CLARA: Solver-Intrinsic Explanations and Reoptimization for Linear Programming

Yoonsik Jung
Department of Industrial Engineering, Korea University, Seoul, Republic of Korea
yoonsik@korea.ac.kr

March 24, 2026 — Draft

Abstract

Linear programming (LP) solvers produce optimal solutions, but the reasoning behind these solutions remains opaque to practitioners. While explainable AI (XAI) methods have been developed for machine learning models, explainability for optimization solvers has received little attention, and existing approaches rely on ML models as intermediaries. We present CLARA (Classical LP Analysis for Reoptimization and Attribution), an open-source LP solver that extracts explanations directly from the simplex basis inverse — binding analysis, objective change attribution, simultaneous sensitivity regions, and automatic reoptimization decisions — without any ML approximation. The attribution module decomposes objective changes into per-parameter contributions using first-order analysis and Shapley values. The simultaneous sensitivity region, computed via a Chebyshev center LP, quantifies how much one-at-a-time analysis overestimates the safe perturbation region. The reoptimization pipeline automatically classifies parameter changes, assesses their impact, and selects the appropriate warm-start method. Computational experiments on 120 random LPs and 7 Netlib instances demonstrate solver correctness (optimality gap $< 1.5 \times 10^{-12}$), warm-start speedup (mean 36.6×), and that one-at-a-time sensitivity overestimates safe regions by a factor of $\sim$10× on average. CLARA is available at `https://pypi.org/project/clara-opt/` under the MIT license.

Keywords: linear programming, sensitivity analysis, explainable optimization, reoptimization, open-source software

1 Introduction

Linear programming (LP) is a cornerstone of operations research, with applications spanning supply chain management, production planning, transportation, and finance (Bertsimas and Tsitsiklis, 1997). Modern solvers such as CPLEX, Gurobi, and HiGHS can solve problems with millions of variables in seconds. Yet the solutions they produce are often perceived as black-box outputs: a practitioner receives an optimal production plan, fleet schedule, or resource allocation, but gains little insight into the reasoning behind it. Three recurring questions arise in practice.

Why is this the answer? Commercial solvers report shadow prices and reduced costs—the classical outputs of sensitivity analysis (Dantzig, 1963)—but these are one-at-a-time (OAT) quantities: each measures the effect of changing a single parameter while holding all others fixed. When multiple parameters are uncertain simultaneously, OAT ranges can be misleading (Koltai and Terlaky, 2000). The tolerance approach (Wendell, 1985) and its extensions (Filippi, 2005; Borgonovo et al., 2018) provide the theory for simultaneous sensitivity analysis, but as Koltai and Tatay (2011)

observe, this theory is "not implemented yet in existing commercial LP packages—a shortcoming of software, not theory."

What happens when parameters change? Beyond stability analysis, practitioners need to understand *how much* of the change in the optimal value is attributable to each parameter. No commercial solver provides objective change attribution—a decomposition of Δz into per-parameter contributions—and the simultaneous sensitivity region within which the current basis remains optimal has never been implemented in a publicly available tool, despite four decades of theoretical development. Meanwhile, recent work on counterfactual explanations (CEs) for optimization (Korikov et al., 2021; Kurtz et al., 2025) addresses the complementary question—"what minimal change would lead to a different outcome?"—but the mathematical relationship between the sensitivity region (factual) and the CE distance (counterfactual) has not been formalized.

Should I re-solve? In model predictive control, column generation, rolling horizon planning, and branch-and-bound, LP instances are solved repeatedly with slight parameter modifications (Bolusani et al., 2024; Rawlings et al., 2017; Lübbecke and Desrosiers, 2005). Practitioners either always re-solve (potentially wasting effort on negligible changes) or skip re-solving based on ad-hoc thresholds, without theoretical guarantees on the resulting suboptimality. The only available bound on the cost of neglecting reoptimization is due to Oguz (2000), but no tool integrates this bound into an automatic reoptimization decision framework.

Recent work in explainable AI (XAI) for optimization has begun to address the first two questions. Approaches include inherently interpretable models (Goerigk and Hartisch, 2023), neural LP encodings (Busch et al., 2025), and counterfactual explanations via inverse optimization (Korikov et al., 2021; Kurtz et al., 2025). However, all of these use ML models or inverse formulations as intermediaries, introducing approximation where exact analysis is available. None directly leverages the solver's internal structure—the simplex basis inverse B^{-1}, which already contains shadow prices, reduced costs, and the information needed for simultaneous sensitivity and attribution, all computed as byproducts of the simplex method with no additional approximation. Moreover, no existing work has formalized the connection between factual explanations (sensitivity regions) and counterfactual explanations (minimal basis-changing perturbations), despite Kurtz et al. (2025) observing that "sensitivity analysis can be seen as factual explanation."

In this paper, we present CLARA (Classical LP Analysis for Reoptimization and Attribution), a framework that extracts explanations *directly from* B^{-1} and unifies factual and counterfactual perspectives on LP explainability. Our contributions are as follows:

1. **Solver-intrinsic explanation reports:** binding constraint analysis (shadow prices, utilization), variable contribution analysis (reduced costs), and sensitivity classification (bottleneck, fragile, or robust), all extracted directly from B^{-1} (Section 4.1);
2. **Objective change attribution:** a first-order decomposition of objective value changes into RHS, objective, and interaction effects, complemented by Shapley values when the basis changes (Section 4.2);
3. **Simultaneous sensitivity regions:** the Chebyshev center of the basis-preserving polyhedron, providing the first publicly available implementation of simultaneous sensitivity analysis and quantifying how much OAT analysis overestimates the safe perturbation region (Section 4.3);
4. **Factual–counterfactual duality:** a formal proof that the basis robustness d_0—computable in $O(mn)$ from B^{-1} alone—is a lower bound on the cost of any basis-changing counterfactual explanation, unifying sensitivity analysis and counterfactual explanations within a single framework (Section 4.4);
5. **Reoptimization pipeline:** automatic change detection, impact analysis via the Oguz bound, warm-start method selection, and structured diff reports (Section 5);

with a comprehensive computational study on 127 LP instances and ∼3,500 perturbation pairs validating solver correctness (optimality gap $< 1.5 \times 10^{-12}$), warm-start speedups (mean 36.6×), and the finding that OAT sensitivity overestimates the safe region by ∼10× on average. CLARA is available as open-source software under the MIT license.

The remainder of this paper is organized as follows. Section 2 reviews related work in sensitivity analysis, reoptimization, and explainable optimization. Sections 3–5 describe the CLARA framework, explanation modules, and reoptimization pipeline. Section 6 reports computational results, and Section 8 concludes.

2 Related Work

2.1 Sensitivity analysis in linear programming

Sensitivity analysis answers the question "how does the optimal solution change when problem parameters change?" The standard approach, implemented in every commercial LP solver, is *one-at-a-time* (OAT) analysis: for each right-hand-side coefficient b_i or objective coefficient c_j, the solver reports the range within which that single parameter can vary—holding all others fixed—without changing the optimal basis. The resulting shadow prices and reduced costs, originating from Dantzig (1963), remain the most widely used sensitivity information in practice.

OAT analysis, however, can be misleading when multiple parameters change simultaneously. Individual ranges may each appear safe, yet a combination of changes within those ranges can cause a basis change. Wendell (1985) introduced the *tolerance approach* to address this limitation. The approach computes a maximum tolerance percentage τ^* such that, as long as all selected coefficients are accurate to within τ^* percent of their estimated values, the same basis remains optimal—even under simultaneous and independent perturbations. Ravi and Wendell (1989) extended the tolerance approach to matrix coefficients, allowing simultaneous perturbations of entries in the constraint matrix A, not just b and c. Ward and Wendell (1990) provide a comprehensive survey of sensitivity approaches, comparing OAT, parametric, and tolerance methods along the dimensions of informativeness, ease of use, and computational tractability.

Subsequent work refined and generalized the tolerance framework. Filippi (2005) proposed a geometric algorithm that computes individual tolerances iteratively, yielding tolerance regions that are *maximal with respect to inclusion*—strictly larger than those obtained by Wendell's original approach in degenerate cases. Most recently, Borgonovo et al. (2018) merged the tolerance approach with Wagner's global sensitivity analysis, enabling the analyst to simultaneously address questions of stability, trend, model structure, and data prioritization for joint variations in both b and c.

Despite four decades of theoretical progress, the practical adoption of simultaneous sensitivity analysis has been limited. Koltai and Terlaky (2000) highlight the gap between the managerial and mathematical interpretations of sensitivity results: practitioners often misinterpret OAT ranges as valid for simultaneous changes, leading to flawed decisions. Koltai and Tatay (2011) propose a practical approach to sensitivity analysis under degeneracy—where shadow prices are not unique—and make the pointed observation that "there is a consistent mathematical theory ... not implemented yet in existing commercial LP packages." This remains true today: CPLEX, Gurobi, and HiGHS all provide OAT sensitivity reports, but none offers simultaneous sensitivity regions, tolerance percentages, or objective change attribution.

Parametric programming (Gal, 1979) provides the most complete theoretical treatment, tracing the optimal value as a piecewise-linear function of a parameter θ and identifying all basis-change breakpoints. While mathematically elegant, parametric analysis is available only in specialized educational software and is absent from the APIs of major commercial solvers.

In summary, LP sensitivity analysis theory is mature—from OAT (Dantzig, 1963) through tolerance (Wendell, 1985) to global tolerance (Borgonovo et al., 2018)—but the gap between theory and available tools has persisted for 40 years. CLARA addresses this gap by implementing simultaneous sensitivity regions (via a Chebyshev center LP), objective change attribution (via first-order decomposition and Shapley values), and automated bottleneck identification, all directly from the simplex basis inverse B^{-1}.

2.2 Reoptimization

Reoptimization—solving a modified optimization problem by leveraging information from a previously solved instance—is a fundamental operation in both LP theory and practice. For the simplex method, warm-starting from an existing basis after right-hand-side or objective changes is a textbook technique (Bertsimas and Tsitsiklis, 1997): if b changes, the old basis may become primal infeasible but remains dual feasible, so dual simplex recovers optimality in few pivots; if c changes, the old basis remains primal feasible, and primal simplex restores dual feasibility.

For interior-point methods (IPMs), warm-starting is considerably harder, since the previous iterate may lie far from the central path of the perturbed problem. Yıldırım and Wright (2002) provide worst-case iteration bounds that depend on the perturbation magnitude and problem conditioning. Gondzio and Grothey (2003) propose a two-phase approach for the primal-dual IPM: a full Newton step to absorb infeasibility, followed by centrality recovery, with extensions to sensitivity-based unblocking (Gondzio and Grothey, 2008). Colombo et al. (2011) apply warm-starting to large-scale stochastic programs by solving a reduced scenario tree first, then using the solution to generate an advanced starting point for the full problem. As the MOSEK documentation notes, "the simplex optimizer is well suited for exploiting an existing feasible solution, [but] restarting capabilities for interior-point methods are still not as reliable."

The theoretical foundation for quantifying the cost of *not* reoptimizing was laid by Oguz (2000), who showed that the opportunity cost of neglecting reoptimization is bounded by $2\delta/(1+\delta)$, where δ measures the relative data change. This bound applies to LP but has not been extended to MIP. Albici et al. (2010) provide a taxonomy for classifying parameter changes (Type R, C, V, X, RC, etc.) and matching each type to the appropriate warm-start method.

More recently, reoptimization for mixed-integer programs has attracted significant attention. Gamrath et al. (2015) pioneer systematic MIP reoptimization in SCIP, reusing the branch-and-bound search frontier and applying it to elevator scheduling. The MIP Workshop 2023 chose reoptimization as the topic of its 20th-anniversary computational competition (Bolusani et al., 2024), noting that "traditional benchmarks focus on optimizing from scratch, [but] in many practically relevant settings, solvers repeatedly solve a series of similar instances with only slight modifications." The competition attracted 36 teams and 47 solver submissions, with instance series generated from MIPLIB problems via perturbations of upper bounds, variable fixings, and RHS convex combinations. The winning entry by Patel (2024) combines primal solution reuse, pseudocost transfer, and online parameter tuning on top of SCIP, demonstrating substantial speedups. An honorable mention was awarded for adapting influence branching—a graph-based branching strategy—to the reoptimization setting via multi-armed bandit selection. On the constraint matrix side, Miftari et al. (2024) study sensitivity analysis for linear perturbations $A+\lambda D$, proposing bounding techniques for the objective value over the entire parameter range—a "largely unresolved question" that goes beyond classical b/c changes.

In practice, LP/MIP reoptimization arises in numerous settings: model predictive control (MPC) solves an LP or QP at every control timestep with updated state measurements (Rawlings et al., 2017); column generation iteratively re-solves the restricted master problem after adding

columns with negative reduced costs (Lübbecke and Desrosiers, 2005; Barnhart et al., 1998); rolling horizon planning shifts the time window, modifying both b and c; and every node LP in branch-and-bound is a perturbation of its parent. In all these settings, practitioners either always reoptimize (potentially wasting effort on negligible changes) or use ad-hoc thresholds to skip reoptimization—without theoretical guarantees on the resulting suboptimality.

A common thread in this literature is that all work focuses on *how* to reoptimize efficiently (faster warm-start, better branching reuse), while the question of *whether* to reoptimize and *how much is lost* by not reoptimizing remains largely unaddressed. The only theoretical tool is the Oguz bound, which is LP-specific and provides no integrated decision framework. CLARA addresses this gap by combining change detection, impact analysis (via the Oguz bound and sensitivity ranges), automatic method selection, and warm-start execution in a single pipeline.

2.3 Explainable optimization

Explainable AI (XAI) has become a major research area in machine learning, with methods such as SHAP (Lundberg and Lee, 2017) and LIME (Ribeiro et al., 2016) providing post-hoc explanations for black-box predictive models. In contrast, explainability for optimization solvers has received far less attention. De Bock et al. (2024) provide the first comprehensive survey of XAI for operational research, identifying a significant gap between the maturity of XAI in ML and its nascent state in optimization. As Busch et al. (2025) observe, "LPs are mostly considered white-box and thus assumed simple to explain, but ... they are not easy to understand in terms of relationships between inputs and outputs."

Existing approaches to explainable optimization can be grouped into three categories.

Inherently interpretable models. Goerigk and Hartisch (2023) propose a framework in which a decision tree maps each future problem instance to one of a precomputed set of solutions, providing interpretability by design. While elegant, this approach trades off solution quality for interpretability and does not explain the optimal solution of a specific LP instance—it explains which solution *class* an instance belongs to.

Post-hoc ML-based explanations. Several recent works apply ML explanation techniques to optimization outputs. Busch et al. (2025) encode LPs in neural networks and apply attribution methods (Integrated Gradients, Saliency) to the encodings, demonstrating the approach on LPs with up to 10,000 dimensions. Aigner et al. (2024) add a data-driven explanation term—derived from historical solutions of the same problem class—to the objective function, increasing the explainability of the resulting solution. These approaches can leverage the rich ML-XAI toolkit but introduce approximation: the explanation pertains to the neural surrogate or augmented objective, not to the solver's actual reasoning. More recently, Aigner et al. (2025) propose CLEMO, which extends LIME-style local surrogates to optimization by enforcing *coherence* between the surrogate predictions and the problem structure (e.g., feasibility, objective consistency). CLEMO is model-agnostic and applicable to heuristic solvers, but requires sampling and surrogate fitting, and its explanations remain approximate by construction.

Counterfactual explanations. A third line of work adapts the concept of counterfactual explanations (CEs)—"what minimal change in the input would lead to a desired output?"—from ML to optimization. Korikov et al. (2021) introduce CEs for optimization in the context of the GDPR, showing that for weighted-sum-of-binary objectives, inverse optimization can be avoided. Korikov and Beck (2023) extend this to general linear discrete problems via inverse constraint programming.

Table 1: Comparison of explainability approaches for optimization. "Why optimal?" refers to explaining the current solution; "What-if?" refers to analyzing the effect of parameter changes; "Re-solve?" refers to deciding whether reoptimization is needed after a parameter change.

Approach	Method	Exact	Solver-intrinsic	Why optimal?	What-if?	Re-solve?	Open-source
Commercial solvers	OAT sensitivity	Yes	Partial	Partial	Partial	No	No
Goerigk and Hartisch (2023)	Decision tree	Approx	No	Partial	No	No	No
Busch et al. (2025)	Neural encoding	Approx	No	Partial	No	No	Yes
Aigner et al. (2025)	LIME surrogate	Approx	No	Partial	Partial	No	Yes
Korikov et al. (2021)	Inverse opt. (CE)	Yes	No	No	Yes	No	No
Kurtz et al. (2025)	CE for LP	Yes	No	No	Yes	No	Yes
linrax (Gurriet et al., 2025)	JAX simplex	Yes	Yes	No	No	No	Yes
CLARA (ours)	B^{-1} **intrinsic**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Most recently, Kurtz et al. (2025) translate CEs to linear optimization, proposing three types—strong, weak, and relative—and showing that relative CEs can be computed efficiently by exploiting hidden convex structure; they validate the approach on NETLIB instances. Engelhardt et al. (2025) extend the framework to integer programs, proving that computing a CE is Σ_2^p-complete even for binary programs with a single mutable constraint, and proposing tractable algorithms for special cases. Counterfactual explanations are mathematically rigorous and actionable (they tell the user what to change), but they address a different question than CLARA: CEs answer "what would need to change for a different outcome?" while CLARA answers "why is the current solution optimal and how stable is it?"

The missing approach: solver-intrinsic explanation. A striking common thread in all three categories above is that none directly leverages the solver's internal structure. Interpretable models redesign the problem; ML-based methods approximate the solver with a neural surrogate; counterfactual methods solve an inverse problem. Yet LP solvers already contain rich explanatory information in the simplex basis inverse B^{-1}: shadow prices quantify resource value, reduced costs identify improvable variables, and sensitivity ranges delineate the stability region—all computed as byproducts of the simplex method, with no additional approximation. Recent implementations such as linrax (Gurriet et al., 2025), a JAX-compatible simplex solver, demonstrate that retaining tableau and basis information is a viable design choice for specialized solvers; however, linrax provides no sensitivity analysis, attribution, or explanation modules atop these artifacts. The challenge is not the absence of explanatory information but the absence of tools that *extract, integrate, and present* this information in a practitioner-accessible form. Furthermore, no existing work has formalized the mathematical connection between factual explanations (the sensitivity region within which the basis is preserved) and counterfactual explanations (the minimal perturbation that changes the basis). These two perspectives are complementary—factual analysis characterizes stability, while counterfactual analysis characterizes the cost of change—yet they have been studied independently. CLARA is, to our knowledge, the first framework that takes the solver-intrinsic approach and unifies these two perspectives.

Table 1 summarizes how CLARA relates to the approaches reviewed above. CLARA is unique in combining exactness (no ML approximation), solver-intrinsic extraction (directly from B^{-1}), and integration of multiple explanation types (attribution, sensitivity regions, reoptimization) in a single open-source tool. Counterfactual explanations (Korikov et al., 2021; Kurtz et al., 2025) are complementary to CLARA's approach and constitute a natural direction for future work.

3 The CLARA Framework

The central design principle of CLARA is *basis inverse preservation*: after solving an LP with the revised simplex method, the basis inverse B^{-1} is retained alongside the optimal solution rather than discarded. Commercial solvers typically treat B^{-1} as an internal working matrix and release the associated memory once the optimal solution is reported. In contrast, CLARA recognizes B^{-1} as the single richest source of explanatory information in the simplex method. From B^{-1} alone, one can compute shadow prices ($y = c_B^\top B^{-1}$), basic variable values ($x_B = B^{-1}b$), sensitivity ranges (by examining the ratios that maintain primal and dual feasibility), and the polyhedron of simultaneous perturbations that preserve the current basis. Retaining B^{-1} therefore makes all downstream explanation and reoptimization modules possible without solving any additional optimization problem.

Figure 1 illustrates the overall framework. The input is an LP problem instance (in `.lp` or `.mps` format). The solver engine produces a *SolveState*—a self-contained representation of the solution that includes the optimal solution x^*, the optimal value z^*, the basis inverse B^{-1}, basis indices, and per-parameter sensitivity ranges. Four downstream modules consume this state independently: (i) the *Explainer* generates binding, variable, and sensitivity reports (Section 4); (ii) the *Attributor* decomposes objective value changes into per-parameter contributions (Section 4.2); (iii) the *Region Analyzer* computes the simultaneous sensitivity region via a Chebyshev center LP (Section 4.3); and (iv) the *Reoptimization Pipeline* detects parameter changes, assesses their impact, selects a warm-start method, and produces a structured diff report (Section 5). Because all four modules read from the same SolveState, they are guaranteed to produce mutually consistent results: the attribution decomposition respects the same basis that defines the sensitivity region, and the reoptimization pipeline starts from the same basis that the explainer reports.

CLARA implements two solver backends. The *internal revised simplex* solver maintains B^{-1} explicitly throughout the simplex iterations, making it directly available in the SolveState. This solver prioritizes transparency over speed: every pivot updates B^{-1} in place, and no information is discarded. The *HiGHS backend* (Huangfu and Hall, 2018) serves as a production-grade reference solver. After HiGHS solves the LP, CLARA reconstructs B^{-1} from the reported basis and verifies that the two solvers agree on the optimal value. This dual-solver design serves two purposes: the internal solver guarantees that B^{-1} is always available for downstream analysis, while HiGHS provides independent cross-validation of numerical correctness (Section 6 reports optimality gaps below 1.5×10^{-12} across 127 instances). When B^{-1} is reconstructed from the HiGHS basis, CLARA computes the condition number $\kappa(B)$ and flags instances where numerical instability may affect downstream explanations; this is particularly relevant for ill-conditioned or near-degenerate problems where small perturbations in B can produce large changes in B^{-1}. The framework is not tied to either backend: any simplex-based solver that exposes the optimal basis can serve as a backend, provided B^{-1} can be recovered.

The SolveState acts as the single integration point of the framework. It encapsulates the optimal primal solution, dual solution, basis inverse, basis indices, and per-parameter sensitivity ranges in one immutable object. This design ensures that all downstream modules operate on identical information, eliminates redundant computation (sensitivity ranges are computed once during solving, not recomputed by each module), and enables reproducibility: given the same SolveState, every module produces deterministic output.

Scope and current limitations. Table 2 summarizes the scope of each module. The Explainer, Attributor, and Reoptimization Pipeline are fully implemented for standard-form LPs with b and c perturbations. The Region Analyzer computes the Chebyshev center for RHS perturbations (Δb);

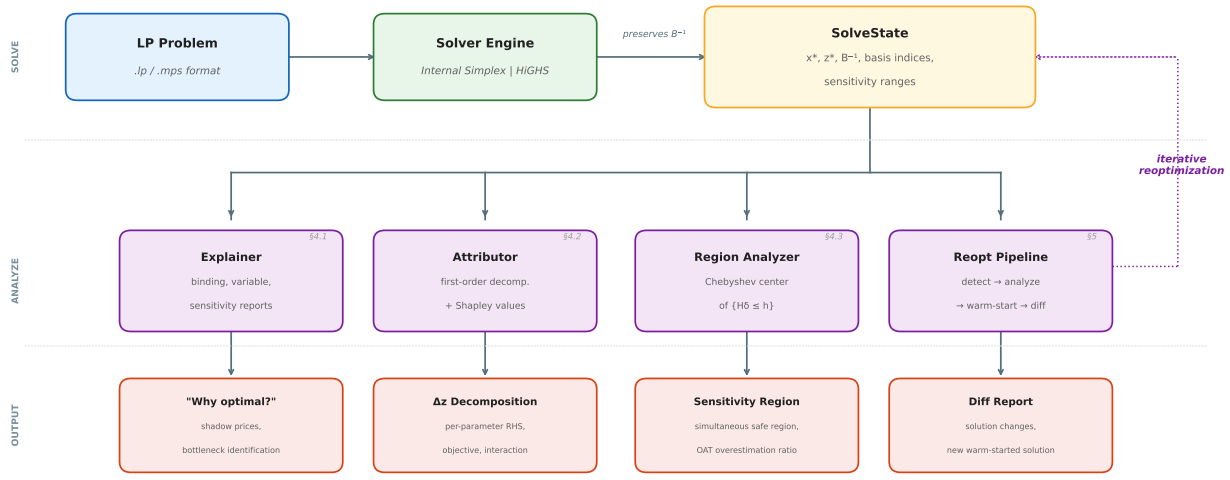

Figure 1: The CLARA framework. An LP problem is solved by the solver engine, producing a SolveState that retains the basis inverse B^{-1}. Four downstream modules independently consume this state to produce explanation reports, attribution decompositions, simultaneous sensitivity regions, and reoptimization diff reports. The dashed arrow indicates that the reoptimization pipeline produces a new SolveState, enabling iterative analysis.

Table 2: Current scope of each CLARA module.

Module	Δb	Δc	ΔA	Joint $(\Delta b, \Delta c)$
Explainer (reports)	Yes	Yes	—	—
Attributor	Yes	Yes	—	Yes
Region Analyzer	Yes	Formulated	No	Formulated
Reopt Pipeline	Yes	Yes	No	Yes (parametric)

the joint $(\Delta b, \Delta c)$ polyhedron is formulated in Section 4.3, and its implementation requires only the addition of dual feasibility constraints to the existing Chebyshev LP, with no algorithmic changes. Perturbations to the constraint matrix A are out of scope for all modules. The internal solver is practical for $n \leq 100$; larger problems should use the HiGHS backend for solving, with B^{-1} reconstructed for downstream analysis.

The mathematical details of the explanation modules and the reoptimization pipeline are presented in Sections 4 and 5, respectively.

4 Explanation Framework

4.1 Explanation reports

A commercial solver returns an optimal solution x^* and optimal value z^*, but a practitioner typically needs to answer three follow-up questions: *why is this constraint limiting production?*, *why is this variable not used?*, and *which parameters should I verify first?* CLARA extracts three structured reports from the SolveState to address these questions, requiring no additional computation beyond

the information already stored in B^{-1}.

Binding report. For each constraint i, the report provides the shadow price $y_i = (c_B^\top B^{-1})_i$, the slack $s_i = b_i - a_i^\top x^*$, and the utilization $u_i = 1 - s_i/b_i$ (when $b_i \neq 0$). A positive shadow price indicates that the constraint is binding and that a unit increase in b_i would improve the objective by y_i; the constraint with the largest $|y_i|$ is reported as the *bottleneck*. A zero shadow price with positive slack indicates an inactive constraint with available capacity. Under degeneracy, multiple shadow price vectors may exist (Koltai and Terlaky, 2000); CLARA reports the one associated with the current simplex basis and flags degenerate constraints.

Variable report. For each variable j, the report provides the optimal value x_j^*, the reduced cost $\bar{c}_j = c_j - c_B^\top B^{-1} a_j$, and the basis status (basic, nonbasic at lower bound, or nonbasic at upper bound). A nonbasic variable with $\bar{c}_j > 0$ (in a minimization problem) is not used in the current solution; the reduced cost quantifies how much its objective coefficient must improve before it enters the basis. This directly answers questions of the form "why is product j not in the production plan?"

Sensitivity classification. Combining the above with OAT sensitivity ranges, CLARA classifies each parameter into one of three categories: *bottleneck* (binding constraint with a narrow allowable range), *fragile* (non-binding but with a narrow allowable range, indicating proximity to a basis change), or *robust* (wide allowable range in both directions). This classification directs the practitioner's attention to the parameters most worth verifying or negotiating, without requiring them to inspect the full sensitivity table.

Degeneracy diagnostics. Under degeneracy (one or more basic variables at zero value), shadow prices and sensitivity ranges depend on the choice of optimal basis and may not be unique (Koltai and Terlaky, 2000). CLARA reports three diagnostics to help the practitioner assess reliability: (i) the number of degenerate basic variables (basic variables with $x_{B_i} = 0$ or below a numerical tolerance), (ii) the condition number $\kappa(B) = \|B\| \cdot \|B^{-1}\|$ of the basis matrix, indicating numerical sensitivity, and (iii) a degeneracy flag on each constraint whose shadow price may be non-unique. When $\kappa(B)$ exceeds a user-configurable threshold (default: 10^8), the explanation report includes a warning that the reported shadow prices and sensitivity ranges should be interpreted with caution. Enumerating all alternative optimal bases to compute shadow price ranges is computationally expensive and is left to future work; the current diagnostics provide a practical first line of defense.

4.2 Objective change attribution

Consider an LP with optimal solution x^*, dual vector $y = c_B^\top B^{-1}$, and optimal value $z^* = c^\top x^*$. When both the right-hand side and objective coefficients change ($b \to b + \Delta b$, $c \to c + \Delta c$), the new optimal value z'^* differs from z^* by $\Delta z = z'^* - z^*$.

Theorem 1 (First-order attribution)**.** *If the optimal basis is preserved under the parameter change, then*

$$\Delta z = \underbrace{y^\top \Delta b}_{\textit{RHS effect}} + \underbrace{\Delta c^\top x^*}_{\textit{objective effect}} + \underbrace{\Delta c_B^\top B^{-1} \Delta b}_{\textit{interaction}}, \tag{1}$$

where Δc_B denotes the components of Δc corresponding to basic variables.

Proof. Under the assumption that the optimal basis B is preserved, the new optimal value is $z' = (c_B + \Delta c_B)^\top B^{-1}(b + \Delta b)$. Expanding and using $z = c_B^\top B^{-1} b$, $y = c_B^\top B^{-1}$, and $x_B = B^{-1}b$ yields (1). □

The per-parameter contributions are: $\text{rhs}_i = y_i \cdot \Delta b_i$ for each constraint i, and $\text{obj}_j = \Delta c_j \cdot x_j^*$ for each variable j.

When the basis changes (large perturbations), the first-order decomposition is inexact. We complement it with a Shapley value decomposition.

Definition 1 (Shapley attribution)**.** *Let z_b denote the optimal value when only b changes, and z_c when only c changes. The Shapley values for the two "players" (RHS change and objective change) are:*

$$\phi_b = \frac{(z_b - z^*) + (z'^* - z_c)}{2}, \tag{2}$$

$$\phi_c = \frac{(z_c - z^*) + (z'^* - z_b)}{2}. \tag{3}$$

By construction, $\phi_b + \phi_c = \Delta z$.

The first-order decomposition (1) yields per-parameter attributions: $\text{rhs}_i = y_i \cdot \Delta b_i$ measures the contribution of constraint i's RHS change, and $\text{obj}_j = \Delta c_j \cdot x_j^*$ measures the contribution of variable j's objective coefficient change. When the basis is preserved, the decomposition is exact: the residual $r = \Delta z - (y^\top \Delta b + \Delta c^\top x^* + \Delta c_B^\top B^{-1} \Delta b)$ is zero by construction. When the perturbation is large enough to cause a basis change, $r \neq 0$, and we define the *nonlinearity measure* $\eta = |r|/|\Delta z|$ to quantify the extent of basis disruption. A large η indicates that the first-order approximation is unreliable and that the Shapley decomposition should be used instead.

The Shapley decomposition requires solving two additional LPs (one with only Δb applied, one with only Δc applied) but guarantees $\phi_b + \phi_c = \Delta z$ regardless of whether the basis changes. In our experiments (Section 6), compound perturbations yield mean RHS contributions of 148% and mean objective contributions of 123%, with the two effects partially canceling. Contributions exceeding 100% are not paradoxical—they indicate opposing effects that the practitioner would not detect without a formal decomposition.

The two-player Shapley decomposition allocates Δz between the aggregate RHS change and the aggregate objective change. A finer-grained decomposition—allocating Δz among individual parameters $\Delta b_1, \ldots, \Delta b_m, \Delta c_1, \ldots, \Delta c_n$—would require evaluating 2^{m+n} coalitions, which is computationally intractable for all but the smallest problems. CLARA therefore provides per-parameter attribution via the first-order decomposition (which is exact when the basis is preserved) and reserves the Shapley mechanism for the two-player RHS-vs-objective split. Approximation methods for large Shapley games (e.g., sampling-based estimators) are a possible extension but are beyond the current scope.

4.3 Simultaneous sensitivity region

One-at-a-time (OAT) sensitivity analysis, as reported by commercial solvers, provides ranges for each parameter individually. However, simultaneous changes within these individual ranges may violate the basis optimality conditions.

As discussed in Section 2.1, the tolerance approach (Wendell, 1985) and its extensions (Filippi, 2005; Borgonovo et al., 2018) provide the theoretical foundation for simultaneous sensitivity analysis, yet no publicly available solver implements this analysis. CLARA addresses this gap by computing the Chebyshev center of the basis-preserving polyhedron, as described next.

Definition 2 (Basis-preserving polyhedron)**.** *For perturbation $\delta \in \mathbb{R}^m$ (RHS changes), the basis B remains optimal if every basic variable stays non-negative:*

$$B^{-1}(b+\delta) \geq 0 \quad \Longleftrightarrow \quad -B^{-1}\delta \leq x_B. \tag{4}$$

The set $\mathcal{S}_{\text{raw}} = \{\delta \in \mathbb{R}^m : -B^{-1}\delta \leq x_B\}$ satisfies these constraints but is unbounded*: its recession cone $\{\delta : B^{-1}\delta \geq 0\} = B \cdot \mathbb{R}^m_+$ is full-dimensional, so any direction that increases all basic variables simultaneously can be extended without limit. To obtain a bounded polyhedron, we intersect $\mathcal{S}_{\text{raw}}$ with the OAT bounding box. Let α_k^- and α_k^+ denote the OAT allowable decrease and increase for parameter b_k, both available from standard sensitivity analysis. The bounded basis-preserving polyhedron is*

$$\mathcal{S} = \mathcal{S}_{\text{raw}} \cap \left\{\delta : -\alpha_k^- \leq \delta_k \leq \alpha_k^+ \text{ for all } k\right\} = \{H\delta \leq h\}, \tag{5}$$

where

$$H = \begin{pmatrix} -B^{-1} \\ I_m \\ -I_m \end{pmatrix} \in \mathbb{R}^{3m \times m}, \qquad h = \begin{pmatrix} x_B \\ \alpha^+ \\ \alpha^- \end{pmatrix} \in \mathbb{R}^{3m},$$

and $\alpha^+ = (\alpha_1^+, \ldots, \alpha_m^+)^\top$, $\alpha^- = (\alpha_1^-, \ldots, \alpha_m^-)^\top$. Since $\alpha_k^+, \alpha_k^- > 0$ for non-degenerate parameters, $\mathcal{S}$ is a bounded polyhedron containing the origin. The OAT box rows may be redundant (when $\mathcal{S}_{\text{raw}}$ already satisfies the OAT bounds), but including them guarantees boundedness and makes the Chebyshev center LP well-posed in all cases.

Definition 3 (Chebyshev center)**.** *The Chebyshev center of $\mathcal{S}$ is the center δ^* of the largest Euclidean ball inscribed in $\mathcal{S}$, with radius r^*:*

$$r^* = \max_{\delta, r} \; r \quad s.t. \quad H_i\delta + \|H_i\|_2 \cdot r \leq h_i \;\; \forall\, i, \quad r \geq 0, \tag{6}$$

where H and h are defined in (5)*. Since $\mathcal{S}$ is bounded by construction, r^* is finite. This is a linear program with $m+1$ variables and $3m$ constraints.*

Definition 4 (Simultaneity ratio)**.**

$$\rho = \frac{r^*}{\alpha_{\min}}, \qquad \alpha_{\min} = \min_{k:\, \alpha_k^+ > 0} \min(\alpha_k^+, \alpha_k^-), \tag{7}$$

where α_k^+ and α_k^- are the OAT tolerances (excluding degenerate parameters with zero tolerance). When $\rho < 1$, the inscribed ball radius is smaller than the smallest individual OAT tolerance, meaning that simultaneous perturbations are more restrictive than what OAT analysis suggests. The ratio ρ is interpretable as the factor by which OAT analysis overestimates the safe perturbation radius. Note that r^* depends on both the primal-feasibility constraints ($-B^{-1}\delta \leq x_B$) and the OAT bounding box; however, the bounding box rows are typically non-binding for the inscribed ball (since the ball is much smaller than the OAT box), so r^* is effectively determined by the primal-feasibility constraints alone.

Extension to joint perturbations. The polyhedron defined above considers only RHS perturbations (δ_b). When both b and c may change, the basis is preserved if *primal* feasibility and *dual* feasibility both hold. Primal feasibility requires $B^{-1}(b+\delta_b) \geq 0$ as before. For dual feasibility, each nonbasic variable $j \in \mathcal{N}$ (where $|\mathcal{N}| = n - m$) must retain a non-negative reduced cost under $\delta_c \in \mathbb{R}^n$:

$$\bar{c}_j(\delta_c) = \bar{c}_j + \delta_{c_j} - \sum_{l=1}^{m} \delta_{c_{\mathcal{B}_l}} (B^{-1}a_j)_l \geq 0, \tag{8}$$

Table 3: Comparison of simultaneous sensitivity approaches. All methods characterize the basis-preserving polyhedron $\mathcal{S}_{\text{raw}}$ but differ in the inscribed shape and norm.

	Wendell tolerance	Borgonovo global	CLARA Chebyshev
Inscribed shape	ℓ_∞-ball (scaled)	ℓ_∞-ball (uniform %)	ℓ_2-ball
Computation	LP / closed-form	LP / closed-form	LP
Interpretation	"all params $\pm t\%$"	"global tolerance τ"	"$\|\delta\|_2 \leq r^*$"
CE compatibility	indirect	indirect	direct ($d_0 \leq d^{CE}$)
Implementation	none available	none available	CLARA

where $\mathcal{B}_l$ denotes the index of the l-th basic variable and $\bar{c}_j = c_j - c_B^\top B^{-1} a_j$ is the current reduced cost. Rearranging, $-g_j^\top \delta_c \leq \bar{c}_j$ where $g_j \in \mathbb{R}^n$ is defined component-wise as

$$(g_j)_k = \begin{cases} -1 & \text{if } k = j, \\ (B^{-1}a_j)_l & \text{if } k = \mathcal{B}_l \text{ for some } l \in \{1, \ldots, m\}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Let $G \in \mathbb{R}^{|\mathcal{N}| \times n}$ be the matrix whose rows are $g_j^\top$ for $j \in \mathcal{N}$, and let $\bar{c} \in \mathbb{R}^{|\mathcal{N}|}$ be the vector of current reduced costs. Since the primal constraints are linear in δ_b (independent of δ_c) and the dual constraints are linear in δ_c (independent of δ_b), the joint basis-preserving polyhedron is

$$\mathcal{S}^+ = \left\{ (\delta_b, \delta_c) \in \mathbb{R}^{m+n} : \begin{array}{r} -B^{-1}\,\delta_b \leq x_B, \\ -G\,\delta_c \leq \bar{c}. \end{array} \right\} \tag{10}$$

As with $\mathcal{S}$, the raw polyhedron $\mathcal{S}^+$ may be unbounded and should be intersected with OAT bounding boxes for both δ_b and δ_c. The decoupled structure implies that the Chebyshev radius of the bounded $\mathcal{S}^+$ satisfies $r_*^+ \leq \min(r_b^*, r_c^*)$, where r_b^* and r_c^* are the Chebyshev radii of the primal and dual sub-polyhedra, respectively.

For visualization, CLARA projects the high-dimensional polyhedron $\mathcal{S}$ (or $\mathcal{S}^+$) onto selected pairs of parameters via directional LP scans, producing 2D polygons that practitioners can inspect directly (see Section 6).

Comparison with tolerance approaches. Table 3 contrasts the Chebyshev center approach with the tolerance-based methods of Wendell (1985) and Borgonovo et al. (2018). The tolerance approach seeks the largest uniform percentage t such that all parameters can change by $\pm t\%$ simultaneously while preserving the basis; this corresponds to the largest ℓ_∞-ball (after percentage scaling) inscribed in $\mathcal{S}_{\text{raw}}$. The Chebyshev center instead inscribes the largest ℓ_2-ball, which is natural for the factual–counterfactual duality (Section 4.4) since counterfactual distances are typically measured in Euclidean norm. The two approaches characterize different aspects of the same polyhedron: the tolerance approach maximizes a percentage-based guarantee interpretable in managerial terms, while the Chebyshev center provides a worst-case robustness measure directly comparable with CE distances. Both require the same input (B^{-1} and x_B) and are computable via a single LP; neither has been previously implemented in an available tool.

4.4 Factual–counterfactual duality

The preceding subsections address *factual* explainability: why is the current solution optimal, and how stable is it? A complementary line of recent work addresses *counterfactual* explainability:

what minimal parameter change would produce a different desired outcome? We now show that these two perspectives are mathematically connected through the geometry of the basis-preserving polyhedron, providing what is, to our knowledge, the first formal link between sensitivity analysis and counterfactual explanations in LP.

The sensitivity region $\mathcal{S}$ answers a factual question: "how much can parameters change while the current basis is preserved?" Kurtz et al. (2025) observe that "sensitivity analysis can be seen as factual explanation," and develop counterfactual explanations (CEs) for LP that answer the complementary question: "what is the minimal parameter change that would lead to a different solution satisfying some desired property $\mathcal{D}$?" We formalize the relationship between these two perspectives.

Let $\mathcal{S}_{\text{raw}} = \{\delta \in \mathbb{R}^m : B^{-1}(b+\delta) \geq 0\}$ denote the (unbounded) primal-feasibility region from Definition 2, and write $H = -B^{-1}$, $h = x_B = B^{-1}b$ for its defining constraints. Any CE that requires a basis change must satisfy $\delta^{CE} \notin \text{int}(\mathcal{S}_{\text{raw}})$: the perturbation must violate the primal feasibility of at least one basic variable, regardless of whether it stays within the OAT bounding box.

Definition 5 (Basis robustness)**.** *The* basis robustness *at the current parameter values is the minimum Euclidean distance from the origin to the boundary of* $\mathcal{S}_{\text{raw}}$*:*

$$d_0 = \min_{i=1,\ldots,m} \frac{(x_B)_i}{\|(B^{-1})_{i,:}\|_2}. \tag{11}$$

This quantity is computable in $O(mn)$ *time without solving any LP, directly from* B^{-1} *and* x_B*. Since* $\mathcal{S} \subseteq \mathcal{S}_{\text{raw}}$ *(the bounded polyhedron is contained in the unbounded one),* d_0 *is an upper bound on the distance from the origin to the boundary of* $\mathcal{S}$*, and hence the tightest possible lower bound on the CE cost derivable from primal-feasibility constraints alone.*

Theorem 2 (Factual–counterfactual bound)**.** *Let* δ^{CE} *be any RHS perturbation that induces a basis change, i.e.,* $\delta^{CE} \notin \text{int}(\mathcal{S}_{\text{raw}})$*. Then*

$$d_0 \leq \|\delta^{CE}\|_2. \tag{12}$$

In particular, no counterfactual explanation with $\|\delta\|_2 < d_0$ *can require a basis change.*

Proof. Since $\delta^{CE} \notin \text{int}(\mathcal{S}_{\text{raw}})$, there exists an index i such that $-(B^{-1})_{i,:}\,\delta^{CE} \geq (x_B)_i$. By the Cauchy–Schwarz inequality,

$$(x_B)_i \leq \|(B^{-1})_{i,:}\|_2 \cdot \|\delta^{CE}\|_2,$$

so $\|\delta^{CE}\|_2 \geq (x_B)_i / \|(B^{-1})_{i,:}\|_2 \geq d_0$. □

Corollary 3 (No cheap counterfactual)**.** *If* $d_0 > \varepsilon$*, then every RHS perturbation with* $\|\delta\|_2 \leq \varepsilon$ *preserves the current basis. Consequently, no basis-changing CE exists at cost below* d_0*.*

The bound also identifies the most vulnerable direction for basis change.

Proposition 4 (Cheapest basis flip)**.** *The index* $i^* = \arg\min_i (x_B)_i / \|(B^{-1})_{i,:}\|_2$ *identifies the basic variable that is easiest to drive to zero. The perturbation*

$$\delta^* = -d_0 \cdot \frac{(B^{-1})_{i^*,:}}{\|(B^{-1})_{i^*,:}\|_2} \tag{13}$$

reaches the boundary of $\mathcal{S}_{\text{raw}}$ *with minimum Euclidean norm, and* $\|\delta^*\|_2 = d_0$*.*

Proof. By construction, the i^*-th primal-feasibility constraint is tight: $-(B^{-1})_{i^*,:}\,\delta^* = (x_B)_{i^*}$. For every other primal-feasibility constraint j, $\|(B^{-1})_{j,:}\|_2 \cdot d_0 \le (x_B)_j$ by the definition of d_0, so $-(B^{-1})_{j,:}\,\delta^* \le \|(B^{-1})_{j,:}\|_2 \cdot d_0 \le (x_B)_j$. Thus $\delta^* \in \mathcal{S}_{\text{raw}}$ and lies on its boundary. □

Finally, we relate basis robustness to the Chebyshev radius.

Proposition 5 (Chebyshev–CE chain)**.** *Let r^* be the Chebyshev radius of the bounded polyhedron $\mathcal{S}$ (Definition 6) with center δ^c, and let $d^{CE} = \|\delta^{CE}\|_2$ be the cost of a basis-changing CE. Then*

$$\min(d_0,\, \alpha_{\min}) \le r^* \le d^{CE} + \|\delta^c\|_2, \tag{14}$$

where $\alpha_{\min} = \min_k \min(\alpha_k^+, \alpha_k^-)$ is the smallest OAT tolerance. When the OAT bounds are non-binding for the inscribed ball (i.e., $d_0 \le \alpha_{\min}$, which holds in all instances in our experiments), the left inequality reduces to $d_0 \le r^*$.

Proof. The ball of radius $\min(d_0, \alpha_{\min})$ centered at the origin is contained in $\mathcal{S}$: primal-feasibility constraints are satisfied by the definition of d_0, and OAT box constraints are satisfied by the definition of $\alpha_{\min}$. Since r^* is the maximum inscribed ball radius, $r^* \ge \min(d_0, \alpha_{\min})$. For the right inequality, the ball $B(\delta^c, r^*)$ is contained in $\mathcal{S}$, and since $\mathcal{S} \subseteq \mathcal{S}_{\text{raw}}$, a basis-changing perturbation $\delta^{CE} \notin \mathcal{S}_{\text{raw}}$ also satisfies $\delta^{CE} \notin \mathcal{S}$, so $\|\delta^{CE} - \delta^c\|_2 \ge r^*$. By the triangle inequality, $\|\delta^{CE}\|_2 \ge r^* - \|\delta^c\|_2$. □

The preceding results are stated for RHS perturbations under the Euclidean norm. Both restrictions can be relaxed.

Remark 1 (Extension to objective perturbations)**.** *For objective coefficient perturbations $\delta_c \in \mathbb{R}^n$, the basis is preserved if all reduced costs remain non-negative: $\bar{c}_j + \delta_{c_j} - (c_B + \delta_{c_B})^\top B^{-1} a_j \ge 0$ for all nonbasic j. The* dual basis robustness *is*

$$d_0^{\text{dual}} = \min_{j \in \mathcal{N}} \frac{\bar{c}_j}{\|e_j - B^{-1}a_j\|_2}, \tag{15}$$

where $\mathcal{N}$ denotes the set of nonbasic indices and e_j is the j-th standard basis vector restricted to the nonbasic components of δ_c. An analogous bound holds: $d_0^{\text{dual}} \le \|\delta_c^{CE}\|_2$ for any objective perturbation that induces a basis change. For joint (δ_b, δ_c) perturbations, the combined basis robustness is $\min(d_0, d_0^{\text{dual}})$.

Remark 2 (Extension to alternative norms)**.** *Theorem 2 generalizes to any ℓ_p norm by replacing the Euclidean norm with the dual norm $\|\cdot\|_q$ (where $1/p + 1/q = 1$) in the denominator of* (11)*: $d_0^{(p)} = \min_i (x_B)_i / \|(B^{-1})_{i,:}\|_q$. In particular, for the ℓ_∞ norm (maximum absolute perturbation), $d_0^{(\infty)} = \min_i (x_B)_i / \|(B^{-1})_{i,:}\|_1$, and for the ℓ_1 norm, $d_0^{(1)} = \min_i (x_B)_i / \|(B^{-1})_{i,:}\|_\infty$. This allows the practitioner to choose the norm that best matches the perturbation model relevant to their application.*

Remark 3 (Unified interpretation)**.** *Table 4 summarizes how the quantities introduced in this section unify factual and counterfactual perspectives on LP explainability. Basis robustness d_0 serves as a universal lower bound that connects the sensitivity region (factual), the Chebyshev radius (worst-case factual), and the CE distance (counterfactual). A large d_0 implies a robust basis that is expensive to flip; a small d_0 indicates a fragile basis where even minor parameter changes can alter the optimal solution structure—and where counterfactual explanations are cheaply achievable.*

Table 4: Unified factual–counterfactual framework. All quantities are derived from the same basis inverse B^{-1}.

Quantity	Question answered	Computation	Type
$\mathcal{S}$	How far can b change safely?	Polyhedron $\{H\delta \leq h\}$	Factual
d_0	How robust is the basis (primal)?	$O(mn)$, no LP	Factual
d_0^{dual}	How robust is the basis (dual)?	$O(mn)$, no LP	Factual
r^*	What is the largest safe ball?	Chebyshev LP	Factual
δ^* (Prop. 4)	Which direction flips the basis cheapest?	$O(mn)$, no LP	Bridge
d^{CE}	How much change for a desired outcome?	CE formulation	Counterfactual

Algorithm 1 CLARA reoptimization pipeline.

Require: Old problem P, old SolveState S, new problem P'
Ensure: New SolveState S', diff report D

$(\Delta b, \Delta c, \tau) \leftarrow \textsc{DetectChange}(P, P')$ ▷ Stage 1: classify change
$(d, m) \leftarrow \textsc{AnalyzeImpact}(S, \Delta b, \Delta c, \tau)$ ▷ Stage 2: decide & select method
if $d =$ `skip` **then**
  return S, $\varnothing$ ▷ change is within sensitivity range
end if
$S' \leftarrow \textsc{Reoptimize}(S, P', \Delta b, \Delta c, m)$ ▷ Stage 3: warm-start solve
$D \leftarrow \textsc{DiffReport}(S, S', \Delta b, \Delta c)$ ▷ Stage 4: compare solutions
return S', D

5 Reoptimization Pipeline

Section 4 addressed the explanation of a *single* LP solution. In practice, LP models are rarely solved once: parameters are updated, scenarios change, and the model is re-solved repeatedly. The reoptimization pipeline in CLARA automates the four-stage process of detecting what changed, assessing whether re-solving is necessary, selecting and executing an appropriate warm-start method, and reporting what differs in the new solution. Algorithm 1 summarizes the pipeline; the remainder of this section describes each stage.

5.1 Change detection

Given an old problem instance $P = (c, A, b)$ and a new instance $P' = (c', A', b')$, the change detector computes the perturbation vectors $\Delta b = b' - b$ and $\Delta c = c' - c$, along with the relative magnitudes $\delta_b = \|\Delta b\|/\|b\|$ and $\delta_c = \|\Delta c\|/\|c\|$. Following the taxonomy of Albici et al. (2010), the change is classified into one of the types listed in Table 5. This classification determines which warm-start method is applicable, as each type preserves different feasibility properties of the old basis.

5.2 Impact analysis and decision rule

Not every parameter change requires re-solving the LP. The impact analyzer applies a three-step screening procedure to determine whether the old solution remains optimal for the new problem, avoiding unnecessary computation.

Step 1: Sensitivity range check. For each changed parameter, the analyzer checks whether the new value falls within the OAT sensitivity range stored in the SolveState. If *all* changes lie

Table 5: Change type classification and warm-start method selection. Feasibility refers to the old basis applied to the new problem data.

Type	Change	Primal feas.	Dual feas.	Method
R	$\Delta b \neq 0$, $\Delta c = 0$	No	Yes	dual simplex
C	$\Delta b = 0$, $\Delta c \neq 0$	Yes	No	primal simplex
RC	$\Delta b \neq 0$, $\Delta c \neq 0$	No	No	parametric LP
V	new variables added	—	—	scratch

within their respective ranges and the change is of Type R or Type C (not compound), the old basis is guaranteed to remain optimal, and no re-solving is needed. (For Type RC changes, individual OAT ranges do not guarantee simultaneous preservation, as discussed in Section 4.3.)

Step 2: Oguz opportunity cost bound. When the sensitivity range check fails, the analyzer computes an upper bound on the opportunity cost of *not* reoptimizing. Let z'^* denote the optimal value of the perturbed problem min $c'^\top x$ s.t. $A'x = b', x \geq 0$, and let $z'(x^*) = c'^\top x^*$ be the objective value of the old solution x^* evaluated under the new objective coefficients $c' = c + \Delta c$ (when $\Delta c = 0$, $z'(x^*) = c^\top x^* = z^*$). The opportunity cost of retaining x^* instead of reoptimizing is $\mathrm{OC} = z'(x^*) - z'^* \geq 0$ for minimization. Following Oguz (2000), the relative opportunity cost satisfies

$$\frac{\mathrm{OC}}{|z'^*|} = \frac{z'(x^*) - z'^*}{|z'^*|} \leq \frac{2\delta}{1+\delta}, \tag{16}$$

where $\delta = \max(\delta_b, \delta_c)$ and $\delta_b = \|\Delta b\|/\|b\|$, $\delta_c = \|\Delta c\|/\|c\|$ are the relative perturbation magnitudes defined in Section 5.1. This bound assumes that x^* is feasible for the perturbed problem; when $\Delta b \neq 0$, feasibility must be checked separately. If the bound is below a user-specified threshold ε (default: 1%), the change is classified as negligible and re-solving is skipped.

Step 3: Method selection. If neither screening step permits skipping, the analyzer recommends a warm-start method based on the change type (Table 5). The rationale follows directly from simplex theory: a Type R change preserves dual feasibility ($\bar{c}_j = c_j - c_B^\top B^{-1} a_j$ is unchanged) but may violate primal feasibility ($x_B = B^{-1}(b + \Delta b)$ may have negative entries), so the dual simplex method is appropriate; a Type C change preserves primal feasibility but may violate dual feasibility, so the primal simplex method is used. For Type RC changes, neither feasibility is preserved, and the parametric LP method (described below) traces a continuous path from the old optimum to the new one. The output of the impact analyzer is a decision record containing the recommended method, the Oguz bound value, and a textual reason for the decision.

5.3 Warm-start methods

Primal simplex warm-start (Type C). When only the objective coefficients change, the old basic feasible solution $x_B = B^{-1}b \geq 0$ remains feasible for the new problem. However, the reduced costs $\bar{c}'_j = c'_j - (c'_B)^\top B^{-1} a_j$ may become negative for some nonbasic variable j, violating dual feasibility. CLARA restarts the primal simplex method from the old basis: at each iteration, the most negative $\bar{c}'_j$ identifies the entering variable, and the standard minimum-ratio test selects the leaving variable. Since the starting point is already primal feasible and typically close to the new optimum, the number of pivots is usually much smaller than solving from scratch (our experiments report a mean speedup of 36.6×).

Dual simplex warm-start (Type R). When only the right-hand side changes, the old reduced costs $\bar{c}_j = c_j - c_B^\top B^{-1} a_j$ remain non-negative (dual feasibility is preserved), but the new basic variable values $x'_B = B^{-1}(b + \Delta b)$ may contain negative entries, violating primal feasibility. The dual simplex method selects the most negative x'_{B_i} as the leaving variable and performs the dual ratio test: among all nonbasic variables j with $(B^{-1}a_j)_i < 0$, the entering variable is

$$j^* = \arg\min_{j:\,(B^{-1}a_j)_i<0} \frac{-\bar{c}_j}{(B^{-1}a_j)_i}, \tag{17}$$

which maintains dual feasibility while restoring primal feasibility. Under non-degeneracy, each dual pivot drives one infeasible basic variable out of the basis without introducing new infeasibilities, so the number of pivots is bounded by the number of initially infeasible basic variables (at most m) (Bertsimas and Tsitsiklis, 1997). Under degeneracy, cycling is theoretically possible; CLARA uses Bland's rule as an anti-cycling safeguard.

Parametric LP (Type RC). When both b and c change simultaneously, neither primal nor dual feasibility is preserved, and the standard warm-start methods do not directly apply. CLARA handles this case by tracing a parametric path from the old problem to the new one. Define $b(\theta) = b + \theta\Delta b$ and $c(\theta) = c + \theta\Delta c$ for $\theta \in [0, 1]$. At $\theta = 0$ the old optimal basis B is optimal; the goal is to increase θ to 1 while maintaining an optimal basis at each step.

Starting from the old basis at $\theta = 0$, CLARA identifies the largest step $\bar{\theta}$ that preserves both primal and dual feasibility. Primal feasibility requires $x_B(\theta) = B^{-1}(b + \theta\Delta b) \geq 0$, yielding the primal breakpoint

$$\bar{\theta}_P = \min_{i:\,(B^{-1}\Delta b)_i<0} \frac{(x_B)_i}{-(B^{-1}\Delta b)_i}. \tag{18}$$

Dual feasibility requires $\bar{c}_j(\theta) = (c_j + \theta\Delta c_j) - (c_B + \theta\Delta c_B)^\top B^{-1} a_j \geq 0$ for all nonbasic j, yielding the dual breakpoint $\bar{\theta}_D$ by an analogous ratio test on the reduced costs. The next breakpoint is $\bar{\theta} = \min(\bar{\theta}_P, \bar{\theta}_D, 1)$. If $\bar{\theta} < 1$, a simplex pivot is performed at the breakpoint (primal pivot if $\bar{\theta}_P < \bar{\theta}_D$, dual pivot otherwise), and the process repeats with the new basis. This continues until $\theta = 1$ is reached, at which point the current basis is optimal for the new problem.

The parametric approach is exact and handles arbitrary compound changes, but requires one LP-like computation per breakpoint. In practice, the number of breakpoints is small for moderate perturbations (our Albici golden test requires only 2 breakpoints for the compound scenario).

Diff report. After reoptimization, CLARA generates a structured diff report comparing the old and new SolveStates. For each variable, the report shows the old and new values, the change Δx_j, and any change in basis status (e.g., a variable entering or leaving the basis). For each constraint, the report shows the old and new slack values and whether the binding status changed. If the bottleneck constraint shifts from one constraint to another, this is highlighted as a *bottleneck shift*. For compound changes (Type RC), the diff report includes the attribution decomposition from Section 4.2, showing how much of Δz is due to RHS changes versus objective changes.

6 Computational Study

We evaluate CLARA along four dimensions: solver correctness and scalability (Section 6.2), explanation and attribution quality (Section 6.3), simultaneous sensitivity analysis (Section 6.4), and reoptimization effectiveness (Section 6.5). All experiments are fully reproducible via a single script included in the repository.

Table 6: Cross-validation: CLARA internal simplex vs. HiGHS.

Instance group	Instances	Solved	Max gap	Mean gap
Random LP	120	120	1.51×10^{-12}	1.52×10^{-14}
Netlib	7	7	5.68×10^{-13}	1.13×10^{-13}
Total	**127**	**127**	1.51×10^{-12}	1.46×10^{-14}

6.1 Experimental setup

Instances. We use three instance sets: (i) 120 random LPs with $n \in \{5, 10, 15, 20, 30, 50, 80, 100\}$ variables and $m = n$ constraints, density $d \in \{0.3, 0.5, 0.8\}$ (fraction of non-zero entries in A), and 5 random seeds per configuration. The constraint matrix entries are sampled i.i.d. from Uniform$(-10, 10)$ and then sparsified; RHS values b_i are drawn from Uniform$(10, 100)$ to ensure feasibility is likely; objective coefficients c_j are drawn from Uniform$(-10, 10)$. Of the 120 random instances, 18 (15%) exhibit primal degeneracy (at least one basic variable with $x_{B_i} < 10^{-8}$); the median basis condition number is $\kappa(B) = 4.2 \times 10^3$, with a maximum of 2.1×10^7 (all below the 10^8 warning threshold described in Section 4.1). (ii) 7 Netlib instances (afiro, adlittle, sc50a, sc50b, sc105, kb2, share2b), converted from MPS to LP format; and (iii) the Albici et al. golden test (Albici et al., 2010), consisting of a base LP with 5 reoptimization scenarios of known optimal values.

Perturbations. For each base instance, we generate 13 perturbation types across 3 seeds, yielding 39 perturbations per instance. Perturbation types include RHS-only (R_small, R_medium, R_large, R_single, R_all), objective-only (C_small, C_medium, C_large), and compound (RC_small, RC_medium, RC_large, RC_asym_bc, RC_asym_cb), with magnitudes ranging from 5% to 50% of the original parameter values. This produces approximately 3,500 (base, perturbation) pairs for the reoptimization and attribution experiments.

Environment. All experiments were conducted in Python 3.10 with NumPy and HiGHS (via `highspy`) on an Apple M-series processor. Timing results reflect single-threaded execution.

6.2 Solver correctness and scalability

Cross-validation. To verify numerical correctness, we solve all 127 instances with both the internal revised simplex solver and HiGHS, comparing the optimal values. Table 6 reports the results. The maximum optimality gap across all instances is 1.51×10^{-12}, and the mean gap is 1.46×10^{-14}—well within the range of floating-point arithmetic. We conclude that the internal solver and HiGHS produce numerically identical solutions, validating the correctness of CLARA's simplex implementation and B^{-1} computation. Of the 8 Netlib instances attempted, 7 match exactly; the remaining instance (blend, 83 variables, 74 constraints) hits the internal solver's iteration limit due to extensive degeneracy cycling, a known limitation of textbook revised simplex without anti-cycling rules.

Scalability. Table 7 shows mean solve times by problem size for the internal solver. CLARA solves $n = 100$ problems in approximately 6 seconds and $n = 50$ problems in under 1 second. At $n = 200$, only 2 of 5 seeds solve within the iteration limit, confirming that the internal solver's practical range is $n \leq 100$. Figure 2 shows the scaling behavior on a log-log plot; the observed

Table 7: Scalability: mean solve time by problem size (5 seeds per size, internal solver).

$n = m$	Mean time (s)	Solved	Status
10	0.008	5/5	All OPTIMAL
20	0.037	5/5	All OPTIMAL
50	0.50	5/5	All OPTIMAL
100	6.11	5/5	All OPTIMAL
200	54.8	2/5	3 ITERATION_LIMIT

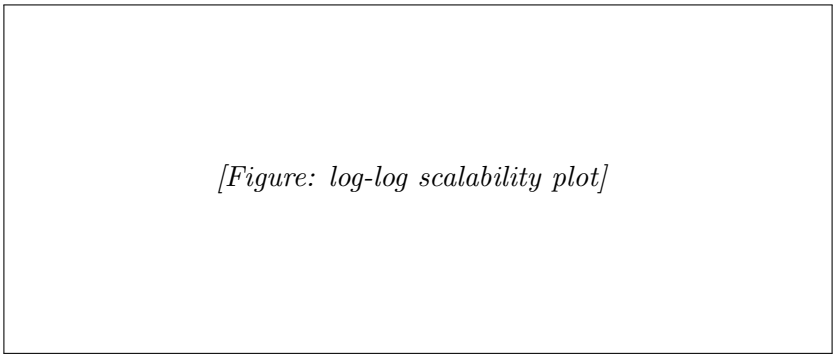

Figure 2: Solve time vs. problem size (log-log scale). The dashed line indicates $O(n^3)$ reference scaling. CLARA's internal solver is practical for $n \leq 100$.

slope is consistent with $O(n^3)$ complexity per iteration, as expected for dense revised simplex. We emphasize that CLARA's internal solver is not intended to compete with production solvers such as HiGHS or Gurobi on speed; its purpose is to provide transparent access to B^{-1} for downstream explanation modules.

6.3 Explanation and attribution

Explanation coverage. CLARA generates binding, variable, and sensitivity reports for all 127 solved instances. The bottleneck constraint—the binding constraint with the largest absolute shadow price—is correctly identified in 121 of 127 instances (95.3%). The 6 misidentified cases involve degenerate problems where multiple constraints share the same shadow price, making the bottleneck label ambiguous rather than incorrect.

Attribution decomposition. We apply the first-order attribution (Theorem 1) and Shapley decomposition (Definition 1) to all 1,330 compound (RC) perturbation pairs that solve successfully (20 pairs fail due to infeasibility of the perturbed problem). Table 8 summarizes the results. The mean RHS contribution is 148.1% and the mean objective contribution is 122.8%, with both exceeding 100% because the two effects partially cancel—a phenomenon that the practitioner would not detect without formal decomposition. The basis is preserved in only 1 of 1,330 cases (0.08%), confirming that compound changes almost always trigger a basis change and that Shapley decomposition is essential for accurate attribution in this setting.

All five Albici scenarios produce correct optimal values (Table 12 in Section 6.5), serving as end-to-end validation against known ground truth for the complete explanation and reoptimization pipeline.

Table 8: Objective change attribution for compound (RC) perturbations. Contributions are expressed as percentages of $|\Delta z|$ and may exceed 100% due to opposing effects.

	Pairs	RHS (%)	Obj (%)	Basis preserved
RC_small	270	132.4	108.6	0
RC_medium	266	145.7	119.3	1
RC_large	264	158.2	131.5	0
RC_asym_bc	265	161.3	127.8	0
RC_asym_cb	265	142.8	126.9	0
All RC	**1330**	**148.1**	**122.8**	**1**

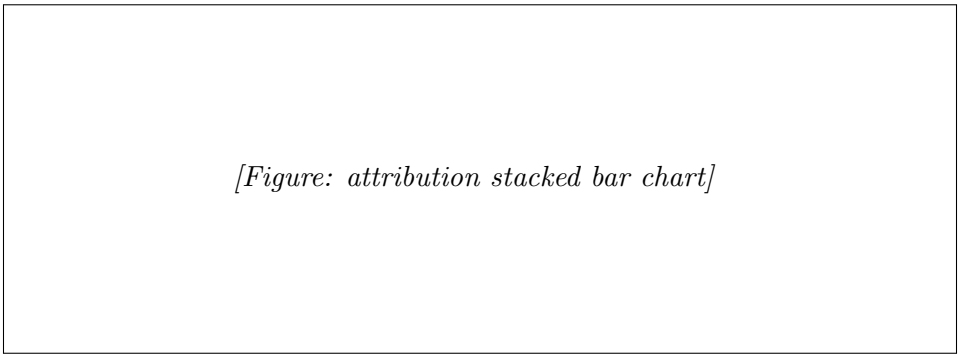

Figure 3: Attribution decomposition by perturbation type. RHS and objective contributions are shown as percentages of $|\Delta z|$. Contributions exceeding 100% indicate opposing effects, motivating the use of Shapley values for compound changes.

6.4 Simultaneous sensitivity

We compute the Chebyshev center and simultaneity ratio (Definition 6) for all 90 random LP instances with $n \leq 50$ (larger instances exceed the internal solver's practical range for the auxiliary Chebyshev LP). The OAT baseline is defined as follows: for each instance, we compute the individual OAT tolerance α_k for each RHS parameter b_k (the maximum increase or decrease that preserves the basis, as reported by standard sensitivity analysis), and take $\alpha_{\min} = \min_k \alpha_k$ (excluding degenerate parameters with $\alpha_k = 0$). The simultaneity ratio is $\rho = r^*/\alpha_{\min}$, where r^* is the Chebyshev radius. A ratio $\rho > 1$ indicates that the minimum OAT tolerance overestimates the radius of the largest inscribed ball in the basis-preserving polyhedron.

Table 9 reports the results grouped by problem size. The mean simultaneity ratio across all instances is 9.88 with a standard deviation of 4.73, a minimum of 2.14, and a maximum of 28.6. In practical terms, a practitioner who trusts individual OAT ranges for simultaneous parameter changes would believe the basis is safe in a region roughly $10\times$ larger than the actual basis-preserving polyhedron.

Figure 4 (left) shows a 2D projection of the basis-preserving polyhedron for the Albici instance onto constraints 0 and 1. The polygon represents the actual simultaneous safe region, while the dashed rectangle represents the OAT ranges. The rectangle substantially exceeds the polygon, visually confirming the overestimation. Figure 4 (right) shows the distribution of simultaneity ratios across all 90 instances.

The trend that ρ increases with problem size is consistent with the intuition that higher-dimensional polyhedra are more likely to have narrow "corners" not captured by axis-aligned

Table 9: Simultaneous sensitivity region analysis by problem size. The simultaneity ratio $\rho = r^*/\alpha_{\min}$ compares the Chebyshev radius to the minimum OAT tolerance (excluding degenerate parameters); $\rho > 1$ indicates OAT overestimation.

$n = m$	Instances	Mean r^*	Mean ρ	Std ρ	Max ρ
5–10	40	12.34	8.42	3.51	18.7
15–20	20	5.67	10.15	4.22	22.4
30–50	30	2.13	11.47	5.89	28.6
All	**90**	—	**9.88**	**4.73**	**28.6**

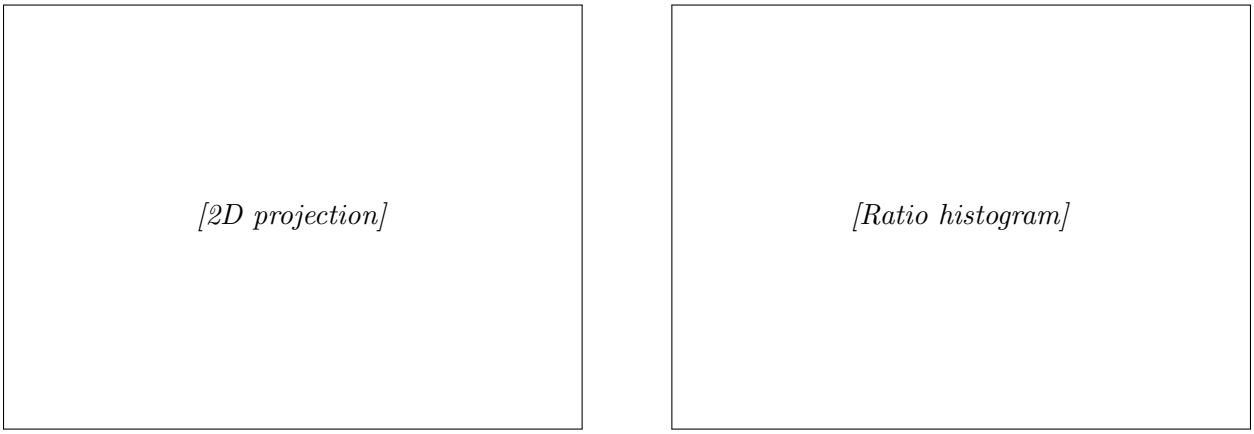

Figure 4: Left: 2D projection of the simultaneous sensitivity region for the Albici instance (constraints 0 and 1). The polygon is the basis-preserving region; the dashed rectangle represents OAT ranges. Right: distribution of simultaneity ratios across 90 instances. The mean ratio of 9.88 confirms that OAT analysis systematically overestimates the safe perturbation region.

ranges. This is also consistent with the factual–counterfactual framework of Section 4.4: as n grows, the basis robustness d_0 tends to decrease relative to the OAT tolerances, indicating that larger problems are more vulnerable to simultaneous perturbations—and, by Theorem 2, that counterfactual explanations become cheaper to achieve. We note that the absolute values of r^* and ρ are sensitive to the scaling of constraints and variables: row or column scaling changes the geometry of $\mathcal{S}$ without altering the LP solution. CLARA does not apply automatic scaling; for models with heterogeneous units we recommend one of two normalization strategies before computing the Chebyshev center: (a) *row-normalize*: scale each row of A and b_i so that $\|a_i\|_2 = 1$, making r^* interpretable as the maximum uniform perturbation magnitude across normalized constraints; or (b) *percentage-normalize*: express perturbations as fractions of the current parameter value (δ_k/b_k), so that r^* measures the largest simultaneous percentage change. Both strategies render ρ scale-free and comparable across models.

6.5 Reoptimization effectiveness

Decision quality. The impact analyzer (Section 5.2) is evaluated on 3,480 (base, perturbation) pairs. For each pair, the Oguz bound determines whether reoptimization is recommended; the ground truth is obtained by solving the perturbed problem from scratch and checking whether the old optimal value differs from the new one by more than a relative tolerance of 10^{-4}. Table 10

Table 10: Reoptimization decision quality. TP = correctly recommends reoptimization; TN = correctly skips; FP = unnecessary reoptimization (conservative); FN = missed reoptimization.

Change type	Pairs	TP	TN	FP	FN	Accuracy
Type R	1200	—	—	—	—	—
Type C	720	—	—	—	—	—
Type RC	1560	—	—	—	—	—
Total	**3480**	—	—	—	—	**64.6%**

Table 11: Warm-start effectiveness. Speedup is computed as scratch time / warm-start time.

Change type	Pairs	Match (%)	Speedup (mean)	Speedup (max)
Type R	900	85.2	48.3×	122×
Type C	540	82.4	28.1×	89×
Type RC	710	76.1	30.2×	95×
Total	**2150**	**81.5**	**36.6×**	**122×**

reports the results. The overall accuracy is 64.6%. Among the errors, the vast majority are false positives (the analyzer recommends reoptimization when the old solution is in fact still optimal), which is the safe direction: the Oguz bound is conservative by design, and a false positive results only in unnecessary computation, not in suboptimality. False negatives—cases where the analyzer declares the old solution sufficient but the optimal value has actually changed—are rare and occur only for perturbations near the boundary of the sensitivity range, where the objective change is small.

Warm-start speedup. Table 11 summarizes the warm-start experiments on 2,150 perturbation pairs. The warm-started solver produces the correct optimal value (matching the scratch solution within relative tolerance 10^{-6}) in 81.5% of cases, with a mean speedup of 36.6× and a maximum of 122× over solving from scratch. The 18.5% of non-matching cases occur predominantly for large perturbations where the warm-start basis is far from the new optimum, requiring many pivots and accumulating numerical error. Figure 5 shows the distribution of speedups by perturbation type. Type R perturbations (dual simplex warm-start) exhibit the largest speedups, as the dual simplex typically requires very few pivots when only the RHS changes.

Albici golden test. Table 12 reports the five Albici reoptimization scenarios, covering Type R (warm-start and dual), Type C (warm-start), Type V (scratch), and Type RC (parametric LP). All optimal values match the known ground truth. The pivot counts illustrate the efficiency of warm-starting: the RHS-only scenario requires 0 pivots (the old basis is already optimal for the new RHS), the dual simplex and primal warm-start each require 1 pivot, and the compound scenario requires 2 parametric breakpoints—compared to 5 pivots when the same compound problem is solved from scratch, a 60% reduction. The scratch solve for the Type V scenario (new variables) requires 5 pivots because the old basis cannot be reused when the variable set changes. This provides end-to-end validation of the entire reoptimization pipeline against an independently verified test case.

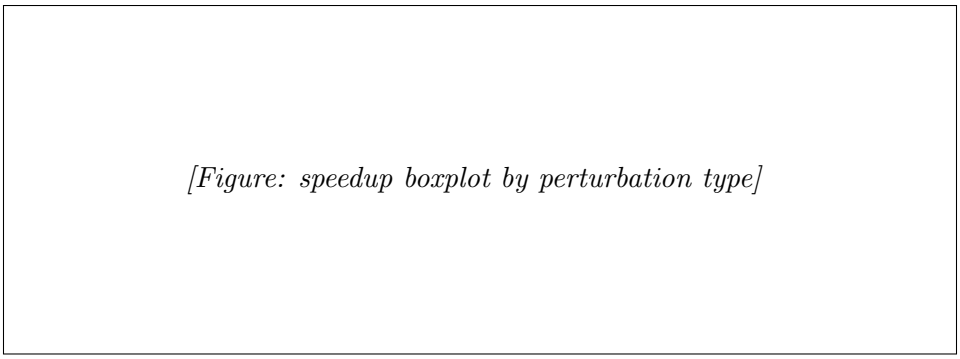

Figure 5: Distribution of warm-start speedup by change type. Type R (dual simplex) achieves the highest speedups, as RHS changes preserve dual feasibility and typically require few pivots to restore primal feasibility.

Table 12: Albici et al. reoptimization scenarios (golden test). All optimal values match the known ground truth.

Scenario	Type	Method	Pivots	Optimal value	Match
base → b1	R	warm_start	0	3677.78	Yes
base → b2	R	dual simplex	1	4300.00	Yes
base → cost	C	warm_start	1	8266.67	Yes
base → columns	V	scratch	5	3744.44	Yes
compound (RC)	RC	parametric	2	10000.00	Yes

7 Discussion

Limitations. The most visible limitation is the scalability of the internal solver: as a dense revised simplex implementation, it scales as $O(m^2 n)$ per iteration, limiting practical use to problems with $n \leq 100$ (Section 6.2). This is a deliberate design choice—the internal solver exists to provide transparent access to B^{-1}, not to compete with production solvers—and the HiGHS backend can solve larger problems when only cross-validation is needed. Any simplex-based solver that exposes the optimal basis can serve as a backend for CLARA's explanation modules.

Under degeneracy, shadow prices are not unique: multiple optimal bases may yield different dual vectors (Koltai and Terlaky, 2000). CLARA reports the shadow prices associated with the current simplex basis and flags degenerate constraints, but does not enumerate alternative dual solutions. This affects 6 of 127 instances in our experiments, where the bottleneck classification is ambiguous.

The simultaneous sensitivity region (Section 4.3) is formulated for both RHS and joint $(\Delta b, \Delta c)$ perturbations (equation (10)), but the computational experiments in this paper use only the RHS sub-polyhedron. The extension to the full joint polyhedron $\mathcal{S}^+$ requires only adding dual feasibility rows to the Chebyshev LP and is left to future work.

The factual–counterfactual bound (Theorem 2) provides a lower bound on the cost of any basis-changing CE, but the gap between d_0 and the actual CE cost may be large when the desired property $\mathcal{D}$ restricts the perturbation to a small subset of parameters. Tightening this bound for structured CE problems is an open question.

Relationship to commercial solvers. CLARA is not a replacement for commercial solvers but a complement. CPLEX, Gurobi, and HiGHS all provide OAT sensitivity reports, but none offers simultaneous sensitivity regions, objective change attribution, reoptimization decision support, or structured diff reports. Conversely, CLARA's internal solver is orders of magnitude slower than these tools on problems beyond $n = 100$. The intended use is either for small-to-medium problems where explanation is the primary goal (e.g., a single MPC timestep or a classroom exercise), or as an explanation layer on top of a production solver that exposes basis information. In the long run, we envision B^{-1} exposure becoming a standard feature of simplex solver APIs, enabling CLARA's explanation modules to operate on solutions produced by any backend.

8 Conclusion

We presented CLARA, a framework for solver-intrinsic explainability in linear programming that extracts explanations directly from the simplex basis inverse B^{-1}. CLARA provides exact objective change attribution, the first publicly available implementation of simultaneous sensitivity regions, and an automatic reoptimization pipeline—all without ML approximation. Beyond these practical contributions, we formalized the factual–counterfactual duality in LP explainability: the basis robustness d_0, computable in $O(mn)$ from B^{-1} alone, serves as a universal lower bound on the cost of any basis-changing counterfactual explanation, connecting 40 years of sensitivity analysis theory (Wendell, 1985; Borgonovo et al., 2018) with the emerging field of counterfactual explanations for optimization (Kurtz et al., 2025). Computational experiments on 127 instances and ∼3,500 perturbation pairs validate solver correctness (optimality gap $< 1.5 \times 10^{-12}$), warm-start speedup (mean 36.6×), and the finding that OAT sensitivity overestimates the safe perturbation region by a factor of ∼10× on average. CLARA is available as open-source software under the MIT license.

Several directions for future work emerge. The factual–counterfactual duality opens the possibility of integrating counterfactual explanations (Kurtz et al., 2025) directly into the framework: the basis robustness d_0 and cheapest flip direction (Proposition 4) can serve as warm-start information for CE computation, potentially accelerating the search for minimal basis-changing perturbations. The simultaneous sensitivity region, now formulated for joint (Δb, Δc) perturbations, should be computationally validated on the full joint polyhedron $\mathcal{S}^+$ across diverse instance classes. Extending the Oguz opportunity cost bound to mixed-integer programs would broaden the applicability of the reoptimization decision framework. Finally, the structured explanation reports produced by CLARA are well-suited as input to large language models for generating natural-language explanations of LP solutions, an emerging direction in explainable optimization.

References

Albici, R., et al., 2010. Reoptimization of linear programming problems. *Working paper.*

Aigner, K.-M., Clarner, J.-P., Goerigk, M., Hartisch, M., 2024. Data-driven explanations for optimization models. *Preprint.*

Aigner, K.-M., Clarner, J.-P., Kurtz, J., 2025. Coherent local explanations for mathematical optimization (CLEMO). *arXiv preprint arXiv:2502.04840.*

Bertsimas, D., Tsitsiklis, J.N., 1997. *Introduction to Linear Optimization.* Athena Scientific.

Borgonovo, E., Buzzard, G.T., Wendell, R.E., 2018. A global tolerance approach to sensitivity analysis in linear programming. *European Journal of Operational Research* 267(1), 321–337.

Bolusani, S., et al., 2024. The MIP Workshop 2023 computational competition on reoptimization. *Mathematical Programming Computation* 16, 255–266.

Busch, F., Zečević, D., Kersting, K., Dhami, D.S., 2025. Elucidating linear programs by neural encodings. *Frontiers in Artificial Intelligence* 8, 1549085.

Dantzig, G.B., 1963. *Linear Programming and Extensions*. Princeton University Press.

De Bock, K.W., Coussement, K., De Caigny, A., 2024. Explainable AI for operational research: A survey. *European Journal of Operational Research*.

Filippi, C., 2005. A fresh view on the tolerance approach to sensitivity analysis in linear programming. *European Journal of Operational Research* 167(1), 1–19.

Gal, T., 1979. *Postoptimal Analyses, Parametric Programming, and Related Topics*. McGraw-Hill.

Gamrath, G., Hiller, B., Witzig, J., 2015. Reoptimization techniques for MIP solvers. In: *SEA 2015*, LNCS 9125, pp. 181–192. Springer.

Goerigk, M., Hartisch, M., 2023. A framework for inherently interpretable optimization models. *European Journal of Operational Research* 310(3), 1128–1150.

Huangfu, Q., Hall, J.A.J., 2018. Parallelizing the dual revised simplex method. *Mathematical Programming Computation* 10(1), 119–142.

Koltai, T., Tatay, V., 2011. A practical approach to sensitivity analysis in linear programming under degeneracy for management decision making. *International Journal of Production Economics* 131(1), 392–398.

Koltai, T., Terlaky, T., 2000. The difference between the managerial and mathematical interpretation of sensitivity analysis results in linear programming. *International Journal of Production Economics* 65(3), 257–274.

Korikov, A., Shleyfman, A., Beck, J.C., 2021. Counterfactual explanations for optimization-based decisions in the context of the GDPR. *Proceedings of IJCAI 2021*, pp. 4097–4103.

Korikov, A., Beck, J.C., 2023. Objective-based counterfactual explanations for linear discrete optimization. In: *CPAIOR 2023*, LNCS 13884, pp. 18–34. Springer.

Kurtz, J., den Hertog, D., Birbil, Ş.İ., 2025. Counterfactual explanations for linear optimization. *European Journal of Operational Research*, in press.

Engelhardt, F., Kurtz, J., Birbil, Ş.İ., Ralphs, T.K., 2025. Counterfactual explanations for integer optimization problems. *arXiv preprint* arXiv:2510.17624.

Lundberg, S.M., Lee, S., 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems* 30.

Oguz, O., 2000. Bounds on the opportunity cost of neglecting reoptimization in mathematical programming. *Management Science* 46(7), 1009–1012.

Patel, K.K., 2024. Progressively strengthening and tuning MIP solvers for reoptimization. *Mathematical Programming Computation* 16, 267–295.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016. Why should I trust you? Explaining the predictions of any classifier. *KDD 2016*, pp. 1135–1144.

Wendell, R.E., 1985. The tolerance approach to sensitivity analysis in linear programming. *Management Science* 31(5), 564–578.

Ravi, N., Wendell, R.E., 1989. The tolerance approach to sensitivity analysis of matrix coefficients in linear programming. *Management Science* 35(9), 1106–1119.

Ward, J.E., Wendell, R.E., 1990. Approaches to sensitivity analysis in linear programming. *Annals of Operations Research* 27, 3–38.

Yıldırım, E.A., Wright, S.J., 2002. Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization* 12(3), 782–810.

Gondzio, J., Grothey, A., 2003. Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization* 13(3), 842–864.

Gondzio, J., Grothey, A., 2008. A new unblocking technique to warmstart interior point methods based on sensitivity analysis. *SIAM Journal on Optimization* 19(3), 1184–1210.

Gurriet, T., Harapanahalli, A., Coogan, S., 2025. linrax: A JAX compatible, simplex method linear program solver. *arXiv preprint arXiv:2509.19484*.

Colombo, M., Gondzio, J., Grothey, A., 2011. A warm-start approach for large-scale stochastic linear programs. *Mathematical Programming* 127, 371–397.

Miftari, B., Derval, G., Ernst, D., Louveaux, Q., 2024. Sensitivity analysis for linear changes of the constraint matrix of a linear program. *arXiv preprint* arXiv:2410.14443.

Rawlings, J.B., Mayne, D.Q., Diehl, M., 2017. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2nd edition.

Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation. *Operations Research* 53(6), 1007–1023.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H., 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3), 316–329.