room. Natural-language templates are interfaces to structured actions, not an evaluation of language understanding. Finally, the lexicographic order itself is a policy choice. A utility must obtain it from operators and governance, not silently adopt the order used in this benchmark.