

# Position Paper: Post-Solve Robustness in Decision Engines: Feasible Regions and Smoothness Under Perturbations

Yi-Xiang Hu

University of Science and Technology of China, Hefei, China, yixianghu@mail.ustc.edu.cn

**Abstract**—Mixed-Integer Linear Programming (MILP) decision engines routinely output nominally optimal plans for high-stakes industrial systems. Yet deployment rarely matches solve-time assumptions: small perturbations in costs, demands, or resource availability can invalidate feasibility or trigger discontinuous shifts to qualitatively different solutions. We argue that this post-solve robustness gap is a missing layer in today’s optimization pipelines and a missing evaluation dimension for learning-enabled decision systems. Rather than replacing robust optimization or stochastic programming, the proposed layer audits a solved incumbent and returns solver-backed evidence about how far that solution can be trusted. We formalize two central objects: (i) an  $\epsilon$ -near-optimal feasible neighborhood in parameter space, capturing when an incumbent remains feasible and near-optimal under perturbations, and (ii) solution smoothness in decision space, capturing whether nearby alternatives with small combinatorial edits remain competitive. We then synthesize the most relevant partial answers from sensitivity and stability analysis, robust optimization, neighborhood search, adversarial testing, and learning-based enhancements, and articulate an agenda for a unified post-solve robustness layer. Concretely, we call for certified inner approximations around the incumbent, probabilistic robustness estimation with calibrated uncertainty, adversarial robustness margins, and learning-based prediction and explanation aligned with solver-backed verification. We conclude with a compact reporting template and evaluation protocol that would make robustness a first-class output of decision engines.

**Index Terms**—Mixed-Integer Linear Programming, Robustness, Sensitivity Analysis, Robust Optimization, Machine Learning, Decision Support.

## I. INTRODUCTION

Modern decision support systems, referred to here as *decision engines*, increasingly rely on Mixed-Integer Linear Programming (MILP) solvers to produce plans in logistics, scheduling, finance, and energy. However, deployment rarely matches solve-time assumptions. Forecast errors, execution noise, and model mismatch perturb costs, demands, and resource availability after an incumbent plan is produced. In MILPs, even small perturbations can flip integer assignments, yielding discontinuous changes in discrete decisions that cause feasibility loss or abrupt shifts to qualitatively different solutions.

This deployment gap is familiar to practitioners. A routing plan that is optimal at solve time may become infeasible after a modest capacity reduction; a unit-commitment schedule may remain feasible but require a qualitatively different set of on/off decisions after a small forecast error. In such settings,

the nominal objective value is not enough. Users also need to know whether the incumbent remains trustworthy, which perturbations are most dangerous, and whether there are nearby fallback solutions that preserve most of the original value.

We therefore argue that MILP decision engines should not be evaluated or deployed without a *post-solve robustness layer* attached to the solved instance. The layer takes a nominal incumbent  $x^*$  and produces a compact robustness report: a certificate or lower bound on feasibility tolerance, a calibrated estimate of failure risk under realistic perturbations, a small set of critical failure modes, and a few high-quality fallback solutions. The intent is not to replace robust optimization (RO), stochastic programming, or receding-horizon control. Those methods act *before* or *during* optimization by redesigning the model or policy. In contrast, the post-solve layer acts *after* optimization by auditing the returned incumbent under a bounded latency budget and presenting the result in a form that supports deployment decisions.

Classical post-optimal sensitivity analysis is well established for linear programs (LPs): one can compute ranges for objective coefficients or right-hand side values within which the optimal basis remains optimal. Extending such analysis to MILPs is difficult because changes in integer variables induce discontinuous shifts in the optimal solution. Early research [1], [2] introduced sensitivity techniques tailored to branch-and-bound, but general MILP sensitivity does not admit LP-style “optimality ranges.” Instead, specialized guarantees have been explored. Dawande and Hooker [2] develop an inference-duality approach that derives linear conditions on parameter changes under which the optimal objective degrades by at most a prescribed tolerance.

A complementary line of work studies stability regions for incumbent MILP solutions. Yi and Lu [3] analyze how far an optimal MILP solution tolerates parameter variation while remaining valid, and local branching [4] reveals whether nearby combinatorial alternatives exist. Bridging sensitivity and robustness, the *radius of robust feasibility* captures the largest uncertainty magnitude for which a fixed solution remains feasible; Goberna et al. [5] survey this notion for linear and integer programs. Taken together, these strands provide valuable partial answers. What remains missing is a unified, solver-attached framework that converts them into a standardized report for deployment.

The central claim of this paper is therefore as practical as

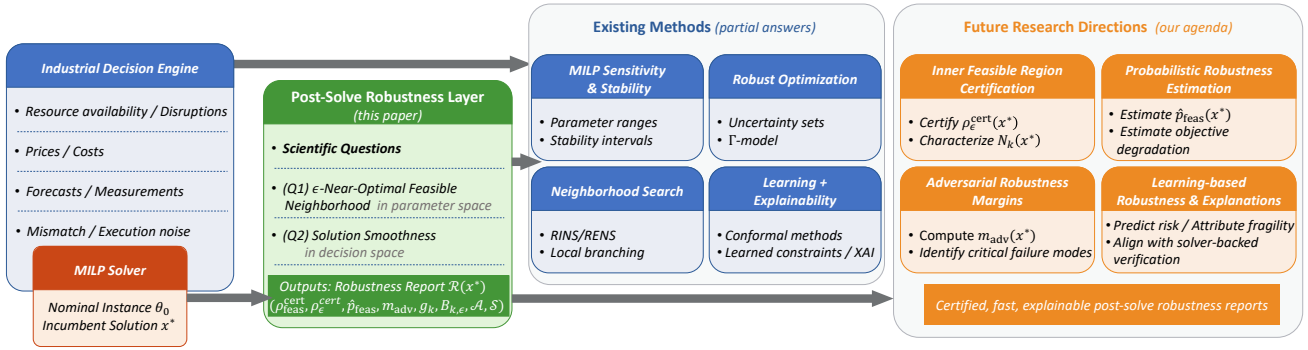


Fig. 1. Existing methods provide partial answers to post-solve robustness, while a dedicated post-solve robustness layer organizes the problem around two scientific questions: the  $\epsilon$ -near-optimal feasible neighborhood in parameter space and solution smoothness in decision space. The layer returns a structured robustness report  $\mathcal{R}(x^*)$  and motivates four future research directions toward certified, fast, and explainable post-solve robustness reports attached to a solved MILP incumbent.

it is conceptual: robustness should become a first-class solver output rather than an informal afterthought. To make that claim concrete, we formalize two core objects. The first is an  $\epsilon$ -near-optimal feasible neighborhood in parameter space, which captures when the incumbent remains feasible and near-optimal under perturbations. The second is solution smoothness in decision space, which captures whether nearby alternatives remain competitive under small combinatorial moves. Together, they provide a language for post-solve trustworthiness.

This paper makes four contributions for the position-paper setting. First, it sharpens the boundary between post-solve robustness and adjacent paradigms, especially RO and classical sensitivity analysis. Second, it formalizes the post-solve problem around certified feasible-and-near-optimal neighborhoods and decision-space smoothness. Third, it proposes a concrete reporting template that limits practitioner overload through a tiered summary. Fourth, it outlines a research agenda spanning inner feasible-region certification, probabilistic robustness estimation, adversarial boundary search, and learning-based prediction and explanation.

The rest of the paper is organized as follows. Section II defines the problem formally. Section III reviews current approaches and gaps. Section IV describes future research directions. Section V discusses industrial relevance, and Section VI concludes.

## II. PROBLEM DEFINITION AND FORMULATION

We consider a parametric MILP

$$z(\theta) = \min_x \{f(x; \theta) : x \in \mathcal{F}(\theta) \cap \mathcal{X}\}, \quad (1)$$

where  $\theta \in \Theta$  collects all perturbable inputs,  $\mathcal{F}(\theta)$  denotes the linear constraints induced by  $\theta$ , and  $\mathcal{X}$  encodes integrality restrictions. Let  $\theta_0$  be the nominal parameter vector and let  $x^*$  be an incumbent optimal solution for  $\theta_0$ . A *post-solve robustness layer* is a procedure that takes  $(\theta_0, x^*)$  and returns a bounded-cost robustness report for deployment.

**Definition 1 ( $\epsilon$ -near-optimal feasible neighborhood in parameter space).** Let  $d_\theta$  be a distance on the parameter space.

For any radius  $\rho \geq 0$ , define the perturbation ball around  $\theta_0$  by

$$\mathcal{B}_\rho(\theta_0) = \{\theta \in \Theta : d_\theta(\theta, \theta_0) \leq \rho\}. \quad (2)$$

To separate pure feasibility from near-optimality, define

$$\mathcal{U}_{\text{feas}}(x^*) = \{\theta \in \Theta : x^* \in \mathcal{F}(\theta) \cap \mathcal{X}\}, \quad (3)$$

and, for tolerance  $\epsilon \geq 0$ ,

$$\mathcal{U}_\epsilon(x^*) = \{\theta \in \Theta : x^* \in \mathcal{F}(\theta) \cap \mathcal{X}, f(x^*; \theta) \leq z(\theta) + \epsilon\}. \quad (4)$$

Thus,  $\mathcal{U}_{\text{feas}}(x^*)$  is the set of perturbed instances for which the incumbent remains feasible, while  $\mathcal{U}_\epsilon(x^*)$  is the set for which the incumbent remains feasible and at most  $\epsilon$ -suboptimal. The special case  $\epsilon = 0$  corresponds to exact incumbent optimality under perturbation.

These sets induce two natural certification targets:

$$\rho_{\text{feas}}^{\text{cert}}(x^*) = \sup\{\rho \geq 0 : \mathcal{B}_\rho(\theta_0) \subseteq \mathcal{U}_{\text{feas}}(x^*)\}, \quad (5)$$

and

$$\rho_\epsilon^{\text{cert}}(x^*) = \sup\{\rho \geq 0 : \mathcal{B}_\rho(\theta_0) \subseteq \mathcal{U}_\epsilon(x^*)\}. \quad (6)$$

The first quantity is a certified feasibility radius, and the second is a certified near-optimality radius. Both serve as solver-backed lower bounds on how far the incumbent can be trusted under the chosen perturbation model. The feasibility radius is closely related to the robust-feasibility radius surveyed by Goberna et al. [5].

**Definition 2 (Solution smoothness in decision space).** Let  $d_x$  be a distance on feasible solutions; for binary or integer decisions, a Hamming distance is a natural choice. For an integer  $k \geq 1$ , define the local neighborhood

$$N_k(x^*) = \{x \in \mathcal{F}(\theta_0) \cap \mathcal{X} : d_x(x, x^*) \leq k\}. \quad (7)$$

Smoothness should capture whether the incumbent is isolated or supported by nearby competitive alternatives. Two useful quantities are

$$g_k(x^*) = \begin{cases} \min_{x \in N_k(x^*) \setminus \{x^*\}} (f(x; \theta_0) - f(x^*; \theta_0)), & \text{if } N_k(x^*) \setminus \{x^*\} \neq \emptyset, \\ +\infty, & \text{otherwise.} \end{cases} \quad (8)$$

and

$$B_{k,\epsilon}(x^*) = |\{x \in N_k(x^*) : f(x; \theta_0) \leq f(x^*; \theta_0) + \epsilon\}|. \quad (9)$$

Here,  $g_k(x^*)$  is the local backup gap and  $B_{k,\epsilon}(x^*)$  is the local backup count. A small backup gap and a large backup count indicate a smooth local landscape. A large backup gap or an empty neighborhood indicates fragility. These quantities are more aligned with operational fallback planning than a generic local-optimality score because they measure both the quality and the availability of nearby substitutes.

These definitions suggest a concrete output object. Let  $\mathcal{D}$  denote a stated perturbation distribution over  $\Theta$  that reflects the intended deployment regime. Given an incumbent  $x^*$ , the post-solve layer should return a report

$$\mathcal{R}(x^*) = (\rho_{\text{feas}}^{\text{cert}}, \rho_{\epsilon}^{\text{cert}}, \hat{p}_{\text{feas}}, m_{\text{adv}}, g_k, B_{k,\epsilon}, \mathcal{A}, \mathcal{S}), \quad (10)$$

where

$$\hat{p}_{\text{feas}}(x^*) = \Pr_{\theta \sim \mathcal{D}} [x^* \in \mathcal{F}(\theta) \cap \mathcal{X}] \quad (11)$$

is the estimated probability that the incumbent remains feasible under the perturbation model, and

$$m_{\text{adv}}(x^*) = \inf \{d_{\theta}(\theta, \theta_0) : \theta \in \Theta, \theta \notin \mathcal{U}_{\text{feas}}(x^*)\} \quad (12)$$

is the smallest perturbation magnitude that invalidates incumbent feasibility. In the basic report template,  $m_{\text{adv}}(x^*)$  refers to the feasibility margin; Section IV-C further distinguishes feasibility and  $\epsilon$ -near-optimality margins. The remaining fields represent a local backup-gap metric, a local backup-count metric, explanatory artifacts  $\mathcal{A}$  such as binding constraints or dominant risk factors, and fallback solutions  $\mathcal{S}$ . Not every deployment requires every field, but this formulation makes the agenda operational: post-solve robustness is not a vague property, but a structured report produced under a post-solve time budget.

The main difficulty is computational. Even membership in  $\mathcal{U}_0(x^*)$  can be hard to verify because one must rule out better competing integer solutions under perturbation. Likewise, estimating  $g_k(x^*)$  or  $B_{k,\epsilon}(x^*)$  may require auxiliary restricted solves. This is precisely why a dedicated research agenda is needed.

### III. CURRENT APPROACHES AND CHALLENGES

Existing methods address fragments of post-solve robustness, yet they rarely compose into a concise, standardized layer attached to a solved MILP. In practice, users need four things at once: trust-region information, failure risk, likely break points, and actionable backups. The current literature usually addresses only one or two of these requirements at a time.

#### A. MILP Sensitivity Analysis and Stability Regions

LP sensitivity analysis provides ranges in which the current basis remains optimal. For MILPs, robustness is harder to characterize because small data changes can flip discrete decisions. Early work introduced sensitivity rules tailored to branch-and-bound, tracking how relaxations and node bounds

evolve under parameter variations [1]. Dawande and Hooker proposed inference-based sensitivity analysis that derives linear conditions on perturbations guaranteeing that objective degradation remains within a prescribed tolerance [2]. These methods are valuable, but they are typically parameter-specific and do not directly produce a deployment-facing report.

Stability-region analysis focuses on the incumbent integer solution itself. Yi and Lu compute coordinate-wise stability intervals by varying one parameter at a time until the incumbent loses optimality [3]. This yields practical diagnostics, yet multi-parameter coupling remains difficult and discrete switches can occur abruptly. Robust-feasibility-radius analysis shifts attention from optimality to feasibility and measures the largest perturbation magnitude under which a fixed solution remains feasible [5]. For general MILPs, however, tight radii remain hard to compute at scale.

#### B. Proactive Robust Optimization Versus Post-Solve Analysis

RO incorporates uncertainty *during* optimization by enforcing feasibility for all realizations in a specified uncertainty set [6], [7]. This yields built-in protection, and the budgeted uncertainty model of Bertsimas and Sim reduces conservatism through the parameter  $\Gamma$  [6]. RO is indispensable when uncertainty sets are known and the extra computational burden is acceptable.

Our claim is narrower. Post-solve robustness does not replace RO. It complements RO in three common cases: when practitioners still solve a nominal model for speed or modeling simplicity; when uncertainty sets are not available at solve time; and when operators need an audit of a realized incumbent before committing to execution. Under this view, RO changes the optimization problem, whereas post-solve robustness evaluates the returned solution under an explicit latency budget and communicates the result in decision-oriented terms.

#### C. Neighborhood Search and Inner Feasible Approximations

Neighborhood search probes local structure around the incumbent. Local branching restricts Hamming distance to the incumbent and re-solves the MILP to find nearby alternatives or certify local optimality within that neighborhood [4]. Related strategies such as RINS and RENS construct neighborhoods by fixing subsets of variables and re-optimizing the remainder [8], [9]. These methods are powerful for discovering nearby alternatives, but they are usually used as improvement heuristics rather than as standardized robustness diagnostics.

More formal work studies inner approximations that guarantee feasibility around an integer point [10]. These constructions align closely with our notion of solver-backed certification. The gap is not conceptual possibility, but packaging: current tools are not usually exposed as a compact certificate, a risk score, and a ranked fallback list suitable for a human operator.

#### D. Simulation and Worst-Case Evaluation

Scenario-based evaluation tests the incumbent under sampled perturbations to estimate feasibility rates and objective variation. This is practical and distribution-aware, but it provides limited guarantees. Worst-case robustness evaluation

searches for an adversarial perturbation within prescribed bounds and connects naturally to robust-optimization separation [11]. It can identify concrete breaking scenarios, but the resulting auxiliary problems may themselves be expensive.

### E. Learning-Based Enhancements and Explainability

Learning-based approaches can accelerate robustness assessment, fill in missing structure, and improve interpretability. Constraint-learning surveys formalize how data augments optimization models while controlling feasibility risk [12]. Conformal mixed-integer constraint learning provides probabilistic feasibility guarantees for learned constraints [13]. Related work integrates neural constraints and decomposition ideas for scalable mixed-integer optimization [14], [15]. On the interaction side, recent systems explore explaining optimization outcomes to practitioners and generating contrastive explanations via constraint reasoning [16], [17], while surveys discuss explainable AI in broader decision-making pipelines [18]. These directions improve usability, but they must remain aligned with solver-backed verification to avoid unsupported robustness claims.

### F. Why Existing Pieces Do Not Yet Form a Solver Output

The missing piece is integration. Sensitivity analysis offers local structure but struggles with joint perturbations. RO offers protection but requires uncertainty sets and changes the optimization problem. Neighborhood search finds backups but does not standardize robustness metrics. Simulation and adversarial testing expose failures but may lack certificates or latency discipline. Learning-based methods accelerate assessment and explanation, but they require calibration and verification. A post-solve robustness layer must combine these pieces into a budgeted workflow and expose a small number of operator-facing outputs.

### G. A Call to Action: Robustness as a First-Class Output and Evaluation Protocol

We call for a shift from treating MILP solvers as black boxes that return a single incumbent to treating them as decision engines that return *an incumbent plus a robustness report*. To avoid overwhelming practitioners, the report should be tiered.

**Tier 1: compact summary.** Four scalar or short-form outputs should appear by default: (i) a certified lower bound on feasibility tolerance, (ii) a calibrated risk estimate under a stated perturbation model, (iii) a smallest adversarial perturbation or ranked critical failure mode, and (iv) one to three near-optimal fallback solutions. Concretely, Tier 1 may expose  $(\rho_{\text{feas}}^{\text{cert}}, \hat{p}_{\text{feas}}, m_{\text{adv}}, \mathcal{S}_{\text{top}})$ , where  $\mathcal{S}_{\text{top}} \subseteq \mathcal{S}$  contains one to three fallback solutions.

**Tier 2: diagnostic details.** For analysts, the engine should expose binding constraints, dominant perturbation directions, neighborhood-search statistics, and sensitivity traces.

**Tier 3: full audit.** For debugging or offline model refinement, the engine may return deeper adversarial traces, scenario rollouts, or solver logs.

This structure also addresses standardization. We advocate a *core metric layer* shared across domains and a *domain policy*

*layer* defined by application owners. The core layer contains solver-agnostic quantities such as feasibility radius, adversarial margin, local backup gap, and empirical failure probability. The domain layer specifies what counts as acceptable risk, which perturbation norms matter, and what action should follow a brittle report. This split allows logistics, energy, and finance to compare methods on common robustness axes while preserving domain-specific safety thresholds.

Finally, every report should be produced under an explicit post-solve time budget  $\tau_{\text{post}}$ . Some fields may be certified exactly, some may be lower bounds, and some may be statistical estimates. This is a feature rather than a defect: a budgeted, anytime report is more useful in practice than an ideal report that arrives too late.

### H. Evaluation Protocol

A post-solve robustness layer should be evaluated as an attached decision service rather than as a standalone predictor. Starting from nominal MILP instances and incumbent solutions  $x^*$ , one generates perturbed instances from a stated deployment model  $D$  over  $\Theta$ . For each nominal solve, the post-solve layer returns a report  $R(x^*)$  under a post-solve budget  $\tau_{\text{post}}$ , and this report is compared against the realized outcomes on the perturbed instances.

Four dimensions are central. First, *effectiveness* measures feasibility retention, near-optimality retention, and fallback quality under perturbation. Second, *calibration* compares estimated risks, such as  $\hat{p}_{\text{feas}}(x^*)$ , with empirical frequencies and checks whether certified radii such as  $\rho_{\text{feas}}^{\text{cert}}$  and  $\rho_{\epsilon}^{\text{cert}}$  match observed coverage. Third, *efficiency* measures latency, auxiliary solver effort, and the trade-off between report completeness and time budget. Fourth, *actionability* measures whether the report improves deployment decisions relative to returning the nominal incumbent alone.

## IV. FUTURE RESEARCH DIRECTIONS

We outline four promising directions for turning the above agenda into a practical post-solve layer.

### A. Inner Approximation of Feasible Regions Around Solutions

A first direction is to characterize the local feasible region around the incumbent  $x^*$ . The goal is to construct an inner approximation of the feasible set near  $x^*$  and to quantify, through neighborhoods such as  $N_k(x^*)$ , whether the incumbent is supported by nearby alternatives with comparable objective values. This turns local structure into an explicit robustness object rather than an implicit by-product of solving.

A practical mechanism is to impose neighborhood constraints and re-solve, as in local branching [4]. Increasing the neighborhood size  $k$  reveals how objective quality changes as one moves away from  $x^*$  and whether the feasible set around  $x^*$  is dense or sparse. Efficient exploration can build on RINS and RENS [8], [9] to discover alternatives of similar quality. A complementary track is to derive certified inner regions through polyhedral constructions [10]. An important open problem is balancing certificate strength, interpretability, and

computational cost so that local robustness reporting becomes routine for large MILPs.

### B. Probabilistic Robustness Estimation

Worst-case guarantees are valuable, but many deployments accept solutions that succeed with high probability under realistic uncertainty. A second direction is therefore to estimate quantities such as  $\hat{p}_{\text{feas}}(x^*)$  and the distribution of objective degradation under a calibrated perturbation model. Scenario evaluation provides a baseline, but future work should improve sample efficiency, concentrate computation near failure boundaries, and provide principled uncertainty quantification.

This direction is especially relevant when exact certification is too expensive. A well-calibrated probability of success is often operationally meaningful, provided the perturbation model is explicit and the estimate is accompanied by confidence information. It also creates a natural interface with online monitoring and recurring solves.

### C. Adversarial Perturbation and Robustness Margins

Complementing probabilistic evaluation, an adversarial perspective seeks the smallest perturbation that breaks the incumbent. Using the notation of Section II, the natural target is the smallest parameter-space displacement that moves the instance outside the trusted region of  $x^*$ . For a tolerance  $\epsilon \geq 0$ , define the adversarial near-optimality margin

$$m_{\text{adv}}^\epsilon(x^*) = \inf \{d_\theta(\theta, \theta_0) : \theta \in \Theta, \theta \notin \mathcal{U}_\epsilon(x^*)\}. \quad (13)$$

Equivalently,  $m_{\text{adv}}^\epsilon(x^*)$  is the smallest perturbation magnitude under which the incumbent becomes infeasible or more than  $\epsilon$ -suboptimal. When only feasibility matters, one obtains the feasibility margin

$$m_{\text{adv}}^{\text{feas}}(x^*) = \inf \{d_\theta(\theta, \theta_0) : \theta \in \Theta, \theta \notin \mathcal{U}_{\text{feas}}(x^*)\}. \quad (14)$$

These quantities provide interpretable robustness margins. They also identify adversarial instances

$$\theta_{\text{adv}}^\epsilon \in \arg \min_{\theta \in \Theta, \theta \notin \mathcal{U}_\epsilon(x^*)} d_\theta(\theta, \theta_0), \quad (15)$$

or feasibility-only analogues, which expose the most vulnerable direction in parameter space.

This direction is related to parametric optimization [3], but it targets coupled multi-parameter failures rather than one-parameter thresholds. Feasibility margins can begin with slack-based diagnostics and solver-assisted verification. Near-optimality margins are harder because one must identify perturbations under which competing solutions overtake  $x^*$ . Practical approaches therefore combine restricted adversarial search, near-optimal alternative enumeration, and targeted auxiliary solves. The output should remain concise: an estimated breaking margin, a representative adversarial instance, the associated dominant constraints or risk factors, and a recommended response such as repair, re-optimization, or operator escalation.

### D. Learning-Based Robustness Prediction and Explanation

A final direction is to integrate machine learning so that post-solve evaluation and communication become faster and more actionable. Surrogate models can predict robustness indicators from instance features, incumbent features, and solver artifacts such as slack profiles or relaxation information. This supports rapid screening when similar MILPs are solved repeatedly.

Learning can also reduce model mismatch by augmenting constraints or penalties from data. Conformal methods provide one principled route for incorporating learned components with statistical guarantees [13], and broader work on optimization with learned components frames this interface cleanly [12]. The crucial requirement is that learned estimates remain coupled to solver-backed verification.

Finally, learning should improve explanations. Robustness reports are most useful when they say not only that a solution is brittle, but why it is brittle and what can be done next. Contrastive explanation mechanisms based on infeasibility reasoning provide a promising foundation [17]. In practice, this means prioritizing the constraints, parameters, and structural choices most responsible for fragility, while still grounding the final claim in explicit solver checks.

## V. INDUSTRIAL IMPACT AND USE CASES

Post-solve robustness is driven by deployment requirements. Industrial decision engines operate under forecast error, execution noise, and model mismatch. In these settings, an operator does not only ask whether a solution is optimal; the operator asks whether it is safe to execute, how likely it is to fail, and what to do if it is brittle. Across domains, the common bottleneck is not only solving the nominal MILP, but deciding whether the solved incumbent is safe to execute under bounded uncertainty.

### A. Supply Chain and Logistics

Logistics applications such as routing, network design, and inventory planning face uncertainty in demand, travel time, and cost. RO can produce plans that remain feasible under demand fluctuations [19]. In pipeline transportation, Moradi and MirHassani show that  $\Gamma$ -robust scheduling improves feasibility under demand uncertainty [20]. A post-solve layer complements these methods by quantifying how much a nominal plan can absorb before repair becomes necessary and by surfacing limited-change fallback plans.

### B. Energy Systems

Power systems require schedules that tolerate uncertainty in load and renewable generation. Robust formulations for unit commitment improve feasibility under forecast error and reduce reliance on emergency actions [11], [21]. A post-solve robustness layer adds a deployment-facing audit: how far the current commitment can drift before violating reserve or ramping constraints, which forecast components are most dangerous, and whether low-regret recourse plans exist nearby.

### C. Manufacturing and Scheduling

Manufacturing and project scheduling face variability in processing times and resource availability [22]. Robust scheduling introduces slack and flexibility to avoid infeasibility cascades under disruptions [21]. In practice, however, managers often prefer minimal schedule edits over complete re-optimization. This makes decision-space smoothness especially relevant because it quantifies whether small repairs preserve most of the nominal objective.

### D. Finance and Portfolio Optimization

Portfolio selection via MILP or MIQP is sensitive to uncertainty in returns and market conditions. Robust and learning-augmented formulations aim to preserve feasibility and reduce instability under adverse outcomes [12], [13]. A post-solve layer complements these formulations by identifying how quickly the incumbent loses feasibility or near-optimality as assumptions drift and by explaining which exposures drive fragility.

## VI. CONCLUSION

This paper argues that nominal optimality is an incomplete output for high-stakes MILP decision engines. We sharpened the notion of post-solve robustness as a solver-attached audit layer that complements rather than replaces robust optimization and classical sensitivity analysis. We formalized two core objects, an  $\epsilon$ -near-optimal feasible neighborhood in parameter space and solution smoothness in decision space, and translated them into a concrete robustness-report template built around certificates, risk estimates, adversarial margins, and fallback solutions.

The main message is actionable. When a solver reports that an incumbent is brittle, the system should not stop at that warning. It should also indicate the dominant failure mode, expose nearby repairs when they exist, and signal when escalation to re-optimization or a more explicitly robust model is necessary. This is the practical meaning of making robustness a first-class output.

We therefore view post-solve robustness as both a research problem and an evaluation problem. Scientifically, it requires new methods for certified local analysis, probabilistic estimation, adversarial search, and learning-based explanation. Operationally, it requires budgeted reporting standards that are informative without overwhelming practitioners. At a minimum, future decision engines should expose a certified tolerance, a calibrated risk estimate, a breaking scenario, and a limited-change fallback recommendation alongside the incumbent. Establishing this interface would improve the reliability, explainability, and deployability of optimization-based decision engines.

## ACKNOWLEDGMENTS

I acknowledge Shang-Hua Teng, Xiang-Yang Li, Feng Wu, and the anonymous reviewers for their comments and helpful feedback.

## REFERENCES

- [1] L. Schrage and L. Wolsey, "Sensitivity analysis for branch and bound integer programming," *Operations Research*, vol. 33, no. 5, pp. 1008–1023, 1985.
- [2] M. Dawande and J. N. Hooker, "Inference-based sensitivity analysis for mixed integer/linear programming," *Operations Research*, vol. 48, no. 4, pp. 623–634, 2000.
- [3] C. Yi and M. Lu, "Mixed-integer linear programming-based sensitivity analysis in optimization of temporary haul road layout design for earthmoving operations," *Journal of Computing in Civil Engineering*, vol. 33, no. 3, p. 04019021, 2019.
- [4] M. Fischetti and A. Lodi, "Local branching," *Mathematical programming*, vol. 98, no. 1, pp. 23–47, 2003.
- [5] M. A. Goberna, V. Jeyakumar, G. Li, and J. Vicente-Pérez, "The radius of robust feasibility of uncertain mathematical programs: a survey and recent developments," *European Journal of Operational Research*, vol. 296, no. 3, pp. 749–763, 2022.
- [6] D. Bertsimas and M. Sim, "The price of robustness," *Operations research*, vol. 52, no. 1, pp. 35–53, 2004.
- [7] A. Ben-Tal, A. Nemirovski, and L. El Ghaoui, "Robust optimization," 2009.
- [8] E. Danna, E. Rothberg, and C. L. Pape, "Exploring relaxation induced neighborhoods to improve mip solutions," *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.
- [9] T. Berthold, "Rens-relaxation enforced neighborhood search," 2007.
- [10] M. Fischetti, A. Lodi *et al.*, "Heuristics in mixed integer programming," *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc, pp. 2–23, 2010.
- [11] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive robust optimization for the security constrained unit commitment problem," *IEEE transactions on power systems*, vol. 28, no. 1, pp. 52–63, 2012.
- [12] A. O. Fajemisin, D. Maragno, and D. den Hertog, "Optimization with constraint learning: A framework and survey," *European Journal of Operational Research*, vol. 314, no. 1, pp. 1–14, 2024.
- [13] D. Ovalle, L. T. Biegler, I. E. Grossmann, C. D. Laird, and M. D. Rubio, "Conformal mixed-integer constraint learning with feasibility guarantees," *arXiv preprint arXiv:2506.03531*, 2025.
- [14] S. Zeng, S. Zhang, F. Wu, S. Tang, and X. Li, "Scalable mixed-integer optimization with neural constraints via dual decomposition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 17, 2026, pp. 14 388–14 396.
- [15] Y.-X. Hu, F. Wu, S. Li, Y. Zhao, and X.-Y. Li, "Ffcg: Effective and fast family column generation for solving large-scale linear program," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 11, pp. 11 238–11 245, Apr. 2025.
- [16] H. Chen, G. E. Constante-Flores, K. S. I. Mantri, S. M. Kompalli, A. S. Ahluwalia, and C. Li, "Optichat: Bridging optimization models and practitioners with large language models," *INFORMS Journal on Data Science*, 2025.
- [17] R. X. Lera-Leri, F. Bistaffa, A. Georgara, and J. A. Rodríguez-Aguilar, "Exploiting constraint reasoning to build graphical explanations for mixed-integer linear programming," in *International Workshop on Explainable, Trustworthy, and Responsible AI and Multi-Agent Systems*. Springer, 2025, pp. 21–39.
- [18] K. Danach, W. H. F. Aly, A. Tarhini, and S. Laouadi, "Toward transparent optimization: A systematic review of explainable ai in decision-making systems," *European Journal of Pure and Applied Mathematics*, vol. 18, no. 4, pp. 6707–6707, 2025.
- [19] X. Hu, J. Lee, and J. Lee, "Two-stage predict+ optimize for milps with unknown parameters in constraints," *Advances in neural information processing systems*, vol. 36, pp. 14 247–14 272, 2023.
- [20] S. Moradi and S. MirHassani, "Robust scheduling for multi-product pipelines under demand uncertainty," *The International Journal of Advanced Manufacturing Technology*, vol. 87, no. 9, pp. 2541–2549, 2016.
- [21] H. Yang, D. P. Morton, C. Bandi, and K. Dvijotham, "Robust optimization for electricity generation," *INFORMS Journal on Computing*, vol. 33, no. 1, pp. 336–351, 2021.
- [22] Y.-X. Hu, Y. Wang, F. Wu, Z. Huang, S. Zeng, and X.-Y. Li, "ischeduler: Reinforcement learning-driven continual optimization for large-scale resource investment problems," *arXiv preprint arXiv:2602.06064*, 2026.