GEAR: A GPU-Accelerated Global Solver for Bounded Continuous Nonlinear Programs via Linear Bound Propagation

Duo Zhou, Hesun Chen, Xiangru Zhong, Grani A. Hanasusanto, Huan Zhang
University of Illinois Urbana-Champaign
duozhou2@illinois.edu, huan@huan-zhang.com

Abstract

We present a novel GPU-accelerated global solver for constrained nonlinear programming (NLP) inspired by the efficient linear bound propagation framework in neural network (NN) verification. Existing work on NN verification, a problem that can be cast as feasibility detection for a nonlinear objective, demonstrated orders-of-magnitude speedups with the linear bound propagation-based framework when compared to traditional solvers. Extending this paradigm to global nonlinear programming requires addressing many challenges: a global constrained NLP solver must effectively exploit all, possibly unstructured, nonlinear constraints to improve the objective, whereas NN verification typically handles simple input box constraints over a well-structured NN and does not directly optimize an objective function. We address these challenges by fully exploiting the linear bounds produced by bound propagation: these linear bounds can be viewed as relaxations of nonlinear constraints and used to provide lower bounds, check infeasibility, and guide the search for a primal solution. In addition, we propose tighter linear relaxations for bilinear, quotient, and other commonly used nonlinear functions in NLP problems, and a projected augmented Lagrangian method tightly coupled to our dual solving procedure to produce primal solutions. Our method solves 421 of 505 bounded and continuous NLP problems in the GAMS and MINLPLib benchmarks under a 180-second limit, more than each of the six global solvers we compare with, including Gurobi and BARON (commercial), and SCIP (open-source). Sixteen of these problems are certified by none of the six global solvers; all of them contain exponential, logarithmic, or trigonometric terms, which are often challenging for traditional approaches. Our method demonstrates a distinct, highly complementary regime for GPU-accelerated global optimization of bounded continuous nonlinear programs. Project webpage is at `https://verified-intelligence.github.io/GEAR`.

1 Introduction

Neural-network (NN) verification asks whether all outputs of a computation graph satisfy a prescribed property over a bounded input set, and can be cast as feasibility detection for a nonlinear objective [11, 22, 47, 34, 17, 18]. Many modern NN verifiers, such as α,β-CROWN, solve this through *linear bound propagation*: linear coefficients are propagated backward through the graph to obtain valid bounds on the output, and the coefficient updates batch as dense tensor operations on GPUs. This pipeline has delivered orders-of-magnitude speedups over traditional verification approaches built on LP, SDP, or MILP solvers [70, 65, 62, 66, 69, 63, 57, 54, 16, 67, 64]. The same paradigm could in principle drive a general global solver for nonlinear programs (NLPs) with continuous, box-bounded

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

variables, where linear bound propagation through the computation graph would supply valid bounds for thousands of problems in a single batched workload [13, 12, 52, 60, 5].

Extending linear bound propagation from NN verification to global NLP is, however, far from automatic. A constrained NLP carries an objective together with many, possibly unstructured, nonlinear equality and inequality constraints over operators such as products, quotients, powers, logarithms, square roots, and trigonometric terms; a global solver must exploit *all* of these constraints simultaneously to drive the objective down, certify optimality, and recover feasible incumbents [10]. NN verification, by contrast, typically operates over a well-structured neural network and a simple input box, and never directly optimizes an objective. Classical global NLP solvers [71, 7, 58, 9, 40] implement spatial branch-and-bound on top of per-node relaxations, McCormick envelopes, optimization-based bound tightening, cut pools, and primal heuristics. These relaxations are mostly polyhedral, although some methods use convex nonlinear relaxations [1, 31] and, for nonconvex quadratic programs, convex quadratic or semidefinite relaxations [44, 14]. Each subdomain may demand its own relaxation solves and nonlinear bookkeeping, which makes thousands of subdomains and constraint outputs hard to evaluate as one GPU-batched workload.

We introduce GEAR, a GPU-accelerated global solver for bounded continuous NLPs that closes this gap by *fully exploiting* the linear bounds produced by bound propagation. The same valid linear coefficients are read simultaneously as: (i) linear relaxations of the original nonlinear constraints, which also clip the input and intermediate domains; (ii) a Lagrangian combination of these linear bounds that yields a constraint-aware lower bound on the objective; (iii) an infeasibility certificate when some nonnegative combination of the relaxations is violated everywhere on the current box; and (iv) a primal seed and a branching score that guide the spatial branch-and-bound search. The objective and all constraints are vectorized as outputs of one computation graph, so linear bound propagation returns coefficient tensors for every constraint across thousands of branch-and-bound subproblems, and the downstream lower bounding, infeasibility testing, primal initialization, and branching all execute as batched tensor operations rather than as a sequence of per-node LP solves. The key observation is that the linear bounds themselves relax every constraint, and their Lagrangian combination is an LP dual that GEAR evaluates in closed form per box. The price, a looser bound per box than an LP with all McCormick planes, is offset in the tree by bounding many boxes per GPU step, which helps on highly nonlinear problems of low and moderate dimension rather than on large structured models (Section 4).

Two further ingredients make this pipeline effective. First, since the tightness of linear bound propagation is determined largely by the per-operator linear relaxations, we develop tighter ones for the operators that dominate NLP but are rare in NN verification. In particular, for the bilinear products that pervade NLP benchmarks, we choose among the McCormick planes by their value on the current box, and we use secant-tangent bounds that follow each operator's curvature and direct bounds for quotients, reciprocals, powers, and trigonometric terms, all designed to keep bound propagation tensorized. Second, we propose a projected augmented Lagrangian method (ALM) for primal recovery that is tightly coupled to the dual procedure: the minimizer of the Lagrangian bound starts the primal search, its multipliers initialize the ALM multipliers, and first-order primal updates batch across thousands of nodes instead of running an SQP or interior-point subproblem sequence.

On 505 bounded continuous NLP instances drawn from the `gams_world` and `minlplib` benchmarks under a 180-second wall-clock limit, feasibility tolerance 10^{-6} and relative gap 10^{-4}, GEAR solves 421 instances, more than each of the six global solvers we compare with, including the commercial Gurobi (401) and BARON and the open-source SCIP. Against the strongest baseline, Gurobi, GEAR solves 38 instances that Gurobi does not (Gurobi solves 18 that GEAR does not), and 16 instances are certified by none of the six global solvers; all of them contain exponential, logarithmic or trigonometric terms, whose compositions are often out of reach for classical LP-relaxation-based pipelines. GEAR therefore complements CPU global solvers with a GPU-accelerated regime for global optimization of bounded continuous nonlinear programs; mixed-integer and unbounded programs are outside the scope of this paper. Our contributions:

- **Linear bound propagation as a unified interface for global NLP**: Inspired by the success of linear bound propagation in neural network verification, we demonstrate its success as a new GPU-accelerated regime for bounded continuous NLP. One set of linear bounds serves every component, and computing them requires no LP solve (Sections 3.2, 3.3 and 3.5).
- **Tighter operator-specific linear relaxations** for the bilinear, quotient, power and trigonometric operators of NLP (Section 3.3), essential to make linear bound propagation successful for NLPs.

- **A projected ALM primal solver tightly coupled to the dual solver** and guided by the linear bounds (Section 3.4).
- **An open-source complementary GPU-batched solver for bounded continuous global NLP**, evaluated on 505 GAMS and MINLPLib instances with promising results against six global solvers.

2 Preliminaries

Bounded Continuous Nonlinear Programs We consider bounded continuous nonlinear programs (NLPs)

$$p^\star = \min_{x \in B_0} f_0(x) \quad \text{s.t.} \quad f_i(x) \le 0, \quad i = 1, \dots, m, \tag{1}$$

where f_0 is the objective, $f_1, \dots, f_m$ are the constraint functions, and $B_0 \subset \mathbb{R}^n$ is a box with finite bounds. Equalities and two-sided inequalities of the source model are written as pairs of one-sided inequalities, and a feasibility tolerance $\tau_{\text{feas}} \ge 0$ is absorbed into every f_i; for example, $g(x) = 0$ becomes $g(x) - \tau_{\text{feas}} \le 0$ and $-g(x) - \tau_{\text{feas}} \le 0$. Lower bounds and feasibility checks therefore refer to the same problem (1). Each f_i is factorable over the operator library of Section 3.3; integer variables, unbounded variables, and unsupported operators are outside our scope.

The objective and all constraints can be written as the outputs of one computation graph,

$$F(x) = \big(f_0(x), f_1(x), \dots, f_m(x)\big), \tag{2}$$

in which every intermediate variable z is produced by one operator, $z = h(y)$, applied to the input x or to earlier intermediate variables y (one variable, or two for binary operators such as products). For a box $B = [\underline{x}, \overline{x}] \subseteq B_0$, the feasible set and optimal value of the subproblem on B are

$$\mathcal{F}(B) = \{x \in B : \ f_i(x) \le 0, \ i = 1, \dots, m\}, \qquad p^\star(B) = \min_{x \in \mathcal{F}(B)} f_0(x), \tag{3}$$

with $p^\star(B) = +\infty$ if $\mathcal{F}(B) = \emptyset$.

Neural Network Verification with Linear Bound Propagation Given a scalar output f of a computation graph and an input box $B = [\underline{x}, \overline{x}]$, neural network (NN) verification asks whether $\min_{x \in B} f(x) > 0$ [61, 60, 13]; in its basic form this is a lower-bounding problem over a box, and constraint-aware verifiers additionally incorporate linear constraints such as activation splits and cutting planes [62, 69, 73]. Linear bound propagation (LBP) [70, 65, 62, 66, 69] answers the basic problem as follows; we follow the notation of [62] and [73], with underlines and overlines for lower and upper bounds. For each nonlinear operator $z = h(y)$, let $Y = [\underline{y}, \overline{y}]$ contain every value of y attained for $x \in B$ (the *intermediate bounds*), and let $\underline{h}, \overline{h}$ be a linear relaxation of h on Y:

$$\underline{h}(y) \le h(y) \le \overline{h}(y) \qquad \forall y \in Y. \tag{4}$$

Starting from f, LBP maintains a linear expression in the intermediate variables that bounds f from below and substitutes these relaxations from the output toward the input: a term γz with $z = h(y)$ becomes $\gamma\, \underline{h}(y)$ if $\gamma \ge 0$ and $\gamma\, \overline{h}(y)$ otherwise, which preserves the inequality; linear operators are substituted exactly. When only the input remains, the result is a linear lower bound (an upper bound can be obtained symmetrically)

$$\underline{f}(x) = a^\top x + c \ \le \ f(x) \qquad \forall x \in B. \tag{5}$$

Intermediate bounds come from running LBP with each intermediate variable as the output, intersected with interval bounds. A linear function attains its minimum over a box at a vertex selected coordinate-wise by the signs of its coefficients a_j, so the scalar bound, called the *concretization* of (5), is closed-form:

$$\min_{x \in B} \underline{f}(x) = c + \sum_{j=1}^{n} \min\{a_j \underline{x}_j, \ a_j \overline{x}_j\}. \tag{6}$$

Each propagation step is a product of coefficient tensors, so propagation from many outputs over many boxes runs as one batched GPU computation. Besides producing this scalar, the coefficients (a, c) describe how the bound varies across B; GEAR keeps them and uses them in all of its components (Section 3.2). Complete verifiers wrap LBP in branch-and-bound that splits either the input box (*input BaB*) or activations (*activation BaB*) [62, 73].

Spatial Branch-and-Bound In its usual rectangular form, spatial branch-and-bound (sBaB) maintains a queue $\mathcal{Q}$ of boxes, initialized with B_0, and an incumbent value $\text{UB} = f_0(\bar{x})$ of the best feasible

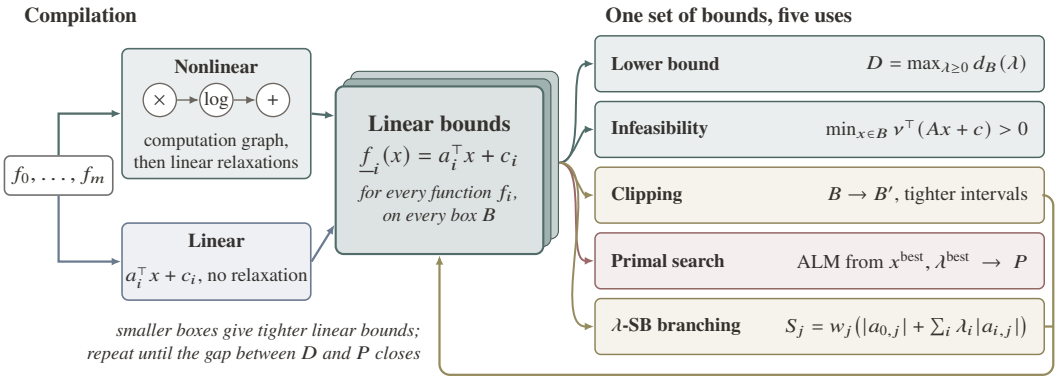

Figure 1: **Overview of GEAR.** Each of $f_0, \ldots, f_m$ is nonlinear or linear. Nonlinear functions are represented as computation graphs, and linear bound propagation applies linear relaxations to them; linear functions need no relaxation. Either way each function has one linear bound on each box B. These bounds give the lower bound D, the infeasibility test and clipping (Section 3.3), the start of the projected ALM search for the incumbent P (Section 3.4), and the branching score (Section 3.5). Smaller boxes give tighter linear bounds, and an instance is solved once the relative gap of Equation (13) is at most 10^{-4}.

point $\bar{x}$ found so far by a *primal* search ($\mathrm{UB} = +\infty$ initially). For each box B it computes a *dual* (lower) bound $\mathrm{LB}(B) \leq p^\star(B)$ and removes B if $\mathcal{F}(B) = \emptyset$ is certified (*infeasibility pruning*) or if $\mathrm{LB}(B) \geq \mathrm{UB}$ (*dominance pruning*). A box may first be replaced by a smaller box that still contains $\mathcal{F}(B)$ (*bound tightening*); a box that survives is split along one coordinate into two children that cover it. At all times $\min\{\mathrm{UB}, \min_{B \in \mathcal{Q}} \mathrm{LB}(B)\} \leq p^\star$; the search stops when this bound is within ε of UB, or when $\mathcal{Q}$ is empty, in which case UB is optimal (or the problem is infeasible if $\mathrm{UB} = +\infty$). Validity requires only that lower bounds, infeasibility certificates, and tightening steps be valid and that incumbents be checked for feasibility; the splitting rule and the primal search affect speed only.

Many LP-based global solvers [71, 7, 58] branch on variable bounds, so each node is a box, but they obtain $\mathrm{LB}(B)$ over a polyhedron: an LP relaxation that may contain auxiliary variables, the model's linear constraints, operator relaxations, and cutting planes. LBP obtains (6) without an LP solve and batches across boxes, but it minimizes over the box alone and by itself does not account for the constraints of (1). Section 3 shows how to obtain constraint-aware lower bounds and infeasibility certificates from LBP alone, and how to pair them with a primal search and a branching rule, keeping all computations batched.

3 GEAR: Global NLP Solving with GPU-Accelerated Linear Bounds

GEAR extends linear bound propagation from verification to constrained global optimization. The central challenge is to make nonlinear constraints useful throughout the search, not merely for checking feasibility, but also for guiding lower bounding and feasible solution finding, while retaining GPU-batched computation.

Figure 1 connects the parts of this section: the graph representation (Section 3.2), the lower bounds built from linear bounds (Section 3.3), the projected ALM primal search (Section 3.4) and the batched branch-and-bound (Section 3.5). Presolve and the remaining components of the full solver only add valid bounds or verified points (Figure 5, Appendix B).

3.1 Why Verification Alone Is Insufficient for Constrained Global NLP

Both global NLP and NN verification problems require valid bounds, but differ in what must be found and certified. A verification problem asks whether a property holds over an input set; it can finish upon finding a counterexample (SAT) or verifying its properties (UNSAT). However, constrained optimization must instead find a point satisfying every constraint (feasible solution) and verify that no feasible point improves its objective by more than the prescribed tolerance (optimal). Its pruning threshold is therefore not fixed: it is the objective value of an incumbent that the solver must produce and improve. Constraint-aware verifiers already incorporate linear inequalities from specifications, activation splits, and cutting planes [62, 69, 73]. The challenge is to extend this machinery into an objective-directed primal-dual search over general nonlinear constraints.

This gap becomes apparent on both sides of the primal-dual search. A search that minimizes f_0 over the box alone can return points that violate $f_i \leq 0$. A lower bound computed from f_0 alone minimizes over the whole box, including infeasible points, so it can stay below $p^\star$ however far we branch. For example, minimizing x on $[0, 1]$ subject to $x \geq 1/2$ has optimum $1/2$, but every box containing 0 retains an objective-only lower bound of 0. More branching cannot remove that box by dominance; a constraint must rule it out or strengthen its bound. Conversely, a strong lower bound cannot drive dominance pruning without a feasible incumbent. Global NLP therefore needs both constraint-aware certificates and a search that improves feasible objective values, not merely a counterexample finder.

Both remedies must stay batched: the solver has to reuse constraint information across many subproblems without building a separate optimization model per box, keep linear constraints exact, and relax products, quotients and powers tightly enough for bounding to be useful. This motivates the shared representation of Section 3.2 and the operator relaxations of Section 3.3.

3.2 A GPU-Compatible Representation of Constrained NLPs

We begin by organizing the objective and constraints into a common computation graph, allowing bounding and primal search to reuse the same operations across many boxes. GEAR compiles them as the outputs of F in (2), with the first output designated as the objective. Forward evaluation, automatic differentiation, and LBP operate on this same graph, and shared subexpressions can be reused when exposed by the parser. The principal parallel dimension is the batch of boxes: each graph operation processes many subproblems with identical structure but different bounds, and all outputs are bounded by the same linear bound propagation.

Linear constraints need no relaxation: LBP substitutes linear operators exactly, so the linear bound of $f_i(x) = a_i^\top x + c_i$ is the constraint itself, with the feasibility tolerance already incorporated as in Section 2. These coefficients are independent of the current box, and when the parser exposes them as an explicit matrix, the solver reuses that matrix across subproblems whenever it constructs its constraint relaxation, tightens bounds, or evaluates candidates.

Beyond a shared graph, the components need a common representation of the objective and constraints on each box, which the linear bounds of propagation provide. On each box B, linear bound propagation provides

$$\underline{f}_i(x) = a_i^\top x + c_i \leq f_i(x) \qquad \forall x \in \mathcal{F}(B), \quad i = 0, \ldots, m. \tag{7}$$

Ordinary LBP gives validity throughout B; validity on $\mathcal{F}(B)$ also accommodates the constraint-tightened propagation introduced below. Across K boxes, the coefficients have logical shapes $K \times (m+1) \times n$ and $K \times (m+1)$. The same linear bounds support objective bounding, infeasibility certificates, clipping, primal initialization, and coordinate scoring without rebuilding a separate optimization model for each operation. A linear bound that cannot reject a whole box may still exclude part of it, tighten an intermediate interval, or become useful in combination with other linear bounds (Appendix A.2).

3.3 Strengthening Dual Bounds using Linear Bounds

Constraint-aware objective bounds. Although the shared linear bounds retain constraint information, ordinary box concretization still ignores feasibility and may minimize the objective over infeasible points. We therefore incorporate the constraints directly into objective bounding through Lagrangian dualization. Every feasible point satisfies $\underline{f}_i(x) \leq f_i(x) \leq 0$ for $i \geq 1$. Stacking these linear bounds, together with any linear cuts, as $A = [a_1, \ldots, a_m]^\top$ and $c = (c_1, \ldots, c_m)^\top$ defines

$$\mathcal{P}(B) = \{x \in B : Ax + c \leq 0\} \supseteq \mathcal{F}(B), \qquad p_{\mathrm{LP}}(B) = \min_{x \in \mathcal{P}(B)} \underline{f}_0(x) \leq p^\star(B). \tag{8}$$

Instead of solving this LP separately at every box, GEAR dualizes $Ax + c \leq 0$ while retaining the box as the inner minimization domain. For $\lambda \geq 0$, write $a(\lambda) = a_0 + A^\top \lambda$; then

$$d_B(\lambda) = \min_{x \in B} \big[\underline{f}_0(x) + \lambda^\top (Ax + c)\big] = c_0 + c^\top \lambda + \sum_{j=1}^{n} \min\{a_j(\lambda)\underline{x}_j, a_j(\lambda)\overline{x}_j\}. \tag{9}$$

We call $d_B(\lambda)$ the *Lagrangian bound*. At feasible points the added terms are nonpositive, preserving a lower bound on f_0; at infeasible minimizers of the objective's linear bound they can be positive, allowing the bound to rise. In the example of Section 3.1, $\lambda = 1$ makes $x + \lambda(1/2 - x) = 1/2$ throughout the box, recovering the constrained optimum. More generally, every $\lambda \geq 0$ gives

$d_B(\lambda) \leq p_{\mathrm{LP}}(B)$, and LP strong duality gives $\max_{\lambda \geq 0} d_B(\lambda) = p_{\mathrm{LP}}(B)$ when the relaxation is feasible. Note that the multipliers λ are thus dual variables of this LP, not KKT multipliers of the NLP: optimizing them closes the gap to the LP over the current linear bounds, not the gap between that LP and the NLP. Each evaluation reuses the existing linear bounds and the concretization of Equation (6), costing $O(mn)$ per box without rerunning bound propagation. Since $Ax_\lambda + c$ is a supergradient at an inner minimizer x_λ, GEAR takes a fixed budget of projected Adam ascent steps and keeps the best evaluated bound, so an incomplete ascent affects tightness, not validity. Because a fixed budget of Adam steps can stop far from the LP optimum, GEAR also proposes multipliers with batched PDHG [3] on the GPU; every $\lambda \geq 0$ gives a valid bound, so the best one is kept (Appendix B.5). The constraints also support incumbent-independent infeasibility pruning:

$$\min_{x \in B} \nu^\top (Ax + c) > 0, \qquad \nu \geq 0, \quad \Longrightarrow \quad \mathcal{F}(B) = \emptyset. \tag{10}$$

GEAR applies this test to every box after propagation. Unit vectors $\nu = e_i$ need no search: a box is discarded when the linear bound of one constraint is positive on all of B. Constraints that are only jointly inconsistent need a nonnegative combination: for the boxes with the lowest bounds, ν is taken from the multipliers of the LP that minimizes the largest violation of $Ax + c \leq 0$ over B, solved on the CPU, and (10) is then evaluated on the GPU (Appendix B.5). Duality, update rules, and certificate derivations are detailed in Appendices A.4 and A.12.

Constraint-driven clipping to shrink the feasible region. Dual optimization strengthens the use of existing linear bounds, but cannot recover information lost when those bounds were constructed. To address this limitation, GEAR feeds constraint information back into propagation by tightening the input and intermediate domains via domain clipping [73]. Relaxed clipping derives coordinate-wise input bounds, replacing B by B' with $\mathcal{F}(B) \subseteq \mathcal{P}(B) \subseteq B' \subseteq B$; it need not compute the smallest enclosing box. Complete clipping instead exploits $\mathcal{P}(B)$ when bounding an intermediate variable in the computation graph. For a propagated linear lower bound $\underline{z}(x)$, it targets $\min_{x \in \mathcal{P}(B)} \underline{z}(x)$ through a dual of the same form as (9), with separate multipliers; upper bounds are handled symmetrically. While [73] uses coordinate descent to handle multiple constraints, which works well when constraints are few in NN verification settings, it becomes slow as the number of NLP constraints grows. To address this, we use gradient descent to optimize the dual variables over the active constraints.

Narrower intervals give tighter linear relaxations of nonlinear operators and stronger linear bounds for the objective and constraints, and an empty interval certifies infeasibility. Clipping uses linear bounds already available from the parent or an earlier propagation, and the tightened intervals need only contain values attained at feasible inputs, which is why (7) states validity on $\mathcal{F}(B)$. The Lagrangian bound thus extracts more from the existing linear bounds, while clipping uses them to improve the domains on which later linear bounds are built (Appendices A.5 and A.11).

Tighter nonlinear operator relaxations. Clipping tightens the domains of nonlinear operators, but the resulting bounds can still be loose when the operator relaxations themselves are weak. GEAR therefore strengthens these local relaxations while keeping one linear bound per output. For a product, McCormick's construction gives two lower and two upper linear planes on a rectangle [39]. A lifted LP can retain all four, while backward linear bound propagation selects a plane according to the sign of its coefficient. GEAR compares the relevant candidates by closed-form concretization: for a nonnegative coefficient, it chooses the lower plane with the largest box minimum; for a negative coefficient, the upper plane with the smallest box maximum. This maximizes the local contribution among the candidates without iterative relaxation-parameter optimization. For the product example in Figure 4, the two lower facets concretize to -10 and 2 on the same rectangle, illustrating that a different valid choice can matter without adding more stored planes. For quotients, GEAR constructs planes directly for the two-input operator instead of composing reciprocal and product relaxations. Candidate planes are used only on denominator intervals excluding zero and only after their validity is established over the entire rectangle. This avoids an unnecessary intermediate relaxation while retaining the same linear interface. Supported univariate operators use secant-tangent constructions, with the tangent slope matched to the secant slope on fixed-curvature intervals. Together, these changes improve the linear bounds that the Lagrangian bound and clipping use, rather than only re-weighting them. However, the relaxation can still be weaker than an LP that keeps all McCormick planes at once; GEAR trades that strength for batched propagation and relies on spatial refinement to reduce the remaining error. Operator formulas, domain guards, and comparisons are in Appendix A.7 and Figure 4.

3.4 Scalable Primal Algorithm on GPU with Projected ALM and Linear Bounds

The mechanisms above strengthen the dual side of global optimization, but even a tight lower bound cannot enable dominance pruning without a feasible incumbent. To efficiently find such incumbents across many boxes, GEAR introduces a GPU-batched projected augmented Lagrangian method that jointly pursues objective improvement and constraint satisfaction. The choice is computational as well as algorithmic: fixed-budget first-order steps need only graph evaluations, automatic differentiation, and elementwise projection. They batch across searches without the per-iteration KKT factorization used by conventional interior-point methods [59]. This makes the projected ALM compatible with the many-box workload of the bounding procedure, rather than requiring a separate nonlinear solver invocation per box.

Adaptive feasibility-objective balancing. The augmented Lagrangian keeps objective improvement and feasibility restoration in the same differentiable search. For $\mu \geq 0$ and $\rho > 0$, GEAR uses the inequality-form function [49]

$$\phi_\rho(x;\mu) = f_0(x) + \frac{\rho}{2}\sum_{i=1}^{m}[f_i(x) + \mu_i/\rho]_+^2 - \frac{\|\mu\|_2^2}{2\rho}, \tag{11}$$

where $[t]_+ = \max\{t, 0\}$. Each outer iteration applies a fixed number of Adam descent steps projected onto B, then updates $\mu \leftarrow [\mu + \rho f(x)]_+$, with $f = (f_1, \ldots, f_m)$. The penalty increases when the maximum violation fails to decrease sufficiently. In the inner gradient, constraint i receives the effective weight $[\mu_i + \rho f_i(x)]_+$. The multiplier therefore carries constraint emphasis across outer iterations, while the current violation adjusts that emphasis within an iteration. This is the mechanism by which ALM couples objective progress to feasibility restoration rather than relying on one fixed penalty coefficient. Gradients use the original objective and constraints, including the exact linear constraints, rather than replacing the NLP by its relaxation. These updates seek improved incumbents; they are not relied upon to certify a lower bound or to solve every local subproblem.

How the dual helps the primal. Although λ weights the linear bounds in the Lagrangian bound (9) whereas μ weights the exact nonlinear constraints in (11), the dual evaluation still provides structural priors that spare the primal search from cold-starting on B. With λ^{best} the best evaluated Lagrangian multipliers and x^{best} an inner minimizer of (9), GEAR starts the search at x^{best}, which already lies in the low-objective basin where the lower bound is attained, and initializes μ with a nonnegative rescaling of λ^{best} so the first outer iteration immediately emphasizes the constraints that bottleneck the relaxation rather than accumulating multiplier weights from zero; neither step assumes that x^{best} is feasible or that λ equals the NLP multipliers. A squared violation penalty on the linear constraints can further guide the search by keeping early gradient steps near the relaxed feasible region. A candidate updates UB only after direct evaluation confirms feasibility (Section 2) and a better objective value, so warm starts and penalties affect progress, not incumbent validity (Appendix A.8).

3.5 Branching and the Full Algorithm

Even with stronger dual bounds and feasibility-oriented primal search, some boxes remain unresolved because their nonlinear relaxations are still too loose. GEAR therefore applies spatial refinement, reconstructing tighter relaxations on smaller domains to progressively close the remaining gap.

Multiplier-weighted branching. Splitting a box while keeping its linear bounds fixed need not raise the minimum across both children; the intended gain comes from reconstructing the nonlinear relaxations on the refined domains. Smart branching (SB) [13, 12], developed for input branch-and-bound in NN verification, therefore splits each input coordinate in turn, computes a fast lower bound on both children, and takes the coordinate with the largest minimum child bound. This needs two bound computations per coordinate and targets a single output. In constrained optimization a split can help by raising the objective's bound or by ruling out a constraint, so GEAR by default scores coordinates without bounding any child: it reuses the coefficients of its linear bounds and the Lagrangian multipliers λ^{best} to score coordinates by weighted linear variation:

$$S_j = w_j \left(|a_{0,j}| + \sum_{i=1}^{m} \lambda_i^{\text{best}} |a_{i,j}| \right), \qquad w_j = \overline{x}_j - \underline{x}_j. \tag{12}$$

Width represents the spatial extent available for refinement, while the weighted coefficients identify directions emphasized by the current bound. Summing absolute contributions avoids cancellation

between objective and constraint terms. The score is a heuristic, not an exact prediction of child bounds. Maximization is restricted to coordinates with $w_j \geq \tau \max_k w_k$, for a fixed $\tau \in (0, 1]$, preventing narrow coordinates from indefinitely displacing wide ones (Appendix A.9). We call branching on the coordinate that maximizes S_j λ-SB.

The full algorithm. Each sBaB subproblem is a box with its own lower bound and linear bounds (7). Each iteration selects up to K open boxes with the lowest lower bounds and processes them as one batch. For each box, GEAR computes new linear bounds by bound propagation with constraint-aware intermediate tightening, tests infeasibility (10), and optimizes the Lagrangian bound (9); the box is discarded if it is infeasible or if its lower bound reaches UB. Projected ALM on open boxes, started from the Lagrangian bound (Section 3.4), can lower UB and so discard more boxes. Each remaining box is split using λ-SB. Its two children inherit its lower bound and linear bounds, which remain valid on smaller boxes, so each child can be clipped right away, before its own propagation. New linear bounds then replace the inherited ones while the linear constraints stay fixed, so the memory per box does not grow with depth.

Soundness decoupled from heuristic progress. Correctness follows the sBaB principle of Section 2: clipping preserves feasible points, infeasibility pruning requires a contradiction, dominance pruning requires $\mathrm{LB}(B) \geq \mathrm{UB}$, and incumbents are checked on the original constraints, so the global certificate is strictly decoupled from whether any heuristic step (dual ascent, ALM, or branching) succeeds. The solver stops at the relative gap of Equation (13), on an empty queue, or at the time limit. The full algorithm and invariant proof appear in Appendices A.10 and A.12; finite termination needs the further assumptions of Appendix A.13.

4 Experiments

Benchmarks. We evaluate GEAR on 505 bounded continuous NLP instances: 265 from `gams_world` and 240 from `minlplib`. Each retained instance is translated into a nonlinear computation graph with continuous, box-bounded variables. Dataset construction, translation and filtering rules, and validation fields are detailed in Appendix C.

Baselines and hardware. We compare against six global solvers, SCIP [9], Gurobi [28], BARON [71], Couenne [6], LindoGlobal [37] and Xpress [8], and six local solvers, Knitro [15], IPOPT [59], CONOPT [21], LGO [46], MINOS [43] and SNOPT [25], all through AMPL. All solvers run on one workstation (8-core 16-thread AMD Ryzen 9700X, 256 GB RAM, one NVIDIA RTX 5090) with a 180 s limit per instance. Every baseline receives all 16 hardware threads in the layout it natively supports and runs one configuration tuned on a 40-instance subset, with feasibility tolerance 10^{-6} and relative gap 10^{-4} as in Equation (13) (Appendix C.5). GEAR runs one configuration on every instance (Figure 5).

Solved criterion. A run counts as *solved* when (i) its returned point satisfies every constraint and bound of the original model, with equalities kept exact, to absolute tolerance 10^{-6} (baseline points are re-evaluated in AMPL), and (ii) its primal value P and certified lower bound D have a relative gap of at most 10^{-4}:

$$(P - D) \,/\, \max(1, |P|, |D|) \;\leq\; 10^{-4}. \tag{13}$$

Local solvers are credited only when their feasible P matches a bound certified by a global solver; flagged or contradicted certificates and infeasibility verdicts are treated as in Appendix C.5.

4.1 Main Results

Coverage. Table 1 reports the comparison. GEAR certifies 421 of the 505 instances (83.4%), 20 more than the strongest global baseline, Gurobi (401), and 57 more than SCIP (364). The lead is concentrated on `gams_world`, where GEAR certifies 251/265 instances (94.7%) against 227 for Gurobi and 209 for SCIP; on `minlplib`, GEAR (170) is ahead of SCIP (155) and behind Gurobi (174). Among the local solvers, Knitro matches a certified optimum on 401 instances. GEAR returns a feasible point on 501 instances.

Runtime profile and complementarity. The baselines solve easy instances in hundredths of a second, whereas GEAR pays a start-up cost of a few seconds for graph compilation and GPU initialization (Figure 2a,b). It overtakes LindoGlobal, Couenne, BARON and Xpress at 6-9 s, SCIP at 15 s and

	gams_world		minlplib		Total		
Global Solver	Feas.	Solved	Feas.	Solved	Feas.	Solved	%
GEAR	**264**	**251**	**237**	170	**501**	**421**	**83.4**
Gurobi	254	227	218	**174**	472	401	79.4
SCIP	262	209	219	155	481	364	72.1
BARON	217	168	186	129	403	297	58.8
Xpress	218	176	173	120	391	296	58.6
Couenne	222	182	152	100	374	282	55.8
LindoGlobal	180	164	169	110	349	274	54.3

	gams_world		minlplib		Total		
Local Solver*	Feas.	Solved	Feas.	Solved	Feas.	Solved	%
GEAR	**264**	**251**	**237**	170	**501**	**421**	**83.4**
Knitro*	263	231	225	170	488	401	79.4
IPOPT*	256	203	202	117	458	320	63.4
CONOPT*	259	205	205	91	464	296	58.6
LGO*	249	206	160	78	409	284	56.2
MINOS*	247	193	133	53	380	246	48.7
SNOPT*	235	182	133	51	368	233	46.1

Table 1: Instances solved in 180 s (gams_world 265, minlplib 240, total 505); left: GEAR and the global solvers, right: GEAR and the local solvers, as in Figure 2. Feas.: a returned point feasible at 10^{-6}. *Local solvers, credited only when they match a certified bound. Bold/underline: best/second among GEAR and the global solvers.

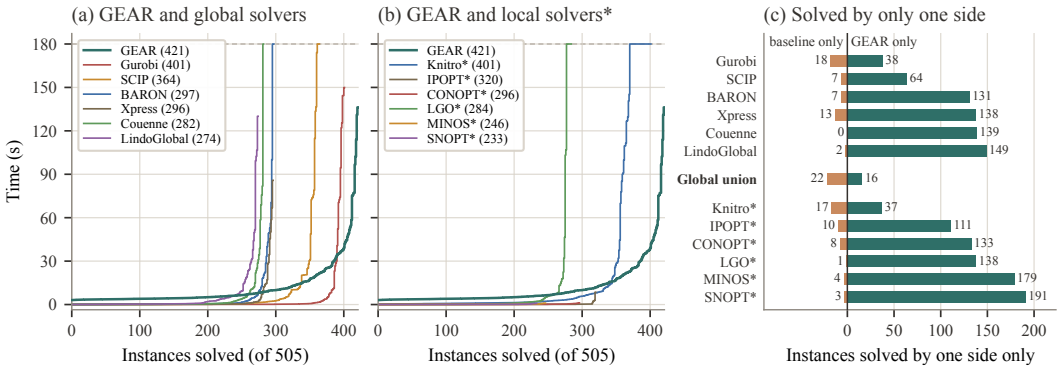

Figure 2: **Solve times and exclusive solves** (180 s, solved criterion of Equation (13)). (a, b) Cactus plots for GEAR with the global and with the local solvers (*, counted only when matched to a globally certified bound). (c) Instances solved by GEAR but not by the baseline (right of zero) and by the baseline but not by GEAR (left of zero), on one scale; Global union: the six global solvers.

Gurobi at 35 s, so its lead comes from the hard tail: on instances both sides certify, the fastest global solver needs a median of 0.03-0.06 s against 4-29 s for GEAR. GEAR solves 38-149 instances that each global solver misses, against 0-18 in the other direction (Figure 2c). Against the union of the six global solvers (427 instances), GEAR adds 16 instances and misses 22; together they cover 443/505.

Where the GPU helps. By problem class (Appendix D.1), the lead comes from low- to medium-dimensional instances with few constraints and composite nonlinearities: GEAR certifies 322/330 instances with at most 20 constraints and 131/147 with transcendental terms, which contain all 16 instances no global solver certifies. Factorable relaxations of composed exponential, logarithmic and power terms are loose. GEAR closes 293 of its 421 solved instances at the root, including 12 of those 16 (Appendix D.2), and the other 128 in the GPU tree, with a median of 318 boxes and at least 10^5 boxes on five of them. On large structured models (pooling, wastewater, water networks, circle packing), whose certificates rely on RLT and other cutting planes, the global solvers remain stronger, and GEAR solves no instance with more than 1,000 variables.

4.2 Ablation Studies

Table 2 adds one group of components per row, from the verification pipeline to the full solver. The first row is the most common setting in NN verifiers (Section 3.1): input branch-and-bound on bound propagation with a PGD attack on the box. It solves nothing. A verifier's attack looks for a counterexample: its points may violate the constraints, so their objective values are not valid pruning thresholds, and it keeps no feasible point to return. PGD itself can reach good points: on the 95 instances without constraints, where every point of the box is feasible, the attack reaches the optimal value on 56, but the pipeline does not keep these points as incumbents. Presolve and the components of Appendix B other than the lifted relaxation supply feasible points and root certificates: 72 instances close at the root and 147 in the branch-and-bound that follows. The projected ALM search (Section 3.4) adds 50. Constraint-aware dual bounds with clipping (Section 3.3) are the largest step of the GPU method (+97): once incumbents exist, the bounds must use the constraints to prune the boxes that infeasible

Configuration	Feasible	Optimal*	Solved	Δ
Bound propagation + PGD	95	56	0	-
+ presolve and CPU solving components	480	396	219	+219
+ projected ALM primal search (GPU)	496	471	269	+50
+ constraint-aware dual bounds and clipping (GPU)	498	472	366	+97
+ lifted and refined operator relaxation (GPU)	501	478	410	+44
+ λ-SB branching (GEAR)	501	478	**421**	+11

Table 2: Cumulative ablation of GEAR. Feasible: the returned point satisfies every constraint to 10^{-6}. *Optimal: that point is also within the relative gap of the best known value, verified by any global solver. Solved: Equation (13) holds, which requires a certified lower bound. Δ: change in solved from the previous components.

points keep open. The lifted relaxation (Appendix B.4), refined operator relaxations (Section 3.3) and λ-SB branching (Section 3.5) close 55 more. The dual and branching steps reuse the same propagated linear bounds, so one set of linear bounds serves every component of the solver. With the ALM search, GEAR already holds an optimal point on 471 instances; the later rows raise the solved count from 269 to 421 and add only 7 optimal points, so they finish the proof rather than find better points.

5 Related Works

Linear bound propagation beyond neural network verification. Linear bound propagation [70, 52, 53, 60, 65] bounds every output of a computation graph, and constraint-aware verifiers tighten it with split constraints, cutting planes and domain clipping [19, 62, 69, 72, 73], also for general nonlinearities and preimages [51, 33]. These methods certify a property against a fixed threshold and never need a feasible point. In optimization the threshold is the value of an incumbent that the solver must find and improve, and an objective-only bound can stay below it forever (Section 3.1). Our view is that the linear bounds verification already computes can drive both sides of this search: combined with Lagrangian multipliers they give a constraint-aware lower bound and an infeasibility test, and the same bounds and multipliers start the primal search and score the branching.

Per-node strength versus batched throughput. Classical global solvers [71, 7, 58, 9, 40, 37, 28] relax a factorable program [55, 36, 31] and solve a relaxation, most often an LP, at every node, strengthened by bound tightening, cutting planes and RLT [50, 26]. An LP-based node gets the strength of an LP that keeps all McCormick planes [39], but nodes are processed one after another, each with its own LP. GEAR makes the opposite trade: it keeps one linear bound per output, computed without an LP solve, so each box is bounded more loosely, but many boxes are bounded in one batched step. The experiments show the two designs to be complementary (Section 4): GEAR leads on composite nonlinearities in low and moderate dimension, and per-node LP strength wins on large structured models.

GPU optimization. GPUs already accelerate parts of both fields. In verification, polyhedral and linear bound propagation run on GPUs [42, 66] and branch-and-bound subproblems are bounded in batches [66, 62]. In mathematical programming, first-order methods such as PDLP [3, 4] replace factorizations with matrix-vector products to solve one large LP, and commercial solvers parallelize their tree search over CPU threads [8]. In deterministic global optimization, GPUs have offloaded McCormick relaxations [27] and interval bounds on partitioned subdomains [68]. GEAR takes the verification route for a global NLP search: the branch-and-bound frontier, its linear bounds and the primal search are tensors in a deep-learning framework [45], so one GPU step can process thousands of boxes, instead of parallelizing within one LP or across CPU threads. Detailed discussion in Appendix E.

6 Conclusion

GEAR demonstrates that linear bound propagation can serve as a unified GPU-batched interface for bounded continuous global NLP, reusing valid linear relaxations for lower bounding, infeasibility detection, primal recovery, and branching to solve a complementary class of highly nonlinear instances of low and moderate dimension.

Future works and limitations. GEAR assumes continuous variables and a finite box; mixed-integer and unbounded NLPs are left to future work. More limitations are discussed in Appendix D.3.

Acknowledgments and Disclosure of Funding

Huan Zhang is supported in part by the AI2050 program at Schmidt Sciences (AI2050 Early Career Fellowship) and NSF (2543005, 2331967). Grani A. Hanasusanto is supported in part by NSF (CCF-2343869 and ECCS-2404413). We thank Yifei Li from UIUC for running part of the AMPL baselines. We thank the anonymous reviewers for their constructive feedback.

References

[1] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998.

[2] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1-2):3–39, 2020.

[3] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. *Advances in Neural Information Processing Systems*, 34:20243–20257, 2021.

[4] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. PDLP: A practical first-order method for large-scale linear programming. *arXiv preprint arXiv:2501.07018*, 2025.

[5] Stanley Bak. nnenum: Verification of ReLU neural networks with optimized abstraction refinement. In *NASA Formal Methods Symposium*, pages 19–36. Springer, 2021.

[6] Pietro Belotti. Couenne: a user's manual, 2009.

[7] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods & Software*, 24(4-5):597–634, 2009.

[8] Timo Berthold, James Farmer, Stefan Heinz, and Michael Perregaard. Parallelization of the FICO Xpress-Optimizer. *Optimization Methods and Software*, 33(3):518–529, 2018.

[9] Ksenia Bestuzheva, Antonia Chmiela, Benjamin Müller, Felipe Serrano, Stefan Vigerske, and Fabian Wegscheider. Global optimization of mixed-integer nonlinear programs with SCIP 8, 2023.

[10] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[11] Robin Brown, Edward Schmerling, Navid Azizan, and Marco Pavone. A unified view of SDP-based neural network verification through completely positive programming, 2022.

[12] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020.

[13] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and M. Pawan Kumar. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31, 2018.

[14] Samuel Burer and Dieter Vandenbussche. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Mathematical Programming*, 113(2):259–282, 2008.

[15] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[16] Shaoru Chen, Lekan Molu, and Mahyar Fazlyab. Verification-aided learning of neural network barrier functions with termination guarantees. *arXiv preprint arXiv:2403.07308*, 2024.

[17] Hong-Ming Chiu, Hao Chen, Huan Zhang, and Richard Y. Zhang. SDP-CROWN: Efficient bound propagation for neural network verification with tightness of semidefinite programming. In *International Conference on Machine Learning (ICML)*, 2025. ICML 2025.

[18] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[19] Liam Davis, Duo Zhou, Huan Zhang, Guy Katz, Clark Barrett, and Haoze Wu. Lookahead branching for neural network verification. In Diego Calvanese, editor, *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence, IJCAI-26*, pages 518–526. International Joint Conferences on Artificial Intelligence Organization, 8 2026. Main Track.

[20] Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip HS Torr, and M Pawan Kumar. Improved branch and bound for neural network verification via Lagrangian decomposition. *arXiv preprint arXiv:2104.06718*, 2021.

[21] Arne Stolbjerg Drud. CONOPT—a large-scale GRG code. *ORSA Journal on Computing*, 6(2):207–216, 1994.

[22] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2020.

[23] Claudio Ferrari, Mark Niklas Müller, Nikola Jovanović, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*, 2022.

[24] Matteo Fischetti and Jason Jo. Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *CoRR*, abs/1712.06174, 2017.

[25] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[26] Ambros M. Gleixner, Timo Berthold, Benjamin Müller, and Stefan Weltge. Three enhancements for optimization-based bound tightening. *Journal of Global Optimization*, 67(4):731–757, 2017.

[27] Robert X. Gottlieb, Pengfei Xu, and Matthew D. Stuber. Automatic source code generation for deterministic global optimization with parallel architectures. *Optimization Methods and Software*, 41(2):308–346, 2026.

[28] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026.

[29] Joey Huchette, Gonzalo Muñoz, Thiago Serra, and Calvin Tsay. When deep learning meets polyhedral theory: A survey. *arXiv preprint arXiv:2305.00241*, 2023.

[30] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

[31] Aida Khajavirad and Nikolaos V. Sahinidis. A hybrid LP/NLP paradigm for global optimization relaxations. *Mathematical Programming Computation*, 10(3):383–421, 2018.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[33] Suhas Kotha, Christopher Brix, J. Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. Provably bounding neural network preimages. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[34] Jianglin Lan, Yang Zheng, and Alessio Lomuscio. Tight neural network verification via semidefinite relaxations and linear reformulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7272–7280, 2022.

[35] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[36] Leo Liberti and Constantinos C. Pantelides. An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, 36(2):161–189, 2006.

[37] Youdong Lin and Linus Schrage. The global solver in the LINDO API. *Optimization Methods & Software*, 24(4-5):657–668, 2009.

[38] Jingyue Lu and M Pawan Kumar. Neural network branching for neural network verification. *arXiv preprint arXiv:1912.01329*, 2019.

[39] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[40] Ruth Misener and Christodoulos A Floudas. ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2):503–526, 2014.

[41] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[42] Christoph Müller, Francois Serre, Gagandeep Singh, Markus Püschel, and Martin Vechev. Scaling polyhedral neural network verification on GPUs. *Proceedings of Machine Learning and Systems*, 3:733–746, 2021.

[43] Bruce A. Murtagh and Michael A. Saunders. MINOS 5.51 User's Guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, 2003. Originally published 1983; revised September 23, 2003.

[44] Carlos J. Nohra, Arvind U. Raghunathan, and Nikolaos V. Sahinidis. Spectral relaxations and branching strategies for global optimization of mixed-integer quadratic programs. *SIAM Journal on Optimization*, 31(1):142–171, 2021.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[46] János D. Pintér. GAMS/LGO nonlinear solver suite: key features, usage, and numerical performance, 2003.

[47] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[48] R Tyrrell Rockafellar. A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical programming*, 5(1):354–373, 1973.

[49] R. Tyrrell Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973.

[50] Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.

[51] Zhouxing Shi, Qirui Jin, Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. Neural network verification with branch-and-bound for general nonlinearities. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2025.

[52] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[53] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

[54] Chuangchuang Sun, Dong-Ki Kim, and Jonathan P How. ROMAX: Certifiably robust deep multiagent reinforcement learning via convex relaxation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5503–5510. IEEE, 2022.

[55] Mohit Tawarmalani and Nikolaos V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Nonconvex Optimization and Its Applications. Springer, 2002.

[56] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[57] Andreas Venzke and Spyros Chatzivasileiadis. Verification of neural network behaviour: Formal guarantees for power system applications. *IEEE Transactions on Smart Grid*, 12(1):383–397, 2020.

[58] Stefan Vigerske and Ambros Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2018.

[59] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[60] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[61] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *USENIX Security Symposium*, 2018.

[62] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[63] Eric Wong, Tim Schneider, Joerg Schmitt, Frank R Schmidt, and J Zico Kolter. Neural network virtual sensors for fuel injection quantities with provable performance specifications. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1753–1758. IEEE, 2020.

[64] Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural Lyapunov control for discrete-time systems. *Advances in Neural Information Processing Systems*, 36:2939–2955, 2023.

[65] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.

[66] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations (ICLR)*, 2021.

[67] Lujie Yang, Hongkai Dai, Zhouxing Shi, Cho-Jui Hsieh, Russ Tedrake, and Huan Zhang. Lyapunov-stable neural control for state and output feedback: a novel formulation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[68] Hongzhen Zhang, Tim Kerkenhoff, Neil Kichler, Manuel Dahmen, Alexander Mitsos, Uwe Naumann, and Dominik Bongartz. Accelerating deterministic global optimization via GPU-parallel interval arithmetic. *arXiv preprint arXiv:2507.20769*, 2025.

[69] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems*, 35:1656–1670, 2022.

[70] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[71] Yi Zhang and Nikolaos V. Sahinidis. Solving continuous and discrete nonlinear programs with BARON. *Computational Optimization and Applications*, 92:1123–1161, 2025.

[72] Duo Zhou, Christopher Brix, Grani A Hanasusanto, and Huan Zhang. Scalable neural network verification with branch-and-bound inferred cutting planes. *Advances in Neural Information Processing Systems*, 37:29324–29353, 2024.

[73] Duo Zhou, Jorge Chavez, Hesun Chen, Grani A Hanasusanto, and Huan Zhang. Clip-and-verify: Linear constraint-driven domain clipping for accelerating neural network verification. *Advances in Neural Information Processing Systems*, 38:174849–174895, 2026.

Appendix Contents

A Methodology Details and Proofs **17**
- A.1 Roadmap and Soundness Requirements . . . 17
- A.2 Batched Linear Bounds from a Multi-Output Graph . . . 17
- A.3 Admissible Linear Constraint Relaxations and Tolerance Convention . . . 18
- A.4 Why Linear Mixing Can Improve a Bound . . . 19
- A.5 How Clipping Works in GEAR . . . 20
- A.6 Safe Linear Concretization over Clipped Relaxation Domains . . . 21
- A.7 Operator Relaxations and Validity Domains . . . 22
- A.8 Primal Recovery Details . . . 26
- A.9 Branching Calibration . . . 28
- A.10 Overall Algorithm of GEAR . . . 28
- A.11 Node Lifecycle and Data Dependencies . . . 29
- A.12 Proofs of the Sound Node Invariants . . . 30
- A.13 Conditional Finite Termination . . . 31

B Complete Description of the Solver **33**
- B.1 Compilation and Presolve . . . 33
- B.2 Root Schedule . . . 35
- B.3 Root Dual Bounds and Certificates . . . 35
- B.4 Lifted Linear Relaxation . . . 37
- B.5 One Step of the GPU Branch-and-Bound . . . 42
- B.6 Primal Search . . . 44
- B.7 Numerics . . . 47
- B.8 Budgets . . . 47

C Benchmark and Experimental Protocol **48**
- C.1 Source Pools . . . 48
- C.2 Canonicalization and Graph Construction . . . 48
- C.3 Inclusion and Exclusion Policy . . . 48
- C.4 Translation Precision and Primal-Dual Soundness . . . 48
- C.5 Baseline Protocol . . . 49
- C.6 GEAR Logs and Scoring . . . 49
- C.7 Released Tables . . . 49

D Analysis of the Results and Limitations **50**
- D.1 Which Problems Benefit Most from the GPU . . . 50
- D.2 Where GEAR Wins and Where It Loses . . . 50
- D.3 Limitations . . . 51

E More Related Works **52**

Appendix A gives the derivations and proofs for the GPU method of Section 3. Appendix B describes every component of the solver evaluated in Section 4. Appendix C gives the benchmark, the baselines and how runs are scored. Appendix D analyzes the results and states the limitations. Appendix E covers further related work.

A Methodology Details and Proofs

A.1 Roadmap and Soundness Requirements

This section supplies the derivations and implementation details omitted from Section 3. It uses the same notation as the main text: $B = [\underline{x}, \overline{x}]$, $\underline{f}_i(x) = a_i^\top x + c_i$, $\mathcal{P}(B) = \{x \in B : Ax + c \leq 0\}$, and the box-domain LP dual $d_B(\lambda)$ of (9). The vector λ belongs to the propagated-row LP, whereas μ belongs to the original-NLP primal heuristic. A bound on $\mathcal{F}(B)$ suffices for all pruning arguments; it need not hold at infeasible inputs excluded during intermediate clipping.

Main-text component	Appendix expansion
Section 3.2: representation	Appendices A.2 and A.3: graph, tensor layout, and validity under clipping.
Section 3.3: objective bound	Appendices A.4 and A.12: LP duality, supergradients, best-iterate updates, and infeasibility certificates.
Section 3.3: clipping	Appendices A.5 and A.6: coordinate updates, intermediate-bound duals, and conservative concretization.
Section 3.3: operator relaxations	Appendix A.7: plane selection, relaxation formulas, validity checks, and examples.
Section 3.4: primal search	Appendix A.8: projected ALM, warm starts, linear-row penalties, and incumbent acceptance.
Section 3.5: integration	Appendices A.9 to A.11: data dependencies, branching, and the full algorithm; Appendix A.13: conditional termination.

Table 3: Location of the details supporting each main-text mechanism.

Certificates versus heuristics. The solver may terminate dual and primal optimization after finite budgets. Pruning nevertheless requires a valid certificate: a lower bound, a contradiction, or a comparison with a feasible incumbent. The relevant requirements are summarized in Table 4. In this appendix a *linear feasibility cut* means the same inequality $\underline{f}_i(x) \leq 0$ as a *linear constraint relaxation*; it does not imply a separately generated cutting-plane family.

Object	Required for validity	Efficiency choice
Affine row	$\underline{f}_i \leq f_i$ on the node's feasible set	Relaxation selection and batching
Constraint relaxation	$\mathcal{F}(B) \subseteq \mathcal{P}(B)$	Inherited-row replacement schedule
Objective bound	Conservative evaluation of $d_B(\lambda)$ with $\lambda \geq 0$	Initialization, optimizer, and iteration budget
Infeasibility test	A strictly positive conservative certificate	Individual rows or nonnegative mixtures
Clipping	Every feasible input and its intermediate values are retained	Update order and number of sweeps
Incumbent	Original-constraint feasibility and direct objective evaluation	ALM schedule, seeds, and warm starts
Branching	Children cover the surviving parent domain	Coefficient score and eligible coordinates

Table 4: Soundness requirements and heuristic choices. The numerical requirements are conditional on conservative evaluations as detailed in Appendix A.3; heuristic choices do not replace them.

A.2 Batched Linear Bounds from a Multi-Output Graph

The objective and all constraints are the outputs of the graph F in (2). A constraint $f_i(x) = a_i^\top x + c_i$ that is linear in x is reproduced exactly by propagation, and when the parser exposes such constraints as an explicit matrix, that matrix is also retained directly. The residual $\max_{i \geq 1} f_i(x)$ can be bounded from constraint rows without appending a hard-max graph operator.

Backward linear bound propagation. For a scalar output selector s, a direct bound-propagation pass seeks

$$L_s(x) = a_s^\top x + c_s \leq s^\top F(x) \tag{14}$$

on its validity set. If $z = h(y)$ has valid linear relaxations on the intermediate interval, backward linear bound propagation uses

$$\gamma h(y) \geq \gamma_+ \underline{h}(y) + \gamma_- \overline{h}(y), \qquad \gamma_+ = \max\{\gamma, 0\}, \quad \gamma_- = \min\{\gamma, 0\}. \tag{15}$$

Affine operators are substituted exactly. Applying this inequality in reverse topological order proves output-row validity. With ordinary intervals it holds on B; with constraint-tightened intervals it holds on $\mathcal{F}(B)$, whose graph trajectories remain within those intervals. Combining cached lower rows with nonnegative weights is also valid, but need not give the same row as rerunning bound propagation on the combined output: relaxation selection and coefficient cancellation can differ. The dual uses cached rows and does not rerun propagation.

Bound records and tensor layout. A mathematical record for an output row is

$$\mathfrak{b}_i(B) = (i, a_i, c_i, V_i), \qquad \mathcal{F}(B) \subseteq V_i, \quad a_i^\top x + c_i \leq f_i(x) \text{ on } V_i. \tag{16}$$

Relaxation parameters and numerical error budgets can accompany this record; the proof only uses its validity statement. For K boxes, the logical coefficient arrays are

$$\mathsf{A} \in \mathbb{R}^{K \times (m+1) \times n}, \qquad \mathsf{c} \in \mathbb{R}^{K \times (m+1)}, \tag{17}$$

where $\mathsf{A}[k, i, :] = a_i^{(k)}$ and $\mathsf{c}[k, i] = c_i^{(k)}$. Exact affine rows are invariant across boxes and may be broadcast rather than duplicated. Each dual update costs $O(mn)$ arithmetic per box. Primal steps still require graph evaluations, and changed intermediate intervals can require new propagation; only operations on the fixed cached rows avoid a graph traversal.

Residual bounds without a max relaxation. Let $r(x) = \max_{1 \leq i \leq m} f_i(x)$, with the unconstrained case handled without a residual test. Each constraint lower row is a lower bound on r. More generally, for $\omega \geq 0$ and $\mathbf{1}^\top \omega = 1$,

$$\sum_i \omega_i \underline{f}_i(x) \leq \sum_i \omega_i f_i(x) \leq r(x) \qquad (x \in \mathcal{F}(B)).$$

If the box minimum of this row is positive, no feasible point exists. Normalization gives a residual interpretation; it is not necessary for the nonnegative-mixture contradiction in (10). The selected mixture family and its search budget determine which infeasible nodes are detected, not whether a successful certificate is valid.

A.3 Admissible Linear Constraint Relaxations and Tolerance Convention

Admissibility means preserving the feasible set, including when rows have been derived under earlier constraints. If a row was obtained at a box B_k, the required statement for use at a descendant $B \subseteq B_k$ is

$$a_k^\top x + c_k \leq f_{i_k}(x) \quad \forall x \in \mathcal{F}(B_k). \tag{18}$$

A record of its constraint index and provenance may be written

$$\mathfrak{c}_k = (i_k, a_k, c_k, B_k). \tag{19}$$

Since $\mathcal{F}(B) \subseteq \mathcal{F}(B_k)$, every feasible descendant point satisfies $a_k^\top x + c_k \leq 0$. A row need not be valid throughout B_k if its construction used feasible-only intermediate bounds. The constraint-tightening argument is inductive: each step starts with valid rows, retains all feasible trajectories, and derives new valid rows from the retained intervals.

Storage and forwarding. The initial active set contains the original affine constraints. At a child it additionally contains the nonlinear rows inherited from its parent. After fresh propagation these nonlinear rows are replaced, not accumulated with all ancestors. Thus the refreshed set does not grow with depth. Replacing rows can lose relaxation strength, but not feasibility preservation. Scalar lower bounds are separately retained by taking the maximum with the inherited bound.

Feasibility tolerances. The functions f_i throughout the method already include the tolerances of Section 2. For example, an equality becomes $g(x) - \tau_{\text{feas}} \leq 0$ and $-g(x) - \tau_{\text{feas}} \leq 0$. Feasibility is then tested against zero, not against the same positive tolerance a second time. Separate inequality and equality tolerances can be absorbed into their respective rows under the same convention. These tolerances define the problem being solved; they are distinct from conservative floating-point error allowances.

Conservative numerical evaluation. The proofs below are exact-arithmetic statements. To use a computed row $(\widehat{a}, \widehat{c})$, suppose an error allowance satisfies

$$|(\widehat{a} - a)^\top x + (\widehat{c} - c)| \le \epsilon_{\text{aff}}(B) \quad \text{on the required validity set.}$$

Then $\widehat{a}^\top x + \widehat{c} - \epsilon_{\text{aff}}(B)$ remains a valid lower row. A row used for clipping must be weakened before it can remove points; subtracting an error only from a final reported scalar does not protect an earlier clipping step. Scalar minima must be rounded or bounded downward, retained intervals outward, and incumbent values conservatively relative to the original model. The allowances must cover operator relaxations, graph translation where relevant, propagation, coefficient mixing, and concretization. The existence and implementation of such allowances are requirements, not consequences of writing a final subtraction.

A.4 Why Linear Mixing Can Improve a Bound

The relaxation targeted by dual ascent is the fixed-row LP in (8), not the original NLP or a lifted multi-facet LP. For a nonempty finite box and fixed affine rows, set

$$a(\lambda) = a_0 + A^\top \lambda, \qquad c(\lambda) = c_0 + c^\top \lambda. \tag{20}$$

The dual function is exactly (9), with inner domain B.

Proposition A.1 (Dual of the propagated-row LP)**.** *For every $\lambda \ge 0$, $d_B(\lambda) \le p_{\text{LP}}(B) \le p^\star(B)$. If $\mathcal{P}(B) \ne \emptyset$, then $\max_{\lambda \ge 0} d_B(\lambda) = p_{\text{LP}}(B)$. If $\mathcal{P}(B) = \emptyset$, then d_B is unbounded above.*

Proof. For $x \in \mathcal{P}(B)$, the mixed objective is at most $\underline{f}_0(x)$, proving weak duality. A nonempty $\mathcal{P}(B)$ is compact, so the finite LP optimum exists; LP strong duality applied with the box bounds retained in the inner minimization gives equality. For the empty case, the compact convex set $\{Ax + c : x \in B\}$ is disjoint from the closed negative orthant. A strict separating vector can be chosen as $\nu \ge 0$ and satisfies $\delta = \min_{x \in B} \nu^\top (Ax + c) > 0$. Hence $d_B(t\nu) \ge d_B(0) + t\delta \to +\infty$. □

Inner minimizer and supergradient. The inner minimization is coordinate-separable. One minimizing corner is

$$x_{\lambda,j} = \begin{cases} \underline{x}_j, & a_j(\lambda) \ge 0, \\ \overline{x}_j, & a_j(\lambda) < 0. \end{cases}$$

If a coefficient is zero, any value in its interval is minimizing. The function d_B is concave and piecewise linear. Evaluating the mixed objective for λ' at x_λ proves

$$d_B(\lambda') \le d_B(\lambda) + (Ax_\lambda + c)^\top (\lambda' - \lambda). \tag{21}$$

Thus the row residual vector is a supergradient. It identifies constraints violated by the inner minimizer, which need not be feasible for the LP.

Fixed-budget updates and inherited bounds. From a nonnegative initialization, take T projected Adam ascent steps using the supergradient, retaining the best conservative evaluation:

$$\text{LB}_{\text{new}}(B) = \max_{0 \le t \le T} \widehat{d}_B(\lambda^{(t)}), \qquad \widehat{d}_B(\lambda) \le d_B(\lambda). \tag{22}$$

The stored node bound is $\max\{\text{LB}_{\text{parent}}, \text{LB}_{\text{new}}(B)\}$. Let λ^{best} maximize the newly evaluated values, and $x^{\text{best}} = x_{\lambda^{\text{best}}}$; this is the warm-start pair, even if the inherited scalar happens to be stronger. Evaluating $\lambda = 0$ provides an objective-only backstop. The theory of Appendix A.13 explicitly assumes a sufficiently accurate backstop; its use is not inferred merely from a finite Adam budget.

An illustrative improvement. For $\min_{x \in [0,1]} x$ subject to $1/2 - x \le 0$,

$$d_B(\lambda) = \lambda/2 + \min\{0, 1 - \lambda\}.$$

The objective-only value $d_B(0) = 0$ ignores feasibility; $d_B(1) = 1/2$ recovers the exact LP and NLP value. In general an improving direction requires ruling out the entire minimizing face, not only one arbitrary minimizing corner.

Active-face improvement. Let $X_0 = \arg\min_{x \in B} \underline{f}_0(x)$ and $h_i(x) = a_i^\top x + c_i$. For this finite polyhedral problem,

$$d'_B(0; e_i) = \min_{x \in X_0} h_i(x) =: \underline{v}_i. \tag{23}$$

Let $\mathcal{V}(B)$ denote the vertices of B, and define

$$\rho_i^B = \min_{\substack{v \in \mathcal{V}(B):\ \underline{f}_0(v) > d_B(0) \\ h_i(v) < \underline{v}_i}} \frac{\underline{f}_0(v) - d_B(0)}{\underline{v}_i - h_i(v)}, \tag{24}$$

where the minimum over the empty set is $+\infty$.

Theorem A.2 (Strict linear-mixture improvement)**.** *For $0 \le \eta < \rho_i^B$, $d_B(\eta e_i) - d_B(0) = \eta \underline{v}_i$. The minimizing set is contained in X_0; if $\underline{v}_i > 0$, sufficiently small positive steps strictly raise the bound. For a cap $\Lambda = [0, \lambda_{\max}]^m$ with $\lambda_{\max} > 0$,*

$$\max_{\lambda \in \Lambda} d_B(\lambda) - d_B(0) \ge [\underline{v}_i]_+ \min\{\rho_i^B/2, \lambda_{\max}\}.$$

Proof. On X_0, minimizing the perturbed objective gives $d_B(0) + \eta \underline{v}_i$. A vertex outside X_0 with $h_i(v) \ge \underline{v}_i$ cannot improve it for $\eta \ge 0$. Any other vertex can first match it at the ratio in (24). Before the minimum such ratio, an optimizer therefore remains in X_0. Evaluating an admissible step proves the final bound when $\underline{v}_i > 0$; otherwise $\lambda = 0$ proves its zero right-hand side. Monotonicity is asserted only on the stated interval, not along the whole ray. □

Best-iterate rate for a reference optimizer. The following bound concerns projected supergradient ascent, not Adam.

Theorem A.3 (Best-iterate rate for projected supergradient ascent)**.** *Let g be concave on a compact convex set Λ, let λ^* maximize it, and let the chosen supergradients satisfy $\|s^t\| \le G$. For $\lambda^{t+1} = \Pi_\Lambda(\lambda^t + \eta_t s^t)$ with $\eta_t = \eta/\sqrt{T}$,*

$$g(\lambda^*) - \max_{1 \le t \le T} g(\lambda^t) \le \frac{\|\lambda^1 - \lambda^*\|^2 + \eta^2 G^2}{2\eta\sqrt{T}}.$$

With $\eta_t = \eta/\sqrt{t}$, the corresponding weighted-iterate argument gives an $O(\log T/\sqrt{T})$ best-iterate bound.

Proof. Projection is nonexpansive, and the supergradient inequality gives

$$2\eta_t \big[g(\lambda^*) - g(\lambda^t)\big] \le \|\lambda^t - \lambda^*\|^2 - \|\lambda^{t+1} - \lambda^*\|^2 + \eta_t^2 G^2.$$

Sum over t and divide by $2\sum_t \eta_t$; the weighted mean gap bounds the smallest iterate gap. The stated schedules give the two bounds. □

A cap can lower the best attainable dual value relative to the unrestricted LP dual. Projected Adam does not inherit this theorem; finite-iterate validity follows independently from Proposition A.1 and conservative evaluation. Rescaling an LP multiplier to initialize ALM remains a warm-start heuristic, not a claim of original-NLP stationarity.

A.5 How Clipping Works in GEAR

Both clipping procedures use currently available valid rows. Write $\mathcal{P} = \{x \in B : Ax + c \le 0\}$ for this temporary relaxation, which may use inherited rather than freshly propagated rows. Its only required relation to the original problem is $\mathcal{F}(B) \subseteq \mathcal{P}$. The procedures are the relaxed and complete clipping mechanisms of Zhou et al. [73], applied to NLP constraint rows.

Relaxed clipping. Let $L_i = \min_{x \in B}(a_i^\top x + c_i)$. For $a_{i,j} > 0$, the minimum contribution of coordinate j to L_i is $a_{i,j}\underline{x}_j$. Holding that coordinate at a feasible x_j and minimizing all other coordinates implies $L_i + a_{i,j}(x_j - \underline{x}_j) \leq 0$. The analogous argument for $a_{i,j} < 0$ gives

$$x_j \leq \underline{x}_j - L_i/a_{i,j} \quad (a_{i,j} > 0), \qquad x_j \geq \overline{x}_j - L_i/a_{i,j} \quad (a_{i,j} < 0). \tag{25}$$

Intersect these bounds with the original box and across all rows. A single sweep costs $O(mn)$ per box and yields $\mathcal{F}(B) \subseteq \mathcal{P} \subseteq B' \subseteq B$. Zero coefficients yield no coordinate update. A positive L_i or crossed coordinate bounds certifies infeasibility. The procedure computes a containing box, not necessarily the tightest axis-aligned hull of the joint polyhedron. Parallel sweeps and repeated sequential updates are both sound when each uses the appropriate current intervals.

Complete clipping. For an intermediate lower row $\underline{z}(x) = a_z^\top x + c_z$, the constraint-aware target is

$$\min_{x \in \mathcal{P}} \underline{z}(x).$$

Its dual is

$$\max_{\beta \geq 0} \min_{x \in B} \big[a_z^\top x + c_z + \beta^\top (Ax + c)\big],$$

with multipliers distinct from the objective multipliers λ. Every evaluated dual value is a valid intermediate lower bound. For an upper row $\overline{z}$, applying the lower-bound construction to $-\overline{z}$ and negating the result gives a valid upper bound. The single-constraint procedure of Zhou et al. [73] evaluates the piecewise-linear dual by sorting breakpoints in $O(n \log n)$ time; multiple constraints are treated through coordinate ascent. Finite updates need not reach the joint LP optimum. Retaining the previous bound alongside new candidates prevents an unfinished update from weakening an intermediate interval.

Why the procedures differ. For $x_1, x_2 \in [0, 1]$ and $x_1 + x_2 \leq 1$, each coordinate can still attain 1, so no valid input hull can decrease either upper endpoint. For $z = x_1 + x_2$, however, the row yields $z \leq 1$ rather than the box bound 2. This tighter interval can strengthen every subsequent relaxation that uses z. The resulting new rows need only hold for feasible graph trajectories, as formalized in Appendix A.3.

Scheduling and infeasibility. Before a child's first propagation, its parent's rows and the original affine rows are available for clipping. Bounds produced later in the pass may be used only in subsequent computations or a repeated pass, not retroactively in their own derivation. Empty input or intermediate intervals certify absence of feasible trajectories. Empty intermediate intervals need not establish emptiness of a separately defined input-only polyhedron that omitted those intermediate conditions; the required conclusion is $\mathcal{F}(B) = \emptyset$.

A.6 Safe Linear Concretization over Clipped Relaxation Domains

Box concretization and polyhedral concretization have different roles. For an affine function $q(x) = a^\top x + c_0$,

$$b_B(q) = c_0 + \sum_j \min\{a_j \underline{x}_j, a_j \overline{x}_j\}$$

is the exact box minimum in real arithmetic. The stronger polyhedral target is the LP

$$\min_x \quad a^\top x + c_0 \qquad \text{s.t.} \quad \underline{x} \leq x \leq \overline{x}, \quad Ax + c \leq 0. \tag{26}$$

This LP explains the target of complete clipping and the objective dual. The validity of the batched dual construction requires no LP solve; LP solvers only propose multipliers (Appendix B.5).

Domain containment. If clipping replaces B by B', the invariant is

$$\mathcal{F}(B) \subseteq \mathcal{P} \subseteq B' \subseteq B, \qquad \mathcal{F}(B') = \mathcal{F}(B). \tag{27}$$

Later concretizations use the current box B'; inherited rows remain valid on its feasible set. The polyhedron is a relaxation target, whereas the inner minimization of $d_{B'}$ is still over the box. Minimizing $\underline{f}_0 + \lambda^\top (Ax + c)$ over $\mathcal{P}$ would impose the same rows twice: their weighted sum is already nonpositive everywhere in that domain, so its maximum over $\lambda \geq 0$ cannot improve on $\lambda = 0$. This is not the dual used in (9).

Equivalent coordinate formulas. For a row $\alpha^\top x \le b$, relaxed clipping can also be written

$$\overline{x}_j \leftarrow \min\left\{\overline{x}_j, \frac{b - \sum_{k\neq j} \min\{\alpha_k \underline{x}_k, \alpha_k \overline{x}_k\}}{\alpha_j}\right\} \qquad (\alpha_j > 0), \tag{28}$$

$$\underline{x}_j \leftarrow \max\left\{\underline{x}_j, \frac{b - \sum_{k\neq j} \min\{\alpha_k \underline{x}_k, \alpha_k \overline{x}_k\}}{\alpha_j}\right\} \qquad (\alpha_j < 0). \tag{29}$$

Both use the minimum activity of all other coordinates. The sign of the denominator determines the inequality direction. Early stopping leaves a larger containing box and is therefore conservative.

Quantity	**Target**	**Safe approximation**
Objective bound	$d_B(\lambda)$, evaluated over the current box	Return no larger value
Infeasibility certificate	$\min_{x\in B} \nu^\top (Ax + c)$	Return no larger value; prune only when positive
Intermediate lower bound	$\min_{x\in\mathcal{P}} \underline{z}(x)$	Any feasible dual value, evaluated downward
Retained intervals	Contain every feasible trajectory	Round outward
Incumbent	Feasible value of the original objective	Check feasibility and objective directly

Table 5: Directions of conservative evaluation. Feasibility shifts have already been absorbed into f_i; numerical margins are separate.

A.7 Operator Relaxations and Validity Domains

Operator	Change relative to the baseline	Mechanism (this work)
$x\cdot y$ (BOUNDMUL)	revised default (upstream fixed one McCormick vertex)	coefficient-aware $r\in\{0,1\}$ per element
x/y on $y_l>0$ (BOUNDDIV)	new opt-in fused mode (upstream decomposes $x\cdot(1/y)$)	8-plane candidate set with per-element coverage
$\log, \sqrt{\cdot}, 1/x, \exp$	revised tangent side (upstream midpoint-style tangent)	parallel-tangent (tangent slope = secant slope)
x^α, $\alpha\notin\mathbb{Z}_{\ge 2}$ (BOUNDPOW)	new branch	non-integer / $x \ge 0$ branch with secant-tangent
BOUNDSCATTERELEMENTS / BOUNDATENINDEX_ADD	new operators	tensor-mode bound_backward / bound_forward / interval_propagate
CLAMPEDMULTIPLICATION	revised	`where(A==0)` guards remove $0\cdot\infty$ NaN paths

Table 6: Operator changes relative to the baseline AUTO_LIRPA implementation. New and revised primitives retain a tensorized affine-bound interface.

This section spells out the per-operator relaxations summarized by Table 6 in Section 3.3. Linear bound propagation composes these local relaxations backward through the computation graph; the rule is local but the resulting linear bound is valid on the retained feasible trajectories, because every local replacement is valid on its recorded interval domain and selected according to the backward-coefficient sign.

Lemma A.4 (Composition of local relaxations)**.** *Let S be a set of inputs whose graph trajectories lie within all current intermediate intervals. Let every nonlinear graph node $v = h(u)$ be equipped, on its current interval domain U, with linear functions $\underline{h}$ and $\overline{h}$ satisfying $\underline{h}(u) \le h(u) \le \overline{h}(u)$ for all $u \in U$. If backward linear bound propagation for a lower bound replaces each occurrence of $\gamma h(u)$ by $\gamma\underline{h}(u)$ for $\gamma \ge 0$ and by $\gamma\overline{h}(u)$ for $\gamma < 0$, then the final expression at the input is an affine lower bound for the selected graph output on S. Ordinary propagation takes $S = B$; constraint-tightened propagation may take $S = \mathcal{F}(B)$.*

The proof is a reverse topological induction using Equation (15). The same argument applies componentwise to batched coefficients and to multi-input operators after flattening their local linear relaxations into the parent variables.

Linear and separable arithmetic. Affine nodes are exact. For $z = \alpha x + \beta y + c$, the backward coefficient on z is distributed as $\alpha\gamma$ to x, $\beta\gamma$ to y, and γc to the bias. No relaxation gap is introduced. Elementwise negation, scalar multiplication, addition, subtraction, indexing, reshaping, and broadcasting are handled in this exact category after canonicalization. This exact propagation matters because it keeps large linear subgraphs from paying relaxation cost before nonlinear operators are reached.

Univariate secant-tangent template. Let $z = h(t)$ on $t \in [\alpha, \beta]$ with $\alpha < \beta$, and define the secant

$$S^h_{\alpha,\beta}(t) = h(\alpha) + \frac{h(\beta) - h(\alpha)}{\beta - \alpha}(t - \alpha). \tag{30}$$

For any $t_0 \in [\alpha, \beta]$, define the tangent

$$T^h_{t_0}(t) = h(t_0) + h'(t_0)(t - t_0). \tag{31}$$

If h is convex on $[\alpha, \beta]$, then $T^h_{t_0}(t) \le h(t) \le S^h_{\alpha,\beta}(t)$. If h is concave, the inequalities reverse: $S^h_{\alpha,\beta}(t) \le h(t) \le T^h_{t_0}(t)$. The implementation chooses t_0 by the secant-slope rule

$$h'(t_0) = \frac{h(\beta) - h(\alpha)}{\beta - \alpha} \tag{32}$$

when the solution lies in the interval; otherwise it uses the endpoint tangent that gives the smaller valid bound gap. For $h \in C^2$ with a bounded second derivative on the interval, the one-dimensional gap is $O((\beta - \alpha)^2)$. This local estimate is distinct from the graph-wide accuracy assumption in Assumption A.6.

Logarithm. For $h(t) = \log t$, the domain guard is $0 < \alpha \le t \le \beta$. The function is concave, so the lower side is the secant and the upper side is a tangent. The secant-slope tangent point is the logarithmic mean

$$t_0 = \frac{\beta - \alpha}{\log \beta - \log \alpha}. \tag{33}$$

If the lower endpoint is at or below zero after outward interval tolerances, the node must be split, rewritten with a stricter domain guard, or rejected; clipping the value of $\log t$ without a corresponding domain proof would invalidate the linear bound.

Square root. For $h(t) = \sqrt{t}$, the guard is $0 \le \alpha \le t \le \beta$. The function is concave on the nonnegative axis, so the lower side is the secant and the upper side is a tangent. For $\alpha < \beta$, the secant-slope tangent point is

$$t_0 = \left(\frac{\sqrt{\alpha} + \sqrt{\beta}}{2}\right)^2. \tag{34}$$

When $\alpha = 0$, the tangent slope at the endpoint is singular; the implementation uses the interior point from Equation (34) for the upper side and keeps the secant for the lower side. If numerical interval propagation produces $\alpha < 0$ for a source expression that is mathematically nonnegative, the linear-bound engine cannot use that fact unless it is represented as a verified bound; otherwise the expression is split or rejected.

Reciprocal. For $h(t) = 1/t$, the interval must be sign-fixed and bounded away from zero. On $\alpha > 0$, h is convex; tangents are lower bounds and the secant is an upper bound. On $\beta < 0$, h is concave; the same secant-tangent construction is used with the lower/upper sides reversed. In both cases the slope-matching condition $h'(t_0) = (h(\beta) - h(\alpha))/(\beta - \alpha)$ has the sign-aware solution

$$t_0 = \operatorname{sign}(\alpha)\sqrt{\alpha\beta} \quad \Longleftrightarrow \quad t_0 = \begin{cases} +\sqrt{\alpha\beta}, & \alpha > 0, \\ -\sqrt{\alpha\beta}, & \beta < 0, \end{cases} \tag{35}$$

since α, β have the same sign under the sign-fixed guard. The naive form $t_0 = \sqrt{\alpha\beta}$ is positive on negative intervals and would lie outside $[\alpha, \beta]$, producing an invalid relaxation; the sign correction restores validity. Intervals crossing zero are never relaxed by a finite linear bound. The solver must split the denominator interval, rewrite the model, or mark the instance unsupported.

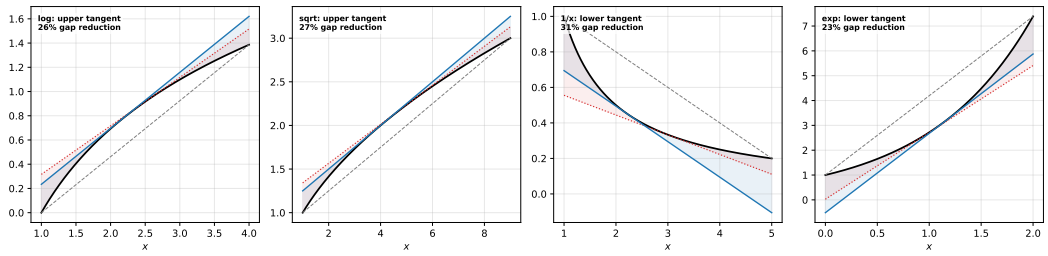

Figure 3: Tangent-side comparison for four convex/concave operators. Black is the operator, gray is the unchanged secant, red is the upstream midpoint-style tangent, and blue is the GEAR parallel-tangent. For concave $\log$ and $\sqrt{\cdot}$ the changed side is the upper tangent; for convex $1/x$ and $\exp$ it is the lower tangent.

Exponential. For $h(t) = \exp t$, the function is convex on every finite interval. The upper side is the secant and the lower side is a tangent. The secant-slope tangent point is

$$t_0 = \log \frac{e^\beta - e^\alpha}{\beta - \alpha}. \tag{36}$$

When the interval is extremely narrow, the implementation evaluates this expression with a stable expm1-style form and otherwise falls back to the midpoint tangent only if the slope-matching calculation is numerically indistinguishable from the midpoint at floating-point precision.

Power functions. For $h(t) = t^p$ with non-integer p, the base interval must be nonnegative and, when derivatives are singular at zero, guarded away from zero according to the implementation tolerance. For $0 < p < 1$, h is concave on $[0, \infty)$ and uses the same lower-secant / upper-tangent pattern as $\sqrt{t}$. For $p > 1$, h is convex on $[0, \infty)$ and uses lower tangents / upper secants. For negative powers, the implementation treats t^p as a reciprocal-power primitive on a positive sign-fixed interval and applies the corresponding convexity rule; if the base can approach zero, no finite linear bound is emitted. Integer powers with odd exponents can be represented on sign-changing intervals, but their curvature changes at zero for most exponents; the interval is split at curvature changes before applying the monotone-curvature template.

Selection among valid affine planes. For $z = h(y)$ on a box Y and backward coefficient γ, let $\mathcal{L}$ and $\mathcal{U}$ contain valid lower and upper planes. The rule used in the main text is

$$q^\star \in \begin{cases} \arg\max\limits_{q \in \mathcal{L}} \min\limits_{y \in Y} q(y), & \gamma \geq 0, \\ \arg\min\limits_{q \in \mathcal{U}} \max\limits_{y \in Y} q(y), & \gamma < 0. \end{cases} \tag{37}$$

This maximizes $\min_{y \in Y} \gamma q(y)$ among the eligible candidates. For $\gamma = 0$ all choices have zero contribution. Each score is a box concretization, so selection can be tensorized. The guarantee is local: other terms, shared intermediate variables, and coefficient cancellation can change which selection gives the best final composed bound.

Bilinear terms. For $z = xy$ on $X \times Y = [x_L, x_U] \times [y_L, y_U]$, the McCormick facets are

$$z \geq x_L y + y_L x - x_L y_L, \qquad z \geq x_U y + y_U x - x_U y_U, \tag{38}$$

$$z \leq x_U y + y_L x - x_U y_L, \qquad z \leq x_L y + y_U x - x_L y_U. \tag{39}$$

An LP relaxation would keep all four facets simultaneously. GEAR keeps one linear bound per output direction, so it must project these facets to one lower or one upper linear plane for each backward coefficient. For a positive parent coefficient γ, lower candidates in Equation (38) are scored by the scalar linear minimum of $\gamma p(x, y)$ on the current local rectangle or cut-tightened box. For a negative parent coefficient, upper candidates in Equation (39) are scored after multiplication by γ. The selected plane is sound because it is one of the McCormick facets, but the projection loses the simultaneous-facet closure; this is the precise reason LP-native solvers can dominate on bilinear-heavy root relaxations. The maximum McCormick envelope width over the rectangle is proportional to $(x_U - x_L)(y_U - y_L)$, so spatial refinement closes this local gap quadratically in the box diameter when both factors shrink.

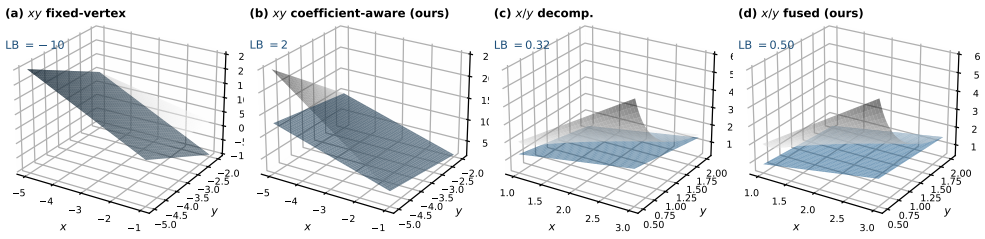

Figure 4: Linear lower bounds for two bivariate operators. On $[-5,-1]\times[-5,-2]$, the fixed product plane gives -10 and the selected McCormick plane gives 2. On $[1,3]\times[0.5,2]$, the reported reciprocal-product decomposition gives 0.32 and the fused quotient plane gives 0.5. The selected bounds equal the true minima on these illustrated rectangles.

Fused division. For $z = x/y$, the denominator interval must satisfy either $y_L \geq \delta_{\text{div}} > 0$ or $y_U \leq -\delta_{\text{div}} < 0$. The fused implementation constructs candidate planes $p(x,y) = \alpha x + \beta y + \gamma$ directly for the quotient on the guarded rectangle. On the positive-denominator branch used in Figure 4, the candidate pool has eight planes: four corner tangents

$$P_T^{(x^\star,y^\star)}(x,y) = \frac{x}{y^\star} - \frac{x^\star y}{(y^\star)^2} + \frac{x^\star}{y^\star}, \qquad (x^\star,y^\star) \in \{x_L,x_U\}\times\{y_L,y_U\}, \tag{40}$$

and four three-corner secant planes obtained by interpolating x/y at any three corners of $[x_L,x_U]\times[y_L,y_U]$. The negative-denominator branch is handled by the same guarded-quotient construction with the inequality orientation reversed by the denominator sign. Because x/y is not globally convex or concave on an arbitrary guarded rectangle, candidate generation alone is not trusted: every candidate is accepted only after a residual validity check. For $y > 0$, a lower plane must satisfy

$$p(x,y) \leq x/y \quad \Longleftrightarrow \quad R_p(x,y) := y\,p(x,y) - x \leq 0 \quad \forall (x,y) \in X \times Y, \tag{41}$$

and an upper plane must satisfy $R_p(x,y) \geq 0$. For $y < 0$, the inequalities reverse because multiplication by y changes sign. The residual R_p is a quadratic-bilinear expression; the implementation checks the required sign with interval bounds, McCormick bounds on the xy term, and exact endpoint checks for the y^2 term. If the residual check is inconclusive, the plane is rejected and the engine falls back to a reciprocal-plus-bilinear relaxation, interval splitting, or unsupported-instance reporting. This accept-after-check rule is the soundness guard for fused division. After invalid candidates are filtered out, the surviving lower and upper planes are scored by the same box-extremum rule as Equation (37).

Trigonometric functions. For $\sin t$ and $\cos t$, global secant-tangent relaxations over long intervals are too loose and may cross curvature changes. The interval is first localized by splitting at points where the second derivative changes sign:

$$\sin t:\ t = k\pi, \qquad \cos t:\ t = \pi/2 + k\pi, \qquad k \in \mathbb{Z}.$$

On each resulting monotone-curvature segment, Equations (30) and (31) applies. If an interval spans multiple periods and splitting would exceed the configured segment budget, a valid coarse interval bound or a documented fallback is required; an unsupported primitive cannot generate a pruning certificate. A local segment bound may be forwarded only when the retained feasible trajectories lie within its validity domain.

Max, min, and residual outputs. Nonlinear NLP constraints are represented as separate graph outputs and affine constraints as exact rows, not through a hard max in the objective. For residual infeasibility, GEAR avoids needing a tight linear bound for the nonsmooth max by reusing the fact that every constraint lower bound is also a lower bound for $r(x) = \max_i f_i(x)$. If a source model contains explicit $\max$ or $\min$ primitives, they must be rewritten into inequalities, split into supported piecewise-smooth regions, or handled by a separate valid relaxation rule before they can participate in pruning.

Validity-domain bookkeeping. Every local relaxation records the interval on which it is valid. The output row is valid on the set of inputs whose trajectories remain within these intervals; after

Algorithm 1 ALM-PGD on the current box B

Require: Current box $B = [\underline{x}, \overline{x}]$; original objective f_0 and tolerance-shifted constraints $f = (f_1, \ldots, f_m)$; seed $x \in B$; $\mu \geq 0$, $\rho > 0$; outer and inner budgets $K_{\text{out}}, K_{\text{in}}$; step schedule $\{\eta_t\}$.

$\bar{x} \leftarrow \varnothing$
for $k = 1, \ldots, K_{\text{out}}$ **do**
  for $t = 1, \ldots, K_{\text{in}}$ **do**
    $g \leftarrow \nabla_x \phi_\rho(x; \mu)$, with ϕ_ρ defined in (11)
    $x \leftarrow \Pi_B(\text{AdamStep}(x, g, \eta_t))$
  end for
  Evaluate $f_0(x)$ and all original shifted constraints at x
  if $f_i(x) \leq 0$ for every i and ($\bar{x} = \varnothing$ or $f_0(x) < f_0(\bar{x})$) **then**
    $\bar{x} \leftarrow x$
  end if
  $\mu \leftarrow [\mu + \rho f(x)]_+$
  Increase ρ if the maximum violation failed to decrease by the prescribed ratio
end for

Ensure: Best accepted candidate $\bar{x}$ among the tested iterates, or $\varnothing$.

constraint-driven clipping, validity on $\mathcal{F}(B)$ is sufficient. When a row is forwarded, each child's feasible set is a subset of the parent's, so the validity implication is preserved. Fresh rows replace the inherited nonlinear rows after the child's pass. No clipping or operator-domain guard may exclude an original feasible trajectory without a valid certificate; otherwise the affected relaxation must be recomputed or not used for pruning.

Primitive	Required domain	Relaxation rule	Failure action
linear arithmetic	finite input intervals	exact coefficient propagation	none
xy	finite rectangle	one coefficient-aware McCormick facet	keep loose facet or split
x/y	denominator sign-fixed and guarded	accept-after-check fused quotient plane	reciprocal rewrite, split, or reject
$1/t$	sign-fixed and guarded	secant-tangent with side by curvature	split or reject near zero
$\log t$	$t > 0$ with guard	concave lower secant / upper tangent	split, strengthen guard, or reject
$\sqrt{t}$	$t \geq 0$ with derivative guard if needed	concave lower secant / upper tangent	split or reject if nonnegativity is not established
t^p	domain compatible with exponent	curvature-local secant-tangent	split at sign/curvature changes or reject
$\sin t, \cos t$	localized monotone-curvature segment	segmentwise secant-tangent	split by period/curvature or reject

Table 7: Operator validity conditions for the linear-bound engine. Outside these domains, GEAR must split, rewrite, fall back to a looser valid rule, or reject the primitive; it must not emit an unchecked linear bound.

A.8 Primal Recovery Details

The primal heuristic evaluates the original graph and the original affine constraints, not only their relaxation. It applies projected first-order steps to the Hestenes-Powell-Rockafellar function in (11), followed by multiplier and penalty updates. These operations seek a useful incumbent; successful solution of the local NLP is not a prerequisite for the lower-bound certificates.

Multiplier update and gradient. The outer update is

$$\mu \leftarrow [\mu + \rho f(x)]_+. \tag{42}$$

For differentiable original functions, the inner gradient is

$$\nabla_x \phi_\rho(x;\mu) = \nabla f_0(x) + \rho \sum_{i=1}^{m} [f_i(x) + \mu_i/\rho]_+ \nabla f_i(x). \tag{43}$$

The squared positive-part penalty is continuously differentiable in its scalar argument; its derivative is zero at the hinge. Any nonsmoothness of the original graph still requires the implementation's derivative convention. Affine constraints contribute their exact gradients a_i. The objective remains in every inner step, so the method does not switch to a feasibility-only objective upon reaching the constraint set.

Projected Adam inner loop. For an inner step $t \geq 1$ with $g_t = \nabla_x \phi_\rho(x_t;\mu)$, a standard projected-Adam update is

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{44}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t \odot g_t, \tag{45}$$

$$\widehat{m}_t = m_t/(1-\beta_1^t), \qquad \widehat{v}_t = v_t/(1-\beta_2^t), \tag{46}$$

$$\widetilde{x}_{t+1} = x_t - \eta_t \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon_{\text{adam}}}, \tag{47}$$

$$x_{t+1} = \Pi_B(\widetilde{x}_{t+1}). \tag{48}$$

All vector operations and box projections batch across starts and boxes. The inner solver uses `AdamClipping_NLP`; its learning-rate normalization and handling of clipped coordinates are implementation details beyond the certificate proof. Adam's second-moment scaling is applied before projection. A smooth box parameterization is an alternative implementation path, not assumed by the projected update above.

Dual-guided seeds. The distinguished seed is

$$x^{\text{best}} \in \arg\min_{x \in B} \big[\underline{f}_0(x) + (\lambda^{\text{best}})^\top (Ax + c) \big].$$

This is the box inner minimizer of the evaluated dual, not an optimizer over $\mathcal{P}(B)$ and not necessarily an LP-feasible point. Additional starts may use the midpoint, a projected incumbent, Sobol points, and minimizers of individual constraint rows. Their use and budget affect search success, not whether an accepted point is feasible.

Multiplier initialization and penalty schedule. A nonnegative rescaling can transfer constraint emphasis from the LP dual to the ALM multipliers, for example

$$\mu_i^{(0)} = \kappa \lambda_i^{\text{best}}, \qquad \kappa = \frac{1 + \|\nabla f_0(x^{\text{best}})\|_\infty}{1 + \|\sum_i \lambda_i^{\text{best}} \nabla f_i(x^{\text{best}})\|_\infty}.$$

Zero initialization is an alternative. The rescaling is a warm start, not a statement that LP and original-NLP multipliers coincide. The penalty is increased when violation reduction is insufficient. Neither this schedule nor finite projected-Adam steps establish a guarantee of feasible recovery on every nonconvex instance.

Use of linear feasibility cuts. Retained rows can additionally guide the primal search through

$$\phi^{\text{cut}}_{\rho,\sigma}(x;\mu) = \phi_\rho(x;\mu) + \frac{\sigma}{2} \|[Ax + c]_+\|_2^2, \qquad \sigma \geq 0. \tag{49}$$

Every original feasible point satisfies the rows, so the extra term vanishes on the feasible set. It can nevertheless change trajectories away from feasibility. This is the "ALM with linear constraints" variant; no extra penalty is used as a substitute for checking the original nonlinear constraints.

Acceptance gate. A candidate improves the incumbent only when

$$\bar{x} \in B_0, \qquad f_i(\bar{x}) \leq 0 \ (i = 1, \ldots, m), \qquad f_0(\bar{x}) < \text{UB}. \tag{50}$$

All functions here use the same tolerance shifts as in the lower-bound problem. Objective and feasibility are evaluated afresh; neither a low ALM value nor satisfaction of the linear rows is sufficient. For numerical certification the original-model checks and conservative evaluation requirements are those of Appendices A.3 and C.4.

A.9 Branching Calibration

λ-SB branches on the coordinate that maximizes the λ-weighted score S_j of Equation (12), a λ-weighted form of smart branching. Its fixed-row interpretation explains the coefficient magnitudes without asserting that the score predicts the nonlinear bounds reconstructed on children.

Fixed-row midpoint calculation. For $q(x) = a^\top x + c_0$ on B, a minimizing coordinate is $e_j = \underline{x}_j$ when $a_j \geq 0$ and $e_j = \overline{x}_j$ otherwise. A midpoint split leaves this coordinate's contribution unchanged in one child and increases its minimum by $|a_j|w_j/2$ in the other, where $w_j = \overline{x}_j - \underline{x}_j$. For a uniformly chosen child of this midpoint split,

$$\mathbb{E}[\Delta_j] = \frac{|a_j|w_j}{4}, \qquad \Delta_j \in \{0, |a_j|w_j/2\}. \tag{51}$$

The minimum across both children is unchanged for this fixed affine row; refinement is useful because later propagation rebuilds the nonlinear relaxations, not because splitting a fixed affine function necessarily raises the global lower bound.

Lemma A.5 (One-step linear branching proxy)**.** *For the mixed row with coefficient $a_j(\lambda) = a_{0,j} + \sum_i \lambda_i a_{i,j}$, the average fixed-row midpoint improvement is bounded by $S_j/4$, since*

$$|a_j(\lambda)| \leq |a_{0,j}| + \sum_i \lambda_i |a_{i,j}|.$$

Here S_j is evaluated using the same nonnegative multipliers as the mixed row.

Proof. Apply the triangle inequality and then (51). □

The implemented score avoids cancellation between outputs. This is a proxy for where refinement may matter; it is not strong branching or an exact estimate of the children's recomputed LP values.

Eligible coordinates and shrinking. For $\tau \in (0, 1]$, choose

$$j^\star \in \arg \max_{j:\, w_j \geq \tau \max_k w_k} S_j. \tag{52}$$

If a positive maximum width remains, midpoint-split the selected coordinate. Along any infinite branch, suppose the maximum width did not tend to zero. Then infinitely many selected widths would exceed a fixed positive number. At least one of finitely many coordinates would be selected infinitely often at such widths, contradicting its repeated halving. Clipping only reduces widths further. Thus every infinite branch shrinks to a point. This argument concerns shrinking, not feasible recovery or finite termination on its own.

A.10 Overall Algorithm of GEAR

Algorithm 2 expands the main-text description with explicit initialization and global-gap bookkeeping. Node operations are batched even though the notation describes their per-box effect. The queue policy is an implementation choice; the termination theorem additionally requires that each batch include a minimum-bound box. No queue policy can invalidate a certificate if all unresolved boxes remain represented.

Algorithm 2 GEAR: batched spatial branch-and-bound

Require: Computation graph F, initial finite box B_0, gap tolerance $\varepsilon > 0$, batch size K, and time limit.

Initialize $\mathrm{UB} \leftarrow +\infty$, $\bar{x} \leftarrow \varnothing$, and $\mathcal{Q} \leftarrow \{B_0\}$ with $\mathrm{LB}(B_0) = -\infty$ and the original affine rows available for clipping.
while $\mathcal{Q} \neq \emptyset$ and time remains **do**
  $L_{\mathrm{glob}} \leftarrow \min\{\mathrm{UB}, \min_{B \in \mathcal{Q}} \mathrm{LB}(B)\}$
  if $\mathrm{UB} < +\infty$ and $\mathrm{UB} - L_{\mathrm{glob}} \leq \varepsilon$ **then**
    break
  end if
  Pop up to K boxes; they stay unresolved until their processing finishes.
  Apply relaxed clipping with available rows; discard certified infeasible boxes.
  Propagate nonlinear output bounds with complete clipping using already available rows; refresh the nonlinear constraint rows.
  Prune boxes with empty intermediate intervals or a positive conservative certificate from (10).
  Evaluate fixed-budget dual updates and set each surviving bound to the maximum of its inherited value and (22).
  Prune boxes with $\mathrm{LB}(B) \geq \mathrm{UB}$ when an incumbent exists.
  Run scheduled ALM-PGD searches using the dual warm starts; update $(\mathrm{UB}, \bar{x})$ only by (50).
  Recheck dominance under the updated incumbent.
  Midpoint-split survivors using (52); push children with inherited lower bounds and fresh nonlinear rows.
end while
$L_{\mathrm{glob}} \leftarrow \min\{\mathrm{UB}, \min_{B \in \mathcal{Q}} \mathrm{LB}(B)\}$, with $\min \emptyset = +\infty$.

Ensure: Incumbent $(\mathrm{UB}, \bar{x})$ if found, global lower bound L_{glob}, and status: gap certified, infeasible, or limit reached.

At a time-limit interruption within a batch, unfinished boxes retain valid bounds and are included in the final active set. An unsupported operator, an inconclusive numerical test, or an unsplittable unresolved box is not an infeasibility certificate: it must remain unresolved or cause an explicitly uncertified termination. In particular, no loop condition evaluates $+\infty - (+\infty)$ before checking that an incumbent exists.

A.11 Node Lifecycle and Data Dependencies

A node stores its current box, a scalar inherited lower bound, and the currently available nonlinear constraint rows. Original affine rows are shared. The following order makes the data dependencies explicit; iteration counts and the primal-call schedule remain implementation parameters.

Step 1: assemble available rows. At the root use the original affine constraints; at descendants combine them with the parent's nonlinear rows. The latter need only be valid on the parent's feasible set, which contains the child's feasible set.

Step 2: clip the input box. Apply relaxed clipping to the available rows. Retain the resulting box and discard the node only if a conservative contradiction is obtained. An inability to certify emptiness is not evidence of feasibility.

Step 3: propagate and tighten intermediate bounds. Process the nonlinear graph over the clipped box, using available rows for complete clipping. Replace inherited nonlinear rows only after the new pass has produced valid rows. A further propagation sweep may use those refreshed rows; a row is never used before it exists.

Step 4: test infeasibility. Test individual rows or selected nonnegative combinations using (10), and check empty intermediate intervals. These certificates do not require a finite incumbent.

Step 5: update the objective lower bound. Optimize λ over the cached coefficients. Set the node bound to the maximum of its inherited bound and the best conservatively evaluated new dual value.

Prune if it is at least the finite incumbent UB. The exact-dominance convention is used throughout this algorithm; stopping at a finite global gap is a separate test.

Step 6: search for a primal incumbent. Run the prescribed primal searches on selected surviving boxes. Accept only candidates satisfying (50). The linear rows and dual outputs are heuristic guidance, not replacements for the original constraints.

Step 7: recheck dominance and prepare the children. A newly improved incumbent can prune other surviving boxes. For those remaining, retain the fresh nonlinear rows, original affine rows, and best scalar lower bound. Row storage does not grow with tree depth.

Step 8: split and return children. Use (52) and midpoint splitting. Each child inherits the parent's lower bound and current nonlinear rows. The children cover the clipped parent, which still contains all of its feasible points.

Action	Effect	Validity dependency
Clipping contradiction	Infeasibility pruning	Feasible trajectories are preserved
Positive row mixture	Infeasibility pruning	Nonnegative weights and valid constraint rows
Objective comparison	Dominance pruning	Valid lower bound and feasible incumbent
Primal recovery	Incumbent update	Original-model feasibility gate
Row inheritance	Reuse before propagation	Feasible-set containment
Spatial split	Refine active coverage	Child boxes cover the surviving parent

Table 8: Node operations and their certificate dependencies.

A.12 Proofs of the Sound Node Invariants

All local pruning rules follow from feasible-set preservation and valid one-sided bounds. The claims hold even if primal search fails or dual optimization terminates early.

Constraint validity and clipping. For $x \in \mathcal{F}(B)$, $\underline{f}_i(x) \le f_i(x) \le 0$, so $\mathcal{F}(B) \subseteq \mathcal{P}(B)$. Coordinate clipping follows from necessary conditions of these inequalities. An intermediate dual lower bound is at most $\underline{z}(x) \le z(x)$ at every feasible point, and an upper bound follows symmetrically. Thus interval tightening retains all feasible trajectories. Reapplying backward linear bound propagation to those intervals preserves output-row validity on $\mathcal{F}(B)$.

Objective lower bound. For every feasible x and every $\lambda \ge 0$,

$$d_B(\lambda) \le \underline{f}_0(x) + \lambda^\top (Ax + c) \le f_0(x) + \sum_i \lambda_i f_i(x) \le f_0(x). \tag{53}$$

Taking an infimum over feasible points gives $d_B(\lambda) \le p^\star(B)$. The maximum of any collection of valid lower bounds is valid, including bounds inherited from a parent. A multiplier cap changes the values reachable by the optimizer but does not invalidate any nonnegative iterate.

Infeasibility. If $\nu \ge 0$, every feasible point has $\nu^\top (Ax + c) \le 0$. A strictly positive box minimum therefore contradicts feasibility. The normalized residual interpretation in Appendix A.2 gives the same conclusion. Failure to find such a vector in a finite search establishes neither feasibility nor the absence of a stronger certificate.

Incumbent validity and active coverage. The acceptance gate ensures that a finite incumbent is feasible for the same tolerance-shifted problem used by the bounds. At an iteration boundary, let $\mathcal{Q}$ contain every unresolved box. Then

$$\begin{gathered} \mathrm{LB}(B) \le p^\star(B) \quad (B \in \mathcal{Q}), \\ \mathrm{UB} = f_0(\bar{x}) \text{ for a checked feasible } \bar{x}, \quad \text{or} \quad \mathrm{UB} = +\infty, \\ \{x \in \mathcal{F}(B_0) : f_0(x) < \mathrm{UB}\} \subseteq \bigcup_{B \in \mathcal{Q}} B. \end{gathered} \tag{54}$$

Clipping preserves the third relation; infeasibility pruning removes no feasible point; exact dominance removes only points unable to improve the incumbent; and children cover the surviving parent. Decreasing UB only shrinks the set on the left. Consequently

$$L_{\text{glob}} = \min\{\text{UB}, \min_{B \in \mathcal{Q}} \text{LB}(B)\} \leq p^\star.$$

If $\mathcal{Q}$ is empty, a finite incumbent is optimal, and no incumbent means the original problem is infeasible. A finite gap at most ε certifies ε-optimality. Mid-batch reporting must include boxes being processed as well as boxes still queued.

Tolerance-based pruning is a different convention. Pruning at $\text{LB}(B) \geq \text{UB} - \varepsilon$ also supports an ε-optimal result, but does not preserve the exact coverage invariant in (54). An algorithm using that convention must retain lower-bound certificates for discarded boxes, or report the appropriate tolerance certificate rather than the minimum over the remaining queue alone. Algorithm 2 uses exact dominance and a separate global-gap stopping rule.

A.13 Conditional Finite Termination

Finite termination needs assumptions beyond the sound invariants in Appendix A.12. The statement below is conditional on local accuracy, successful primal recovery, detected infeasibility, and an explicit queue policy. It is not a completeness proof for a fixed-budget ALM implementation. We use $\text{diam}(B) = \max_j w_j$, the infinity-norm diameter, throughout this subsection.

Assumption A.6 (Relaxation modulus and value oscillation)**.** On the finite, guarded domains under consideration, suppose the original functions are continuous and there are $C_{\text{rel}} > 0$, $q > 0$, and $L_f > 0$ such that the current affine rows obey

$$\sup_{x \in B} \{f_i(x) - \underline{f}_i(x)\} \leq C_{\text{rel}} \, \text{diam}(B)^q, \qquad i = 0, \ldots, m, \tag{55}$$

$$\sup_{x,y \in B} |f_0(x) - f_0(y)| \leq L_f \, \text{diam}(B). \tag{56}$$

Row validity on $\mathcal{F}(B)$ is separately maintained as in the main text. The error inequality above is an additional whole-box accuracy assumption, not a consequence of feasible-only validity.

Local relaxations versus a graph-wide modulus. For a C^2 univariate operator with bounded second derivative, secant-tangent error is quadratic in the local interval width. Bilinear error scales with the product of the two factor widths. For $\sqrt{t}$ on $[0, \beta]$, the parallel secant-tangent gap is $\sqrt{\beta}/4$. These are local statements. A uniform graph-wide exponent and constant also depend on composition, intermediate interval behavior, and domain guards; the local formulas alone do not prove a universal $q = 2$ for every supported graph. Assumption A.6 retains this as an explicit assumption.

Assumption A.7 (Sound oracle and primal recovery)**.** Suppose the following properties hold for the evaluated node procedure.

1. *Objective backstop.* For a function $e_{\text{clip}}(s) \to 0$ as $s \to 0$, with $e_{\text{clip}}(0) = 0$, its stored bound satisfies $\text{LB}(B) \geq d_B(0) - e_{\text{clip}}(\text{diam}(B))$.

2. *Local primal recovery.* For every $\varepsilon_P > 0$, there is $\delta_P(\varepsilon_P) > 0$ such that processing a box containing a feasible point with diameter at most δ_P includes a primal search returning a feasible $\bar{x}_B$ with $f_0(\bar{x}_B) \leq p^\star(B) + \varepsilon_P$.

3. *Small-scale infeasibility detection.* There is $\delta_I > 0$ such that every infeasible box with diameter at most δ_I is certified by the performed clipping or mixture tests. For a mixture certificate with exact value r_B, it suffices that $r_B > \tau_{\text{num}} + e_r(\text{diam}(B))$.

4. *Residual accuracy.* The corresponding computed certificate satisfies $r_B - e_r(\text{diam}(B)) \leq \widehat{r}_B \leq r_B$, with $e_r(s) \to 0$ and $e_r(0) = 0$, and pruning uses $\widehat{r}_B > \tau_{\text{num}}$. Here $\tau_{\text{num}} \geq 0$ is a numerical safety margin, not the feasibility tolerance already absorbed into the functions.

These assumptions concern actual evaluated bounds and successful tests, not merely the existence of an LP optimum or an infeasibility vector. In particular, local primal recovery is not established for ALM-PGD by this paper. Small-scale infeasibility detection is likewise stronger than feasible-set preservation. The assumptions make the distinction between soundness and conditional completeness explicit, in the spirit of spatial global-optimization analysis [55].

Theorem A.8 (Conditional ε-global convergence)**.** *Assume Assumptions A.6 and A.7, midpoint splitting with (52), and preservation of inherited lower bounds. Suppose each batch includes a box with the smallest stored lower bound in the queue. In the absence of a time limit, Algorithm 2 terminates with an ε-optimal incumbent when the problem is feasible, or certifies infeasibility otherwise. Choose $0 < \varepsilon_P < \varepsilon$ and $\delta > 0$ such that $\delta \le \min\{\delta_P(\varepsilon_P), \delta_I\}$ and*

$$\sup_{0 \le s \le \delta} \{C_{\mathrm{rel}} s^q + L_f s + e_{\mathrm{clip}}(s)\} + \varepsilon_P \le \varepsilon.$$

For initial widths w_j^0, define $M_j = \max\{0, \lceil \log_2(w_j^0/(\tau\delta)) \rceil\}$ when $w_j^0 > 0$, and $M_j = 0$ otherwise, with $M = \sum_j M_j$. The number of batch iterations before certification is bounded by

$$N_{\mathrm{iter}} \le 2^{M+1} - 1, \tag{57}$$

so a batch size at most K processes at most $K(2^{M+1} - 1)$ boxes.

Proof. For a feasible box, let x_0 minimize its objective row over B and let x_F be a feasible minimizer. The two accuracy assumptions give

$$d_B(0) = \underline{f}_0(x_0) \ge f_0(x_0) - C_{\mathrm{rel}} \operatorname{diam}(B)^q \ge p^\star(B) - L_f \operatorname{diam}(B) - C_{\mathrm{rel}} \operatorname{diam}(B)^q.$$

Hence processing a feasible box of diameter at most δ supplies both a lower bound and an incumbent with $\mathrm{UB} - \mathrm{LB}(B) \le \varepsilon$. An infeasible box at that scale is detected by assumption.

Call a queued box *unresolved at tolerance* when no finite incumbent exists or $\mathrm{LB}(B) < \mathrm{UB} - \varepsilon$. Once a box is resolved at tolerance, all its descendants remain so: they inherit its lower bound, and the incumbent can only decrease. Before the global stopping test succeeds, the best-bound queue policy therefore processes at least one previously unprocessed unresolved box in each batch. An unresolved path cannot continue through a processed box of diameter at most δ.

Every split on such a path has maximum width greater than δ, so its selected width exceeds $\tau\delta$. After t previous splits of coordinate j, its width is at most $w_j^0/2^t$; clipping can only reduce it. Thus coordinate j is split at most M_j times on an unresolved path. The unresolved binary tree has depth at most M and at most $2^{M+1} - 1$ nodes. Each batch processes at least one of these nodes, proving the iteration bound. Additional resolved boxes processed in the same batch cannot delay termination beyond this bound. When no unresolved box remains, the global gap test succeeds or the queue is empty. The reported certificate follows from the sound invariants. □

B Complete Description of the Solver

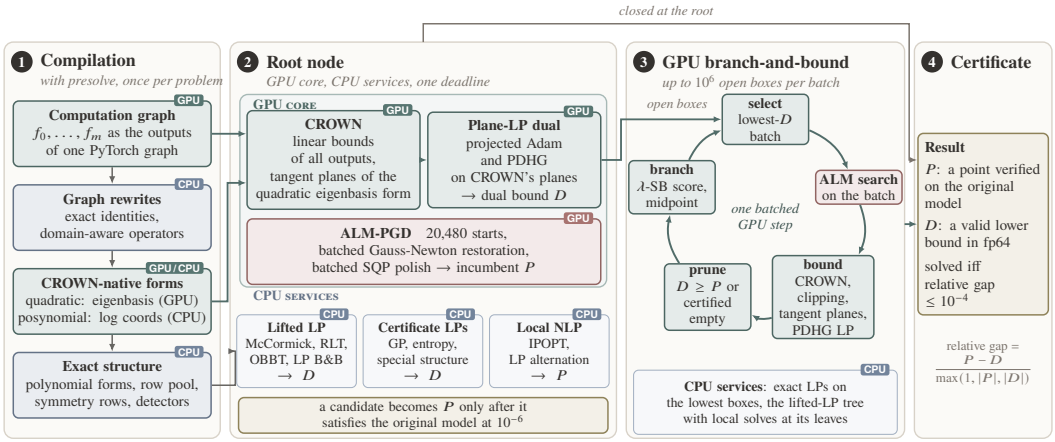

Figure 5: **All components of the solver evaluated in Section 4.** (1) Compilation and presolve: the model becomes one computation graph whose outputs are the objective and all constraints, and recognized structure is rewritten into forms whose tangent planes CROWN computes itself. (2) At the root, the GPU core computes CROWN bounds, the plane-LP dual bound D and the batched ALM search for an incumbent P; CPU services add bounds and points. A candidate becomes P only after it satisfies the original model at 10^{-6}, and the run stops once the relative gap of Equation (13) is at most 10^{-4}. (3) Open boxes enter the batched GPU branch-and-bound loop. (4) The result is certified by the gap rule of Section 4. The tag on each box shows whether it runs on the GPU or the CPU.

This section lists every technique used by the solver evaluated in Section 4, in the order in which a run meets them (Figure 5). The GPU core of Section 3 is summarized only where it interacts with the other components. Every technique either adds a valid bound, removes a region that provably contains no better feasible point, or proposes a point that is accepted only after the feasibility check of Equation (50); the soundness argument of Appendix A.12 therefore covers all of them, for valid relaxations and one consistent tolerance-relaxed feasible set. All statements below refer to that set, to the stated box and cutoff scope, and to the stated numerical enclosures: optimization routines only propose points or multipliers, and algebraic identities with independently evaluated enclosures decide validity.

B.1 Compilation and Presolve

Computation graph. The AMPL model is translated into one PyTorch graph whose outputs are the objective and all constraints (Appendix C); linear rows keep their exact coefficients and are also appended as outputs, so that every bound below sees every row. Tracing and the rewrites of this subsection are Python passes on the CPU, run once per graph; the graph is then evaluated and bounded on the GPU, batched over boxes.

Exact graph rewrites. Before bounding, the graph is normalized by identities that hold exactly, so that CROWN's local relaxations see the structure (Figure 6). Identical gathers and common subexpressions are merged. A product $u \cdot u$, $u \in [\ell, v]$, becomes $\mathrm{Sqr}(u)$, whose tangent at $\mathrm{proj}_{[\ell,v]}(0)$ attains the exact minimum; the two McCormick lower planes of $u \cdot v$ restricted to $v = u$ are the tangents at ℓ and v (the upper planes restrict to the secant), and CROWN keeps only one lower plane. A product $u \cdot v$ with $v = cu$ proved elementwise in rational polynomial arithmetic becomes $c\,\mathrm{Sqr}(u)$, and a nearly proportional pair is split as $uv = cv^2 + v(u - cv)$ with the residual kept exactly. For $c \neq 0$, a quadratic $cz^2 + kz$ in an input or shared node z becomes $c(z + s)^2 - cs^2 + rz$ with $s = \mathrm{fl}(k/2c)$ and the exact rational remainder $r = k - 2cs$, which centres the square near the quadratic's vertex (exactly at the vertex when $r = 0$). A variable exponent becomes $x^y = \exp(y \log x)$, exact for $x > 0$.

Domains and poles. These guards act elementwise in every bounding pass on the GPU. An operator with a restricted domain ($\log$, $\sqrt{\cdot}$, non-integer powers) intersects its input interval with the domain,

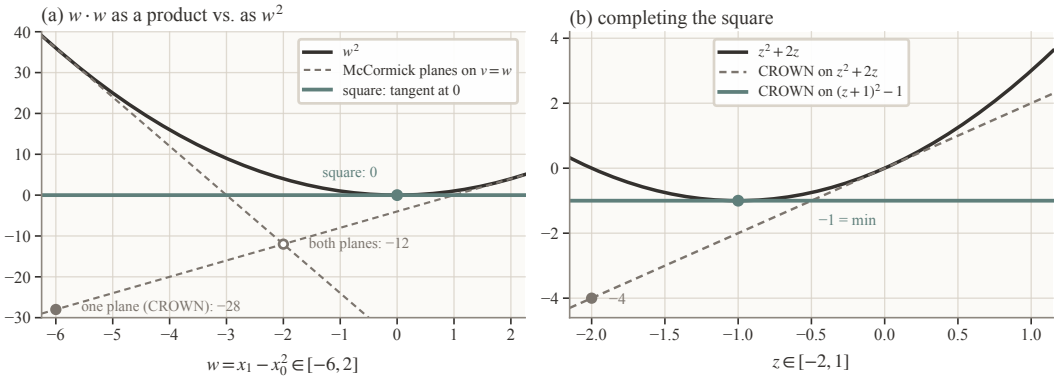

Figure 6: Exact rewrites that tighten CROWN. (a) $w = x_1 - x_0^2 \in [-6, 2]$ on $[-2, 2]^2$: restricted to $v = w$, the McCormick lower planes of $v \cdot w$ are the endpoint tangents of w^2; CROWN keeps the one with the larger box minimum and certifies -28 (-2800 for the Rosenbrock term $100w^2$), both planes would give -12, and the square's tangent at $\mathrm{proj}(0)$ gives the exact minimum 0. (b) On $[-2, 1]$, CROWN bounds $z^2 + 2z$ by $2z$, i.e. -4 (as interval arithmetic); after completing the square, the tangent of $(z + 1)^2$ at 0 gives the exact minimum -1.

$[\ell, \upsilon] \leftarrow [\max(\ell, 0), \upsilon]$, since no point outside it belongs to the model; an endpoint at 0 stands for an excluded limiting boundary (for $\log$ and negative powers the domain is $x > 0$), poles and undefined sides are handled separately, and the resulting bounds apply only where the original graph is defined. A power x^p with constant $p < 0$ is convex and decreasing on $(0, \infty)$: it is bounded by a tangent below and the secant above, and on $[0, \upsilon]$ by the tangent alone (at a point of $[\upsilon/2, \upsilon]$), because the secant would pass through the pole. Where an element e has no relaxation on one side (a pole or an infinite endpoint in its interval), each backward coefficient Λ_{re} that selects that side is replaced by the element's interval contribution, $\Lambda_{re} z_e \geq \min(\Lambda_{re} \underline{z}_e, \Lambda_{re} \overline{z}_e)$; only rows whose contribution is not finite get the always valid bound $-\infty$, and all other rows propagate unchanged. A reciprocal across a pole, $0 \in (\ell, \upsilon)$, keeps the excluded gap $1/x \notin (1/\ell, 1/\upsilon)$ through shifts and scalings, so that, e.g., $(1/x)^2 \geq \min(\ell^{-2}, \upsilon^{-2}) > 0$. Every intermediate interval is the intersection of its CROWN and interval bounds.

Exact structure and the row pool. A symbolic walk on the CPU attempts to compute, for every supported node, an exact sparse polynomial with rational coefficients in the inputs and in transcendental atoms ($\exp$, $\log$, powers, $\sin$, $\cos$), with a budget counted in term operations rather than time; nodes with unsupported operators, exhausted budget or errors get an unknown form and are excluded from the exact structural proofs. It feeds the certificates, the lifted LP, the exact sparse derivatives used by IPOPT and the pattern propagation of Appendix B.5. All rows enter one pool as $\ell_k \leq q_k(x) \leq h_k$, with q_k the non-constant part and the feasibility tolerance τ added once; rows $q_j(x) \leq \beta_j$ with $q_j = s_j q$ for one direction q merge exactly into $\max_{s_j < 0} \beta_j / s_j \leq q \leq \min_{s_j > 0} \beta_j / s_j$ (rounded outward once), and rows implied by the root box are masked. Two global facts are proved on the exact forms. A constraint row that equals $\alpha f_0 + \beta$ with $\alpha < 0$ and threshold t gives the floor $f_0 \geq (t + \tau - \beta)/\alpha$, applied to every box. A row $a\, h(x)^2 + \beta \leq t$ with $a > 0$ gives $|h| \leq r = \sqrt{(t + \tau - \beta)/a}$ (rounded up), and the CROWN lower planes $p_\pm(x) \leq \pm h(x)$ on the root box turn it into two linear rows $p_\pm(x) \leq r$, global on the root box; for nonlinear h each batch recomputes stronger planes on its own boxes, and these rows are local to those boxes and their descendants. A negative radicand certifies that the square row has no feasible point.

Symmetry. On the exact forms (CPU), a chain of disjoint variable blocks $B_1, \ldots, B_m$ with equal root boxes is accepted when every exchange of two consecutive blocks maps the objective to itself and the multiset of rows to itself, compared exactly. Adjacent exchanges generate all permutations of the blocks, so every feasible point can be permuted, at the same objective value, into $x_{B_1[1]} \leq \cdots \leq x_{B_m[1]}$ (the first coordinates of the blocks). These $m - 1$ ordering rows are added as global linear rows, which constrained concretization uses at every node on the GPU, and incumbents are sorted into the ordered region. Chains used together are disjoint. This removes symmetric copies of the optimum without excluding all of them; the transformation preserves the real objective and feasible

set exactly, although a floating-point evaluation order may differ for the permuted point, which is not re-gated when an incumbent is sorted.

Rewrites for native CROWN bounds. Two proved structures are rewritten into small graphs whose convexity CROWN's own relaxations exploit. A quadratic objective (Appendix B.3) is written at an anchor $\hat{x}$ in the eigenbasis of its Hessian on the free coordinates, $H = V\,\mathrm{diag}(\lambda)V^\top + E$ with E the residual of the decomposition:

$$f_0(x) = f_0(\hat{x}) + g^\top d + \tfrac{1}{2}\textstyle\sum_k \lambda_k z_k^2 + \tfrac{1}{2} d^\top E d, \qquad d = x - \hat{x}, \quad z = V^\top d, \tag{58}$$

an exact identity whose last term is at least $-\frac{1}{2}\|E\|_2 \max_{x\in B}\|d\|^2$ on a box B. The graph $\mathrm{Linear} \to \mathrm{Sqr} \to \mathrm{Linear}$ is built on the GPU (Hessian by automatic differentiation, symmetric eigendecomposition) and rebuilt when the anchor moves. Each posynomial $p(x) = \sum_k c_k x^{a_k}$ ($c_k > 0$, real exponent vectors a_k) is written in $y = \log x$ as $\sum_k \exp(a_k^\top y + \log c_k)$, a sum of convex terms (Figure 8); this graph is built and bounded on the CPU.

B.2 Root Schedule

The root steps below run in a fixed order under the single deadline of the run. After every step the gap rule (13) is checked against the best verified point and the best bound, and the run ends as soon as it holds; otherwise the remaining open boxes start the GPU branch-and-bound of Section 3. In the run of Section 4, 293 of the 421 solved instances close at the root and 128 in the tree. By the first bound that reaches $P - \varepsilon$ on each instance, the lifted LP and its branch-and-bound close 214, CROWN 35, the separable floors 20, the posynomial and relative-entropy certificates 16, the special-structure certificates 7 and another root bound 1; in the GPU tree, 94 close from the root box and 33 from the lifted-LP leaves, and the infeasibility of `mosel_start_folioqpgraph` is proved in the tree. The first bound to close an instance need not be the only one that would. The primal searches of Appendix B.6 overlap these steps in parallel root mode.

B.3 Root Dual Bounds and Certificates

CROWN and separable floors. CROWN, without α optimization, gives the planes (7) of all outputs over the root box on the GPU; these planes are also the rows of the dual LP. When the objective decomposes along the graph as $f_0 = \kappa + \sum_j c_j t_j(x)$ with at most 64 terms, each term is bounded separately,

$$f_0 \;\ge\; \kappa + \sum_{c_j>0} c_j \max\bigl(\underline{t}_j^{\mathrm{IBP}}, \underline{t}_j^{\mathrm{CROWN}}\bigr) + \sum_{c_j<0} c_j \min\bigl(\overline{t}_j^{\mathrm{IBP}}, \overline{t}_j^{\mathrm{CROWN}}\bigr) \quad \text{on the root box,}$$

and this value is a floor for every box. The terms are bounded on the GPU one after another and summed on the host.

Quadratic tangent bound. A degree propagation on the graph proves on the CPU a sufficient bound $\widehat{\deg} f_0 \le 2$ ($\deg x_i = 1$, $\deg(uv) = \deg u + \deg v$, and ∞ for non-polynomial operators; failure does not imply that f_0 is not quadratic). If it succeeds and $n \le 2{,}000$, then $f_0(x) = f_0(\hat{x}) + g^\top d + \frac{1}{2} d^\top H d$ exactly for any anchor $\hat{x}$, with $g = \nabla f_0(\hat{x})$ and a constant Hessian H. H and an estimate $\lambda_{\mathrm{lb}} = \lambda_{\min}(H) - 10^{-9}(1 + \|H\|_F)$ of a lower bound on its smallest eigenvalue are computed on the GPU in floating point; the cushion is a numerical safety allowance, not a verified enclosure of the true Hessian, so the bounds below are rigorous only if it holds. The anchor $\hat{x}$ is an approximate KKT point of the root problem with every row relaxed by the tolerance (accepted from IPOPT's exit status and a finite feasibility tolerance, without a KKT proof), found by IPOPT on the CPU (at most 10 s), whose solve also proposes incumbents. The anchor then moves to each improved incumbent, except that an anchor of a convex objective with affine rows, whose plane then bounds the feasible set by $f_0(\hat{x})$ when it is an exact minimizer, is kept unless an incumbent improves on $f_0(\hat{x})$. Any anchor gives a valid plane when the curvature and data errors are enclosed, but tightness at the anchor is not guaranteed. The tangent plane alone gives the quadratic tangent bound (QTB)

$$f_0(x) \;\ge\; f_0(\hat{x}) + g^\top d + \tfrac{1}{2}\min(\lambda_{\mathrm{lb}}, 0)\textstyle\sum_j \max\bigl\{(\hat{x}_j - \underline{x}_j)^2, (\overline{x}_j - \hat{x}_j)^2\bigr\} \qquad (x \in B),$$

which lowers every direction by the most negative curvature. The solver instead runs CROWN on the rewrite (58), per box on the GPU: with $[\underline{z}_k, \overline{z}_k]$ the interval of z_k on B, the term $\frac{1}{2}\lambda_k z_k^2$ is bounded

by the tangent at $\mathrm{proj}_{[\underline{z}_k,\overline{z}_k]}(0)$ if $\lambda_k > 0$ and by the chord if $\lambda_k < 0$ (Figure 7). On a box that holds $\hat{x}$ the positive directions reproduce QTB's plane, and elsewhere they can be tighter, when the interval of z_k excludes 0; a negative direction is bounded by its chord, exact at both ends of its interval, instead of by the most negative curvature at the farthest corner of the box. The resulting plane, lowered by $\frac{1}{2}\|E\|_2 \max_{x\in B} \|d\|^2$ (with $\|E\|_2$ computed in floating point) and an empirical rounding cushion, replaces the CROWN objective plane in a second dual LP over the same rows (Appendix B.5). When $\lambda_{\mathrm{lb}} > 0$ the curvature is also kept: for the multipliers $y \geq 0$ of that LP (and for $y = 0$), $f_0(x) + y^\top(Gx - h) \geq f_0(\hat{x}) + g^\top d + \frac{\lambda_{\mathrm{lb}}}{2}\|d\|^2 + y^\top(Gx - h)$ is separable and minimized at $x_j^\star = \mathrm{proj}_{[\underline{x}_j,\overline{x}_j]}(\hat{x}_j - q_j/\lambda_{\mathrm{lb}})$ with $q = g + G^\top y$; the bound is evaluated rigorously after linearizing $\frac{\lambda}{2}t^2 \geq st - \frac{s^2}{2\lambda}$ at $s = \lambda(x_j^\star - \hat{x}_j)$.

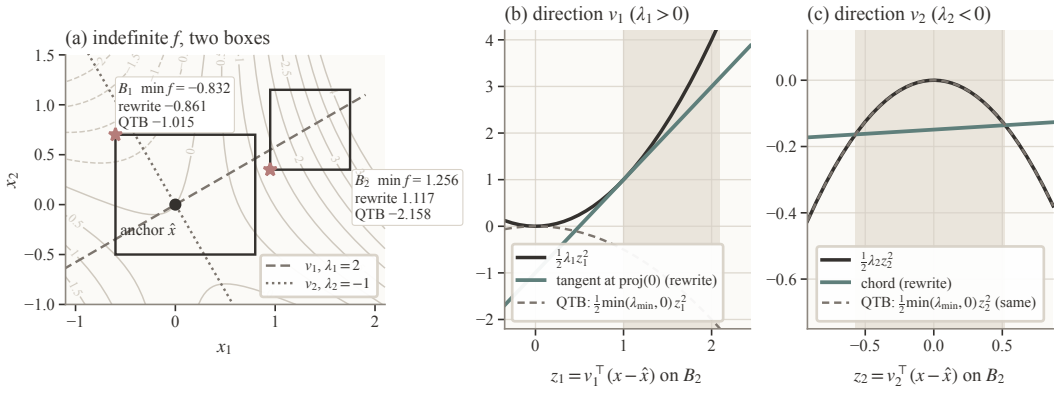

Figure 7: QTB vs. CROWN on the eigenbasis rewrite, for $f(x) = g^\top x + \frac{1}{2}x^\top Hx$ with eigenvalues $(2, -1)$, anchor $\hat{x} = 0$ and $g = (0.4, -0.3)$. (a) Contours, eigen-directions and two boxes ($\star$: box minimizers). On $B_1 \ni \hat{x}$ the rewrite certifies -0.861 and QTB -1.015 against the minimum -0.832; on B_2, away from the anchor, 1.117 and -2.158 against 1.256. (b,c) On B_2, the rewrite bounds $\frac{1}{2}\lambda_k z_k^2$ on the interval of z_k (shaded) by the tangent at $\mathrm{proj}(0)$ ($\lambda_1 > 0$) or by the chord ($\lambda_2 < 0$); QTB replaces every λ_k by $\min(\lambda_{\min}, 0)$ and then $\|x - \hat{x}\|^2$ by its maximum over the box.

Convex objectives and rows. A non-quadratic objective proved convex on the root box (composition rules on the CPU, each univariate second derivative enclosed in interval arithmetic on 64 uniform slices of its range, augmented with dyadic slices at both ends) contributes its tangent plane $f_0(\hat{x}) + \nabla f_0(\hat{x})^\top(x - \hat{x})$, lowered by the enclosure errors, to the same second LP. Constraint rows $g_i(x) \leq t_i$ that are quadratic and admitted as numerically near-convex ($\lambda_{\min}$ estimate $\geq -10^{-8}$, the negative-curvature term below being kept), or convex by the same rules, give at the incumbent $\hat{x}$ the global linear rows $g_i(\hat{x}) + \nabla g_i(\hat{x})^\top(x - \hat{x}) + \frac{1}{2}\min(\lambda_{\mathrm{lb},i}, 0)\max_{x\in B_0}\|x - \hat{x}\|^2 \leq t_i + \tau$ (the curvature term only for quadratic rows, B_0 the root box, minus a rounding cushion). They are refreshed when the incumbent improves, for at most four rounds, and only when there are at most 50 candidate quadratic rows; the Hessians are computed on the GPU, one row at a time.

Posynomial (geometric-program) certificate. When the root box is strictly positive and the exact forms show the objective to be a posynomial plus a constant, the substitution $x = e^y$ makes the objective and every posynomial row convex in y; other rows are dropped, which only weakens the bound. A positive monomial equality with tolerance is equivalent to two log-affine inequalities when both endpoints are positive; in the enabled CROWN certificate path both sides are instead written as positive monomial or reciprocal rows and relaxed by exponential tangents. A Kelley cutting-plane LP in the epigraph of the objective, whose constant offset κ is added back,

$$\min\ \kappa + t \quad \text{s.t.} \quad t \geq \pi_0^{(j)}(y), \quad \pi_i^{(j)}(y) \leq t_i + \tau, \quad \log \underline{x} \leq y \leq \log \overline{x},$$

is solved by an LP solver on the CPU, where each cut $\pi^{(j)}(y) = \sum_k e^{m_k}(1 + a_k^\top y + \log c_k - m_k) \leq p(e^y)$ is CROWN's tangent of the exponential terms of the log-coordinate rewrite at $m_k = a_k^\top y^{(j)} + \log c_k$ (clamped to the term's range), taken at the box centre, the incumbent and then each LP point (Figure 8). CROWN only proposes the slopes: the intercepts and the LP's dual value are re-evaluated in 50-digit interval arithmetic, so the reported bound is rigorous for the cut data. The loop stops when

a round after the fifth no longer improves the bound, or after 20 s. The certificate, CROWN included, runs on the CPU.

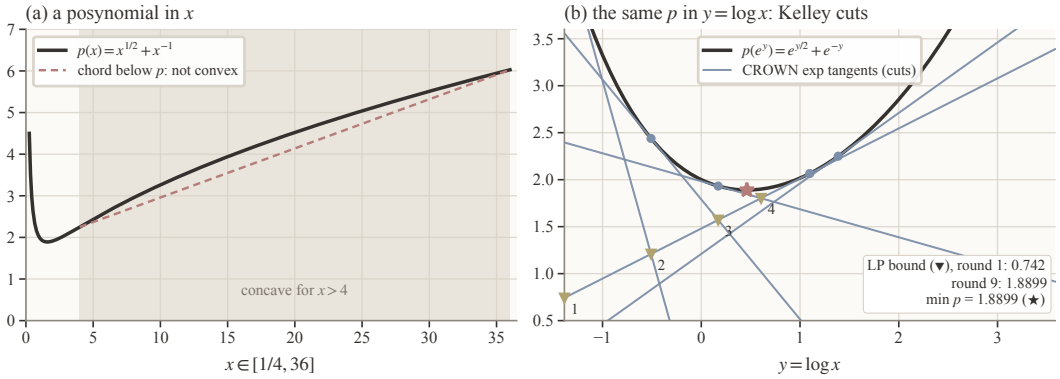

Figure 8: Posynomial certificate for $p(x) = x^{1/2} + x^{-1}$ on $[1/4, 36]$ (CPU). (a) p is concave for $x > 4$, so it is not convex in x. (b) In $y = \log x$, $p(e^y) = e^{y/2} + e^{-y}$ is convex and CROWN's tangents of the exponential terms are global lower lines. The Kelley LP over these cuts (points: box centre, incumbent $x = 4$, then each LP point; ▼: LP minimizers of rounds 1-4) certifies 0.742 in round 1 and the minimum 1.8899 to four digits in round 9.

Relative-entropy certificate. When the variables enter through $y_j = e^{x_j}$, the objective is a sum of relative-entropy terms $w\,u\log(u/v)$ plus an affine part, with weights $w \geq 0$ and u, v proved strictly positive on the transformed box (then jointly convex), and the rows are affine in y. The implementation recognizes numerators y_j or 1 and affine denominators, retains usable affine rows and drops unsupported ones. Each term gets its own epigraph variable with outward-rounded tangent cuts, and the LP dual value of the LP solver's multipliers, evaluated in interval arithmetic, is the bound (CPU).

Special-structure certificates. Three detectors match exact algebraic patterns of the forms and rows and prove a floor with a dedicated argument; numerical solvers only propose. (i) A polygon area in polar coordinates whose law-of-cosines rows bound every pairwise distance by d: the convex hull of the N vertices has diameter at most d, hence perimeter at most $2Nd\sin\frac{\pi}{2N}$ and area at most $Nd^2\sin^2\frac{\pi}{2N}/\tan\frac{\pi}{N}$, evaluated in interval arithmetic on the CPU; for the tolerance-relaxed polar model the floor additionally carries the allowance for the order rows' backward angles (with d the relaxed distance threshold). (ii) A normalized symmetric eigenvalue block with an affine objective in the eigenvalue: for any computed factor L and $\rho \geq \|A - sI - LL^\top\|_\infty$ (induced infinity norm, the maximum absolute row sum; rounded outward), $A - (s-\rho)I \succeq LL^\top \succeq 0$, so $\lambda_{\min}(A) \geq s - \rho$. Because the eigenpair rows hold only within a tolerance, with $\|q\|^2 \geq L_q > 0$, $|(Aq - \lambda q)_i| \leq \epsilon_i$ and $f_0 = a\lambda + b$, $a > 0$, the floor is $b + a\big(s - \rho - \sqrt{\sum_i \epsilon_i^2/L_q}\big)$; the eigendecomposition, the Cholesky factor and the residual run on the GPU. (iii) Near-binary assignment rows: for $\tau < \frac{1}{4}$, $|x(x-1)| \leq \tau$ puts x within $\delta = 2\tau/(1+\sqrt{1-4\tau})$ of $\{0,1\}$; after proving $0 \leq x \leq 1$, the assignment structure and $m\delta + \epsilon < \frac{1}{2}$ for every cardinality row, every relaxed-feasible point rounds to a partition. Each enumerated partition receives an outward-rounded lower bound over its allowed near-binary deviations, and a cover search minimizes the sum of these bounds in GPU kernels (prices from an LP solver on the CPU). Each pattern matches a single family of the benchmark (`polygon`, `eigmin`, `grouping`).

B.4 Lifted Linear Relaxation

The plane LP (8) relaxes every row by its own plane; the lifted LP keeps each nonlinear term as one column shared by all rows. It is built once at the root. Its LP solves (bound tightening, the lifted root bound and the LP branch-and-bound) are floating-point LP solves on the CPU, serial and synchronous between GPU iterations, whose multipliers are then evaluated by the certified weak-duality bound below; the GPU batches bound the boxes admitted by the dense budget over the same rows with PDHG, and the other boxes keep their other bounds.

Lifting. Problems with at most 2,000 variables and a finite root box are lifted. A row f_i ($i = 0$ is the objective) whose exact form (Appendix B.1) is a polynomial in the inputs and in transcendental atoms (at most 64 in the problem) becomes linear in the columns:

$$f_i(x) = s_i\big(z(x)\big), \qquad s_i(z) = k_i + g_i^\top z, \qquad z = (x,\, t,\, w). \tag{59}$$

There is one column $t_a = \phi_a(\alpha_a^\top z + \alpha_{a0})$ per atom of an affine argument ($\phi \in \{\exp, \log, x^p, \sin, \cos\}$, $1/x$ included) and one column $w_k = z_{a_k} z_{b_k}$ per distinct product; a monomial of higher degree is split into halves ($x^4 = (x^2)(x^2)$), so both factors of a product are earlier columns. Quadratic rows take their coefficients from automatic differentiation at $x = 0$ on the GPU, with a numerical sanity check at four random points (not a coefficient-error enclosure); the other rows take them from the exact forms on the CPU. On a box B, interval arithmetic over the columns gives a column box $Z(B) = [l^z, u^z]$. With the relaxation and RLT rows $E_B z \le e_B$ below, the base linear rows included, and an incumbent value P, define

$$\mathcal{L}(B, P) = \big\{z \in Z(B) :\ s_i(z) \le 0\ (i \ge 1),\ E_B z \le e_B,\ s_0(z) \le P\big\}. \tag{60}$$

In exact arithmetic every $x \in B$ that satisfies the rows with $f_0(x) \le P$ lifts to $z(x) \in \mathcal{L}(B, P)$, so $\min\big(P, \min_{\mathcal{L}(B,P)} s_0\big)$ bounds the problem on B. The first-level product endpoints of $Z(B)$ are computed in floating point without outward rounding (deeper levels are widened), so this containment holds up to rounding, which the rows' cushion absorbs only numerically: each right-hand side is relaxed by $10^{-12}\big(1 + |k_i| + \sum_j |g_{ij}| \max(|l_j^z|, |u_j^z|)\big)$. The implementation invokes the standalone lifted root and tree bound only when the objective and every row with a finite threshold are lifted (a partial relaxation would also be valid, if weaker); the other cases use it for bound tightening and for the per-box bound (with CROWN planes for the other rows). A convex quadratic objective over linear rows is left to the quadratic tangent bound (Appendix B.3), which is exact there.

Why lifting helps. Let R_i hold the points (x, v, w) that satisfy the relaxation of row i (for the objective, $v \ge s_0$) and the relaxations of the columns it uses, and let π project onto (x, v). A row relaxed by itself in x-space keeps a private copy of w, so the best per-row relaxation minimizes v over $\bigcap_i \pi(R_i)$; the lifted LP shares w and minimizes over $\pi\big(\bigcap_i R_i\big) \subseteq \bigcap_i \pi(R_i)$. The inclusion can be strict (Figure 9).

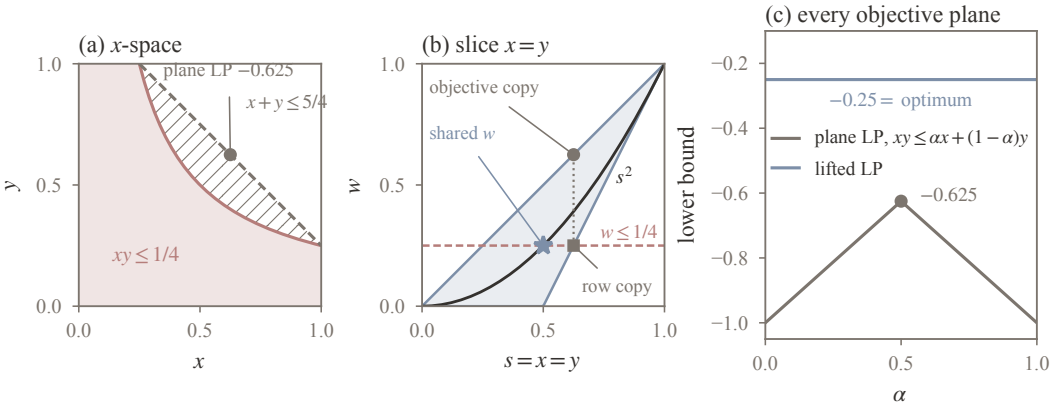

Figure 9: Per-row planes versus one shared column on $\min -xy$ subject to $xy \le \frac{1}{4}$ over $[0, 1]^2$ (optimum $-\frac{1}{4}$). (a) The plane LP relaxes the objective to $-(\alpha x + (1-\alpha)y)$ (from $xy \le x$, $xy \le y$) and the row to $x + y - 1 \le \frac{1}{4}$; it admits the hatched points. (b) On the diagonal $x = y = s$, its optimum gives the same product two values: $\frac{5}{8}$ in the objective's copy and $\frac{1}{4}$ in the row's. Full McCormick envelopes with one copy per row still give -0.625; one shared column w gives -0.25. (c) Over all objective planes the plane LP reaches at most -0.625 (at $\alpha = \frac{1}{2}$); the lifted LP reaches -0.25, the optimum. All values are LP optima.

Operator relaxations. A product $w = x_a x_b$ gets the four McCormick faces of $(x_a - l_a)(x_b - l_b) \ge 0$, $(u_a - x_a)(u_b - x_b) \ge 0$, $(u_a - x_a)(x_b - l_b) \ge 0$ and $(x_a - l_a)(u_b - x_b) \ge 0$; for example, $w \ge l_b x_a + l_a x_b - l_a l_b$. A square gets its secant $w \le (l + u)x - lu$ and the tangents $w \ge 2\theta x - \theta^2$

at $\theta \in l + \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}(u - l)$. An atom of fixed curvature on its argument range $[\underline{\alpha}, \overline{\alpha}]$ (exp; log on a positive range, x^p on a one-signed range of its domain) gets five tangents on its convex side and the secant on the other. sin (and $\cos \alpha = \sin(\alpha + \frac{\pi}{2})$) gets support lines on both sides $\sigma = \pm 1$:

$$\sigma\,(t - m\alpha) \le H_\sigma(m) = \max_{\alpha \in [\underline{\alpha}, \overline{\alpha}]} \sigma(\sin\alpha - m\alpha). \tag{61}$$

These hold on a range of any width. For $|m| \le 1$, $H_\sigma(m)$ is the largest of $\sigma(\sin\alpha - m\alpha)$ at the two ends and at the first and last $\alpha = \sigma \arccos m + 2\pi k$ in range; for $|m| > 1$ the ends suffice. The slopes m are cos at $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ of the range and the tightest slopes through either end (by bisection). At the root, two LPs per trigonometric atom certify its argument range, and 16 cutting-plane rounds add the slope at the LP point. Floating-point atom values are widened by 10^{-14} relative, a numerical allowance that assumes libm accuracy, including for large-argument phase reduction. The relaxations are rebuilt on every box: in dense batches on the GPU for PDHG, and sparse on the CPU for the LP solver (Figure 10).

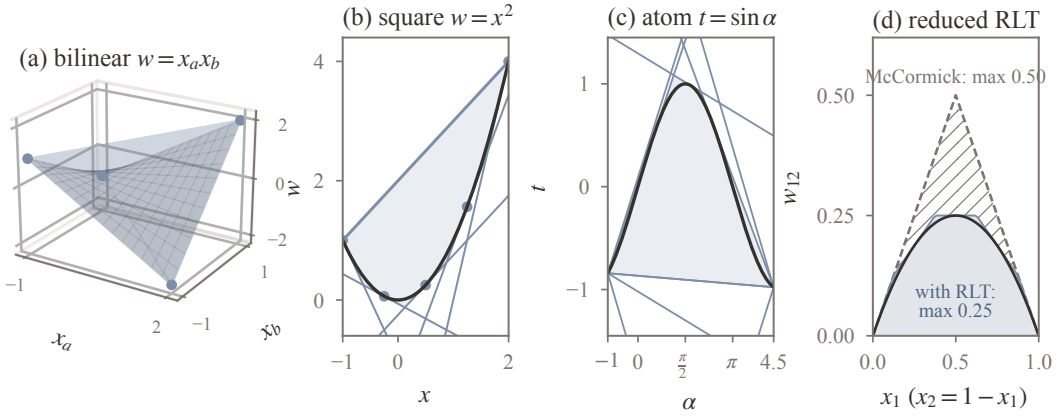

Figure 10: Relaxation rows on one box. (a) The four McCormick faces of $w = x_a x_b$, the convex hull of the saddle over the box. (b) $w = x^2$: the secant above, and tangents at $l + \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}(u - l)$ below. (c) $t = \sin\alpha$ on $[-1, 4.5]$, across two inflection points: support lines $\sigma(t - m\alpha) \le H_\sigma(m)$ with tangent slopes at $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ of the range and the tightest slopes through both ends. (d) Reduced RLT for $x_1 + x_2 = 1$ on $[0, 1]^2$ with the product $x_1 x_2$. Multiplying by x_1 and by x_2 adds $w_{11} + w_{12} = x_1$ and $w_{12} + w_{22} = x_2$. The projection on (x_1, w_{12}) shrinks from the McCormick triangle (hatched, $\max x_1 x_2 = 0.50$) to a band over $x_1(1 - x_1)$ ($\max 0.25$, exact).

Cleared denominators and convex objectives. A row $p(x)/m(x) \le \tau$ whose denominator m is a monomial of strictly positive root variables adds the cut $m(x)\big(p(x)/m(x) - \tau\big) \le 0$ when this polynomial has degree at most two. The cut uses exact rational coefficients with an outward conversion error, and it is dropped when a later threshold is looser. A non-quadratic objective proved to be $q_0 + \sum_k c_k \varphi_k(q_k)$ ($c_k > 0$, φ_k convex, q_k quadratic) gets one epigraph column $e_k \ge \varphi_k(q_k)$ per term, with linear cuts $e_k \ge \varphi_k(\theta) + \varphi_k'(\theta)\big(s_{q_k}(z) - \theta\big)$, s_{q_k} being q_k as an affine function of the columns (convexity of $\varphi_k \circ q_k$ in x is not needed). Its tangents are taken at 12 grid points and up to 32 retained points, and on the CPU each LP adds tangents at its own solution (at most 8 rounds). Both structures are recognized once on the CPU.

Reduced RLT. A pair of opposite linear rows $g^\top x - b \in [-\varepsilon^-, \varepsilon^+]$ (an equality with its tolerance) is multiplied by x_i when $x_i x_j$ is a product column for some j in the support of g. Since $x_i(g^\top x - b) = \sum_j g_j w_{ij} - b x_i$, this gives $-\varepsilon^- x_i \le \sum_j g_j w_{ij} - b x_i \le \varepsilon^+ x_i$ when $l_i \ge 0$, the reversed pair $\varepsilon^+ x_i \le \sum_j g_j w_{ij} - b x_i \le -\varepsilon^- x_i$ when $u_i \le 0$, and the interval product $[l_i, u_i] \cdot [-\varepsilon^-, \varepsilon^+]$ otherwise (when x_i crosses zero), with $\varepsilon^\pm$ taken from the outward-relaxed source rows. Missing products $x_i x_j$ become new columns with their own relaxations. The rows are added after the McCormick warm-up below, all of them or none: at most 1,200 RLT rows and 7,000 columns. The selection runs once on the CPU, and the RLT block of each box is assembled on the host, also for the GPU batches.

Bound tightening against the cutoff (OBBT). Two minimization LPs per column give certified lower bounds $D_j^- \le \min_{\mathcal{L}(B,P)} z_j$ and $D_j^+ \le \min_{\mathcal{L}(B,P)}(-z_j)$, and

$$l_j \leftarrow \max\big(l_j,\, D_j^-\big), \qquad u_j \leftarrow \min\big(u_j,\, -D_j^+\big). \tag{62}$$

The new box keeps every point that satisfies the rows with $f_0 \le P$, the incumbent included (an audit checks it). The targets are the variables of the products and, for the auxiliary bounds below, the atom and product columns, admitted by their measured shrink per LP. The relaxations are rebuilt before each target. At the root, a McCormick warm-up (at most 5 s), RLT and a second pass (10 s in all) run at most five rounds each, until the mean relative shrink falls below 1%. Later passes run after an incumbent improvement of more than 10^{-4} (relative), or while a pass cut short by its budget was still shrinking (5 s each, 20 s in all, until a pass shrinks the mean width by less than 2%). The LPs are serial solves on the CPU; every GPU box is intersected with the latest box, elementwise on the GPU (Figure 11a).

Lifted root bound. Once an incumbent exists, the unified root (Appendix B.2) runs the node solve below on the current box, after the LP is built and again after the warm-up and RLT. The run ends if this bound closes the gap.

LP branch-and-bound. The tree runs on the CPU, best first and one node at a time, and carries the cutoff row $s_0 \le P$. At a node B', up to three passes of outward-rounded interval propagation over the rows of $\mathcal{L}(B', P)$ tighten the box, and the relaxations are rebuilt after each pass. Dual simplex then solves the LP from the parent's basis (at most $5(m+N)$ iterations and 10 s). OBBT probes run on the 32 variables with the largest violation times width when the LP has at most 1,200 rows; probing stops when the first ten probed nodes show that fewer than 20% of the probes pay. A node keeps $\beta(B') = \max(\beta(\text{parent}), D(y))$, and the tree bound is $D = \min(P, \min_{\text{open}} \beta)$. The branching variable maximizes

$$\min\Big(1, \frac{\rho_j}{0.05 \max_i \rho_i}\Big)\Big(\sum_{k \ni j} |\hat{w}_k - \hat{z}_{a_k}\hat{z}_{b_k}| + \sum_{a \ni j} |\hat{t}_a - \phi_a(\hat{\alpha}_a)|\Big) \tag{63}$$

at the LP point, where ρ is the width relative to the tree's root box. The split is at the LP point clamped to the middle 80%. The tree stops when the gap closes, after 30 s, when its root bound lies further below the GPU root bound than that bound lies below P, on a plateau, or, for a hand-off, at $\max(256, 32n)$ open leaves. For incumbents, IPOPT starts from the best node's LP point at nodes $128, 256, \ldots, 2048$ until one start improves the incumbent. At the end, the leaf LP points, leaf LPs re-solved with their rows at half the tolerance, and blends towards the incumbent are checked at 10^{-6}, and IPOPT starts from the three lowest leaves. A verified improvement lowers the cutoff.

Weak-duality and Farkas certificates. For an LP $\min\{c^\top z + c_0 : Gz \le h,\ z \in [l^z, u^z]\}$ and any $y \ge 0$,

$$D(y) = c_0 - h^\top y + \sum_j \min\big(r_j l_j^z,\, r_j u_j^z\big) \ \le\ \text{LP optimum}, \qquad r = c + G^\top y. \tag{64}$$

The CPU LP solver and PDHG only propose y; $D(y)$ is evaluated with outward rounding (Appendix B.7) by one routine, on CPU tensors for the CPU solver and on the GPU for PDHG. A box is declared empty only when some $y \ge 0$ gives $\min_{z \in [l^z, u^z]} y^\top(Gz - h) > 0$, that is, (64) with $c = 0$, $c_0 = 0$ is positive. If the rows used include the incumbent cutoff $s_0 \le P$, this proves only that no point with $f_0 \le P$ exists in the box (the box is retired by dominance, not shown infeasible for the original model). Such a y is a Farkas ray, the multipliers of the phase-one LP $\min\{\eta : Gz - \eta\mathbf{1} \le h,\ z \in [l^z, u^z],\ \eta \ge 0\}$, a single row of the propagation, or a probe interval that misses the box. Selected lifted-LP solves, notably those with RLT, are repeated without presolve when presolve calls the LP infeasible and budget remains; an infeasibility status alone never suffices without a separately evaluated certificate.

Hand-off of open leaves. The open leaves (B_i, β_i) cover every point of the box that satisfies the rows with $f_0 < P$, so the root bound rises to $\min(\min_i \beta_i, P)$. At most 8,192 leaves are copied to the GPU once, and they become the first boxes of the GPU tree with their bounds (Figure 11b).

Certificate transfer. For the last LP of each open leaf (box B, cutoff P_0), $r = c + G^\top y$ and $\kappa = c_0 - h^\top y$ are computed once in exact rational arithmetic and stored as floats with error bounds. On $\mathcal{L}(B, P_0)$, $c^\top z + c_0 = \kappa + r^\top z + y^\top(h - Gz) \geq \kappa + r^\top z$, and the leaf's rows, built on B, hold on every $B' \subseteq B$. So, while the incumbent is at most P_0, every GPU box $B' \subseteq B$ has

$$\beta(B') \;\geq\; \min\Big\{P_0,\; \kappa + \sum_j \min\big(r_j l'_j,\; r_j u'_j\big)\Big\}, \qquad [l', u'] = Z(B') \cap [l^z, u^z], \tag{65}$$

and it keeps the larger of this and its own bound. The value grows as B' shrinks, through the columns with $r_j \neq 0$; at an exact optimal primal-dual pair these sit at a bound in the leaf LP, but this is only intuition, since validity does not require complementarity. In exact arithmetic the value is nondecreasing for nested intervals; the floating-point values are kept monotone by the explicit maximum with the box's own bound. Each batch matches its boxes to the scopes on the host (a box hierarchy) and evaluates the bounds on the GPU, without an LP (Figure 11c).

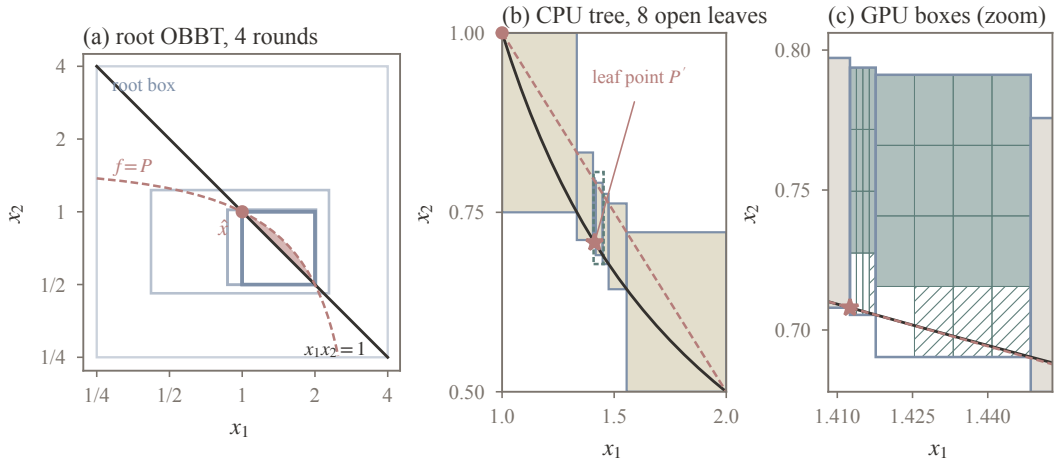

Figure 11: From the root box to the GPU tree on $\min x_1 + 2x_2$ subject to $x_1 x_2 \geq 1$ over $[\frac{1}{4}, 4]^2$, with incumbent $\hat{x} = (1, 1)$, $P = 3$ (optimum $2\sqrt{2} \approx 2.828427$). (a) Root OBBT with the cutoff row $x_1 + 2x_2 \leq P$ (log axes). Four rounds shrink the box to approximately $[1, 2] \times [\frac{1}{2}, 1]$ at the displayed precision, the bounding box of $\{x_1 x_2 \geq 1,\, f \leq P\}$ (shaded), and raise the node bound (with propagation) from 2.588 to 2.667. The toy LP rows carry no modelling slack; candidate acceptance uses 10^{-6}. Without the cutoff row the box does not move. (b) The lifted-LP tree on this box, stopped at 8 open leaves for display (the solver stops at $\max(256, 32n)$); node propagation removed the white parts. A leaf's LP point verifies as $P' = 2.828429$ (star). (c) Against P', 6 leaves fall to their own bounds in the GPU tree. The other two (dashed window in (b)) are cut into 16 boxes each. Of these 32 boxes, the leaf's certificate $\kappa + r^\top z$ alone prunes 24 (filled), fresh lifted LPs prune 4 more (hatched), and 4 stay open. Computed with a small re-implementation of the same rows.

Auxiliary bounds for CROWN. Inside its box, an OBBT interval of an atom or product column holds for every point that satisfies the rows with $f_0 \leq P_0$; so does the column box of a leaf certificate. Every graph node whose exact form is affine in the columns, $v = a_0 + \sum_j a_j z_j$, gets the interval $a_0 + \sum_j [\min, \max](a_j \underline{z}_j, a_j \overline{z}_j)$, summed in exact rationals and rounded outward, and this interval is intersected with CROWN's intermediate bounds of v. These bounds hold only below P_0, so the objective bound they produce is capped at P_0. Matching and the exact sums run on the host, once per new scope; the intersection runs on the GPU.

Per-box lifted bound in the GPU tree. Every GPU batch also bounds its boxes over $\mathcal{L}(B, \infty)$, with no cutoff row and no propagation. Rows that are not lifted, and the objective when it is not lifted, enter as their CROWN planes on B. The LPs of the lowest-bound boxes that fit in $2.5 \cdot 10^7$ dense coefficients run 500 PDHG iterations on the GPU, with $D(y)$ certified every 50. A CPU LP solver handles the 16 lowest in floating point (0.5 s per batch). The RLT and epigraph rows are assembled per box on the host. Where this bound sets a box's bound, the box branches on the largest violation (Appendix B.5).

Continuation. The open leaves still cover every point below a lower incumbent, so the lifted tree can resume from them; if the root had no incumbent, the tree starts here instead. It runs only while the GPU tree's bound has not passed the lifted tree's D, and the GPU tree counts as ahead until it has visited twice as many boxes as it received. The tree is first continued once at the same incumbent, to measure its progress. After that, a continuation needs an incumbent improvement of more than 10^{-4} (relative) or a previous continuation that halved its gap. It also needs a measured gain per second (bound gain beyond the GPU bound, plus incumbent gain) that is positive and at least the GPU tree's. A call takes at most $\min(30\,\mathrm{s},\ 0.3\times$ the remaining time), and all calls together at most 60 s unless the last one halved its gap. Each call runs on the CPU while the GPU loop waits.

B.5 One Step of the GPU Branch-and-Bound

An iteration selects a batch, runs the scheduled primal search on it (Appendix B.6), bounds and prunes it, and splits the survivors (Figure 13); per-box work is batched on the GPU, while the host keeps P (the incumbent value, UB in Section 2) and D.

Selection (GPU). The batch $\mathcal{B}_t$ holds the k_t boxes of $\mathcal{Q}$ with the lowest keys, $\max_{\mathcal{B}_t} \mathrm{LB} \le \min_{\mathcal{Q}\setminus\mathcal{B}_t} \mathrm{LB}$ (a top-k in every iteration), so it contains a box of minimum bound, as Appendix A.13 requires. $k_0 = 2$; k doubles while the last batch was full and less than 45% of the device memory is reserved, and is halved for good after an out-of-memory error, the batch returning to $\mathcal{Q}$ unchanged.

Bounding. The tensor bounding kernels and the admitted dense PDHG batches run on the GPU, batched over the boxes. Host work includes the symbolic and scope processing, the RLT and epigraph row assembly and the scheduled LP solves; budget-excluded boxes keep their other bounds. Interval bounds are computed first; the boxes they do not settle are bounded by CROWN, which yields the planes $\underline{f}_i(x) = a_i^\top x + c_i$ of (7). Every concretization (objective and intermediate nodes) is constrained by the rows available before the pass: the parent's planes (the objective plane cut at the incumbent included), the linear rows and the presolve cuts, at most the 16 non-redundant rows with the largest maximum over the box. The rows act one after another, each through its single-row dual

$$\beta_k \in \arg\max_{\beta\ge 0} \min_{x\in B} \big[(a+\beta a_k)^\top x + b + \beta c_k\big], \qquad (a,b) \leftarrow (a+\beta_k a_k,\ b+\beta_k c_k),$$

a concave piecewise-linear function of β maximized over $\beta = 0$ and the nonnegative breakpoints $-a_j/a_{kj}$, $a_{kj} \neq 0$ (a fused kernel in double precision); a row whose slope stays positive as $\beta \to \infty$ proves the box empty. After branching, each child is clipped with the fresh planes by one sweep of (25). The objective bound is raised by the dual $d_B(\lambda)$ of (9) over the fresh rows $Ax+c \le 0$ (constraint planes, linear rows and presolve cuts; we write $G = A$, $h = -c$). Every $\lambda \ge 0$ gives $d_B(\lambda) \le p_{\mathrm{LP}}(B)$ (Theorem A.1), so three methods only propose multipliers (Figure 12). (i) Twenty projected Adam steps (22) from $\lambda = 0$ with learning rate 0.05, keeping the best iterate (not outward-rounded, Appendix B.7). (ii) PDHG on the same LP: with $x = \underline{x} + w \odot z$, $z = D_c\tilde{z}$, $w = \overline{x} - \underline{x}$, ten rounds of Ruiz equilibration $\tilde{G} = D_r G\,\mathrm{diag}(w) D_c$ and diagonal steps $\tau_j = 1/\max(\sum_i |\tilde{G}_{ij}|, 10^{-12})$, $\sigma_i = 1/\max(\sum_j |\tilde{G}_{ij}|, 10^{-12})$,

$$\tilde{z}^+ = \Pi_{[0,\bar{z}]}\big(\tilde{z} - \tau \odot (\tilde{a}_0 + \tilde{G}^\top \tilde{y})\big), \qquad \tilde{y}^+ = \big[\tilde{y} + \sigma \odot (\tilde{G}(2\tilde{z}^+ - \tilde{z}) - \tilde{h})\big]_+,$$

with $\tilde{a}_0 = D_c\,\mathrm{diag}(w) a_0$, $\tilde{h} = D_r(h - G\underline{x})$ and $\bar{z} = D_c^{-1}\mathbf{1}$, for 500 iterations (which only improve the candidate multipliers; certification needs no convergence), warm-started from the batch mean of the previous multipliers; every 50 iterations, and at $\lambda = 0$, d_B is evaluated at $\lambda = D_r\tilde{y}$ with outward rounding and the best value is kept. (iii) For the 16 boxes with the lowest bound (when $mn \le 2\cdot 10^5$), the LP solved in floating point on the CPU in a pool of at most eight threads; its multipliers $\max(-\text{marginals}, 0)$ are scored by the same rigorous d_B on the GPU. When the solver reports infeasibility, the multipliers ν of the phase-one LP $\min\{t : Ax + c \le t\mathbf{1},\ t \ge 0,\ x \in B\}$ are checked as a Farkas certificate (10), $\min_{x\in B} \nu^\top(Ax + c) > 0$, under the same rounding, and the box is removed. The quadratic-tangent plane (a second LP over the same rows, again PDHG plus exact LP solves on its 16 lowest boxes), the lifted LP rows, the transferred certificates and the objective floors of Appendices B.1 and B.3 each contribute a valid bound, and the box keeps the largest.

Pattern propagation. When the objective is affine and every row is affine, a single product, or a signed sum of squared differences (in this benchmark, only the `kall` circle-packing models), three

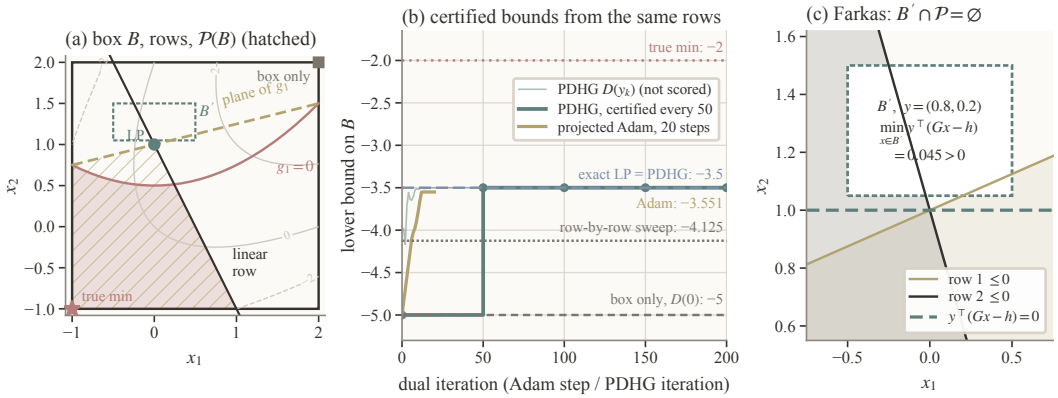

Figure 12: Bounding from CROWN planes on a toy: $\min\ x_1x_2 - \frac{1}{2}x_1^2 - \frac{1}{2}x_2^2 + x_1 + x_2$ s.t. $g_1 = x_2 - \frac{1}{4}x_1^2 - \frac{1}{2} \leq 0$ and $x_1 + \frac{1}{2}x_2 \leq \frac{1}{2}$ on $B = [-1, 2]^2$. (a) Feasible set (shaded), $\mathcal{P}(B)$ from the CROWN plane of g_1 and the linear row (hatched), and the minimizers of the objective plane over B and over $\mathcal{P}(B)$ and of the problem. (b) Bounds from the same rows: box only -5 (interval arithmetic: -8); the row-by-row single-row duals of constrained concretization, -4.125; 20 projected Adam steps, -3.551; PDHG, certified at iteration 50, and the exact LP, $-3.5 = p_{\mathrm{LP}}(B)$; the minimum is -2. (c) On a child box B' each row alone is satisfiable, but the phase-one multipliers $y = (0.8, 0.2)$ give $\min_{x\in B'} y^\top(Gx - h) = 0.045 > 0$, so B' is removed.

rounds of outward interval propagation per box replace the bounding step, and boxes are split along the coordinate with the largest uncertainty of the squared terms.

Pruning (GPU). A box is removed if $\mathrm{LB}(B) \geq P$ (equality included: no strictly better point remains), if a row's lower bound exceeds its threshold, $\underline{f}_i(B) > \tau_i$, or if $\mathrm{LB}(B) = +\infty$, an emptiness certificate of the constrained concretization or of an LP (Equation (10); with a cutoff row among the rows used, a dominance certificate). Children are tested again on entry, together with $l_j > u_j$ after clipping and a positive minimum of the linear rows.

λ-SB branching score (GPU). With $a_r^\top x + b_r$ ($r = 0, \ldots, m$; $r = 0$ the objective) the linear bounds of the box (Equation (7)), $w = u - l$ and the Lagrangian multipliers $\lambda^{\mathrm{best}} \geq 0$, the default choice is

$$j^\star = \arg\max_j\ S_j, \qquad S_j = w_j\Big(|a_{0j}| + \sum_{i=1}^{m} \lambda_i^{\mathrm{best}} |a_{ij}|\Big),$$

the λ-SB score of Equation (12) (the relative-width fallback and the count guard below are part of the evaluated solver). Each term $\frac{1}{2}|a_{rj}|w_j$ is, for its row separately, the gain of that row's box minimum: for the half B^+ of a midpoint split along j that excludes the minimizing vertex of row r, only the j-th term of Equation (6) changes,

$$\min_{x\in B^+}\big(a_r^\top x + b_r\big) = \min_{x\in B}\big(a_r^\top x + b_r\big) + \tfrac{1}{2}|a_{rj}|\, w_j, \tag{66}$$

which for $r = 0$ raises the objective plane's minimum on the far child and for a constraint row can prune it. The λ-weighted sum over rows makes S_j a branching heuristic only: it does not guarantee an equal increase in either child's recomputed bound or in the global lower bound.

Overrides (GPU; lifted LP points partly from the CPU). With $\mathrm{rw}_j = w_j / w_j^0$ the width relative to the root box, four rules replace $j^\star$ in turn. (i) Where the lifted LP (Appendix B.4) sets the bound, $j^\star$ maximizes $\sum_{k\ni j} |z_k^\star - \phi_k(z^\star)|$, the violation at the LP point $z^\star$ of the lifted products and atoms containing x_j, over $\mathrm{rw}_j \geq 0.05 \max_i \mathrm{rw}_i$ (if positive); halving x_j halves the McCormick gap bound $w_i w_j/4$ of each product $x_i x_j$. (ii) If $\mathrm{rw}_{j^\star} < 10^{-3} \max_i \mathrm{rw}_i$, the relatively widest coordinate is taken. (iii) The relatively widest variable of the rotation pairs $a(x_i^2 + x_j^2) = k$ is split while its relative width exceeds 0.15. (iv) Without constraint rows and with $n \leq 6$, both children of every coordinate are bounded for the 10^4 lowest boxes, and $j^\star$ maximizes $\max(\Delta_j^-, 10^{-12}) \max(\Delta_j^+, 10^{-12})$, $\Delta_j^\pm$ the children's gains over $\mathrm{LB}(B)$.

Count guard (GPU). With $c(B)$ the number of splits since the box was last split along its relatively widest coordinate, the last rule is $j^\star \leftarrow \arg\max_j \mathrm{rw}_j$ if $c(B) \geq n_{\mathrm{open}}(B)$, the number of coordinates of positive width. A widest split then comes at least once in every $n+1$ splits, and in exact arithmetic (or while every committed midpoint split genuinely halves its interval) n of them halve $\max_j \mathrm{rw}_j$, since widths never grow; a box at depth k thus satisfies

$$\max_j \mathrm{rw}_j(B) \leq 2^{-\lfloor k/(n(n+1))\rfloor}, \tag{67}$$

the exhaustive subdivision of Appendix A.13, whatever rules (i) to (iv) chose (Figure 13d), with $\mathrm{rw}_j = 0$ when $w_j^0 = 0$. In binary64 an interval of adjacent floats has its midpoint at an endpoint, so the rate cannot be claimed below representable resolution; finite termination also needs the relaxation, primal-recovery and infeasibility-detection assumptions of Appendix A.13.

Split, clipping and child keys (GPU). The box is cut at $m = (l_{j^\star} + u_{j^\star})/2$ into $\{x_{j^\star} \leq m\}$ and $\{x_{j^\star} \geq m\}$, which share the floating-point m and cover B exactly. Each child C is clipped by the parent's rows, the objective row at right-hand side P included (Section 3.3), and queued with the key

$$\kappa(C) = \max\Big(\mathrm{LB}(B),\ \mathrm{rd}_\downarrow \min_{x \in C}\big(a_0^\top x + b_0\big)\Big), \tag{68}$$

where $a_0^\top x + b_0$ is the parent's objective plane and $\mathrm{rd}_\downarrow$ subtracts the forward-error bound $(n+4)\mathbf{u}\big(\sum_i |a_{0i}|(|c_i| + r_i) + |b_0|\big)$ of the vertex formula (c, r the centre and half-width of C, $\mathbf{u}$ the unit roundoff) and rounds toward $-\infty$; the allowance assumes finite intermediates, covered underflow and $2(n+4)^2\mathbf{u} \leq 1$. With $q_B = \min_{x \in B}(a_0^\top x + b_0)$ and $\Delta = \frac{1}{2}|a_{0j^\star}|w_{j^\star}$, by Equation (66) the far child's key is, up to rounding, $\max(\mathrm{LB}(B), q_B + \Delta)$: it rises above $\mathrm{LB}(B)$ by $\max(0, q_B + \Delta - \mathrm{LB}(B))$, which is less than Δ when the parent's bound is stronger than its objective plane.

Reported bound and gap rule (GPU and host). A removed box holds no feasible point or has $\mathrm{LB}(B) \geq \theta$, where θ is the floor stored at its removal: the incumbent value $P' \geq P_t$ for dominance, and for the epsilon pruning of the evaluated configuration (the M3 reduction) the rounded-up threshold $P - \max(\epsilon_{\mathrm{abs}}, \epsilon_{\mathrm{rel}} \max(1, |P|))$. With $\theta_{\min}$ the smallest such θ, the run reports

$$D_t = \max\Big(D_{t-1},\ \min\big(\theta_{\min},\ \min_{B \in \mathcal{Q}_t} \mathrm{LB}(B)\big)\Big) \ \leq\ p^\star, \qquad p^\star \leq P_t, \tag{69}$$

(D_0 the root bound, $p^\star$ with the tolerance absorbed), which is a valid lower bound when every stored floor θ is. Under epsilon pruning the floor is below the incumbent, so this bound may be weaker than P_t even when the queue is empty. The cap matters: the box holding an optimum is removed once its bound reaches an exact incumbent, and the survivors' minimum may then exceed $p^\star$. The run stops when Equation (13) holds with $P - D \geq -10^{-5}$, when $\mathcal{Q} = \emptyset$ (then $P = p^\star$ if only exact dominance retirement was used, or infeasibility if $P = +\infty$; with tolerance-based retirement, queue exhaustion certifies the stated gap, not exact optimality), or at the deadline; a bound above P by more than 10^{-5} is an error and never certifies.

B.6 Primal Search

Acceptance gate (GPU, one point at a time). Every proposed point is evaluated on the compiled original model and accepted only if

$$\bar{x} \in B_0, \qquad \max_i [f_i(\bar{x})]_+ \leq 10^{-6}, \qquad \max_i [a_i^\top \bar{x} - b_i]_+ \leq 10^{-6} \tag{70}$$

over the nonlinear and linear rows (equalities as two rows), i.e. Equation (50) with $\tau_{\mathrm{feas}} = 10^{-6}$; then $P \leftarrow \min(P, f_0(\bar{x}))$. The in-tree ALM applies the same test to its batch residuals. Candidate generators enforce $\bar{x} \in B_0$ by construction (clamping or projection), and the routine checks the unscaled nonlinear and linear residuals in floating point. No merit or rescaled residual decides, so $P \geq p^\star$ up to the floating-point evaluation of f_0 and the rows (no outward-rounded objective evaluation).

Root ALM-PGD (GPU). At the root, ALM-PGD runs on the GPU, IPOPT in a CPU thread beside it, and LP alternation in a CPU process when the exact forms are a QCQP with a product term. The ALM minimizes Equation (11) over $R = 20{,}480$ starts on rescaled rows,

$$\phi_\rho(x; \mu) = f_0(x) + \frac{\rho}{2} \sum_i \Big[\frac{f_i(x)}{s_i} + \frac{\mu_i}{\rho}\Big]_+^2 - \frac{\|\mu\|^2}{2\rho}, \qquad s_i = \max\big(1, \mathrm{RMS}_r[f_i(x^{(r)})]_+\big),$$

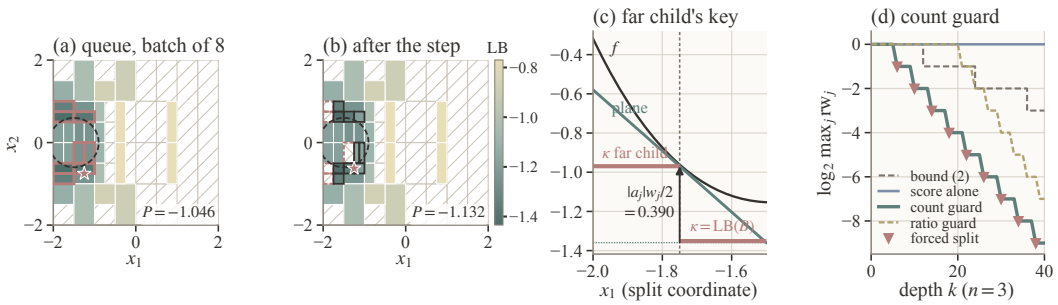

Figure 13: **One GPU branch-and-bound step on a toy**: $\min x_1^4/4 - x_1^2 + 0.3x_1 + x_2^2/2$ subject to $(x_1+1.5)^2 + x_2^2 \geq 0.36$ on $[-2,2]^2$, with lowered tangent planes in place of CROWN. (a) Queue at iteration 9, coloured by LB; the 8 lowest boxes form the batch (red); hatching marks regions removed by $\mathrm{LB} \geq P$ (/) or certified empty (×). (b) After the step: 7 boxes split (black), 4 removed by $\mathrm{LB} \geq P$ and 1 certified empty (dashed); $D = -1.434 \leq p^\star \approx -1.255 \leq P = -1.132$. (c) The far child's objective-plane minimum rises by $|a_j|w_j/2 = 0.390$ (Equation (66)), and its key by 0.382 above the parent's bound. (d) A path on which the score never picks x_3 ($n=3$, coefficients $(10^3, 10^3, 0)$): the count guard forces a widest split every fourth step and stays below Equation (67).

s_i halved toward 1 after each outer iteration, with $\mu \leftarrow \mu \odot s_{\mathrm{new}}/s_{\mathrm{old}}$ to keep raw multiplier units; the scales are computed separately for projected and unprojected root starts when the affine projection below is active. Inner step $t \leq T = 200$ of outer iteration $k = 0, 1, \ldots, 19$ is

$$x \leftarrow \Pi_{B_0}\Big(x - \mathrm{clip}\big(\eta_k(1 - 0.9\tfrac{t}{T})\max_j w_j\,\hat{d}_t,\ \pm\eta_k w\big)\Big), \qquad \eta_k = 0.01 \cdot 0.6^k,$$

with $\hat{d}_t = m_t/\big(\sqrt{v_t/(1-\beta_2^t)} + \epsilon\big)$ the Adam direction (the first moment is not bias-corrected, unlike Equation (48)); then $\mu \leftarrow [\mu + \rho f/s]_+$, and $\rho \leftarrow \min(2\rho, 10^6)$ ($\rho_0 = 1$) where the largest violation fell by less than 25%. Start 0 is the box centre and the rest are uniform, except a quarter scaled log-uniformly around $\Pi_{B_0}(0)$ and a quarter projected onto the affine equalities $Ex = d$ detected as opposite rows with constant Jacobians, $x \leftarrow \Pi_{B_0}(x - E^\top(EE^\top)^+(Ex - d))$, at the start and after every Adam step. The early-exit test $\|\nabla\phi_\rho\| \leq \max(\rho^{-0.8}, 10^{-5})$ synchronizes with the host at every inner step.

Restoration first (GPU, host control). Once IPOPT has a verified value P, progress is measured by

$$r(x) = \max\Big(\frac{v(x)}{10^{-6}},\ 1 + \frac{[f_0(x) - (P-\tau)]_+}{\tau}\Big) \geq 1, \qquad \tau = \max\big(10^{-5}, 10^{-4}\max(1, |P|)\big),$$

v the largest violation, so $r = 1$ at a feasible point better than P by the gap tolerance. The ALM runs its endgame (below) once and stops if the restored basin matches P within τ; otherwise it continues only while the measured decrease of $\log\min r$ per inner step can reach 0 within the remaining inner steps.

Gauss-Newton/Levenberg-Marquardt restoration (GPU). Domains without a point that passes the gate are restored from their archived point and least violated end points (256 seeds and 50 iterations at the root; 8 seeds per box, 30 iterations and 2 s in the tree). In unit-box coordinates z, with rows normalized as $t_i = (f_i + \frac{1}{2}\cdot 10^{-6})/\|\nabla_z f_i\|$, an approximate restoration step is computed (not the exact solution of a constrained trust-region subproblem): the damped Gauss-Newton direction

$$(J_{\mathcal{A}}^\top J_{\mathcal{A}} + \lambda I)\, d_{\mathrm{GN}} = -J_{\mathcal{A}}^\top t_{\mathcal{A}}, \qquad d = d_{\mathrm{GN}} \min\big(1, \Delta/\|d_{\mathrm{GN}}\|_\infty\big), \tag{71}$$

over the coordinates not held at a bound, with J the Jacobian of the normalized residuals with the row norms frozen at the linearization (norm 1 for zero-gradient rows), the active set $\mathcal{A}$ growing from $\{t_i > 0\}$ by the rows the step would violate (three rounds), λ from $10^{-6}\max\mathrm{diag}(J_{\mathcal{A}}^\top J_{\mathcal{A}})$ and Δ adapted to accepted steps, and the length the first $\alpha \in \{1, \ldots, 2^{-7}\}$ (one batch) that decreases $\sum_i([f_i]_+/\|\nabla f_i\|)^2$ or $\max_i [f_i]_+$. At the root, still infeasible domains then get 250 normalized descent steps on $\max_i [f_i]_+$.

SQP polish with an interior-point QP (GPU). ALM stops at the first feasible points, under a large penalty, so up to 32 distinct points (half the best feasible, half the best of the 5% least violated) and the first 32 starts are polished (50 iterations, 5 s; in the tree 8 points, 1 s) by solving the always feasible elastic QP

$$\min_{d,\,v} \nabla f_0^\top d + \tfrac{1}{2} d^\top H d + \nu\, \mathbf{1}^\top v \quad \text{s.t.} \quad f(z) - 0.9\cdot 10^{-6} + Jd \le v, \quad v \ge 0, \quad 0 \le z + d \le 1, \tag{72}$$

with H a Powell-damped BFGS Hessian of the Lagrangian. A Mehrotra interior-point method solves it with one $n \times n$ Cholesky factorization of $H + J^\top \operatorname{diag}(\sigma) J + \operatorname{diag}(\omega_{\text{lo}} + \omega_{\text{hi}})$ per iteration (v eliminated; σ, ω from the multiplier-to-slack ratios), at most 20 iterations replayed from a CUDA graph; steps pass an Armijo test on $f_0 + \nu \sum_i [f_i - 0.9 \cdot 10^{-6}]_+$ with one second-order correction. Aiming at $0.9 \cdot 10^{-6}$ uses the tolerance the lower bounds also allow.

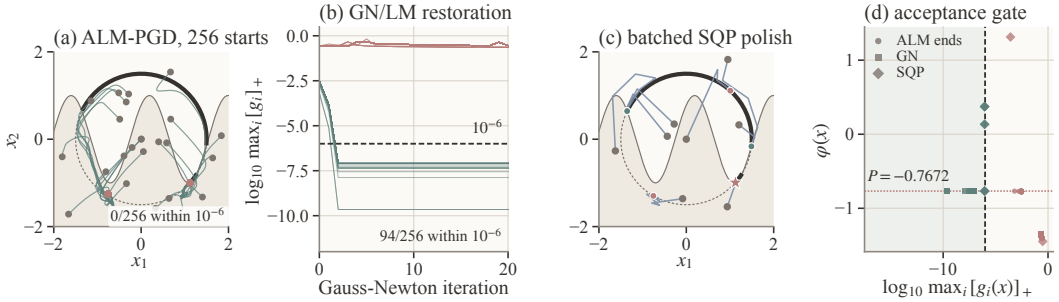

Figure 14: **Root primal pipeline on a toy**: $\min x_2 + 0.2x_1$ subject to $x_1^2 + x_2^2 = 2.25$ (two rows) and $x_2 \ge \sin(3.5x_1 + 0.9)$ on $[-2, 2]^2$ (feasible set: thick arcs; local minima -0.767, 0.134, 0.373). (a) ALM-PGD from 256 starts (20×200 steps; every 11th trajectory drawn): no end point is within 10^{-6}. (b) Restoration from 256 seeds: 94 reach 10^{-6}, most in two iterations; the others did not reach the feasible arcs. (c) SQP from 10 points: 8 pass the 10^{-6} test (the target is $0.9 \cdot 10^{-6}$). (d) The gate accepts 102 of the 522 points; $P \approx -0.767211$ for the tolerance-relaxed problem (the quoted local minima refer to zero-slack constraints).

IPOPT at the root (CPU). IPOPT (MUMPS) uses the exact sparse first and second derivatives of the structure walk (Appendix B.1), checked for non-finite values, and starts from the box centre and, while more than 20% of its effective budget (the shared root budget of 60 s, see Budgets) remains, from the minimizer of the root CROWN planes $\arg\min_{x \in B_0} \{a_0^\top x : a_i^\top x + b_i \le 0,\ i \ge 1\}$. It targets $f_i \le \frac{1}{2} \cdot 10^{-6}$, polishes an accepted point at $f_i \le 0.99 \cdot 10^{-6}$, and finishes a rejected one in the box at $f_i \le 0$.

LP alternation (CPU process). The blocks $x_{B_1}, \ldots, x_{B_K}$ are the colour classes of the product graph together with product-free variables, while variables with a unary nonlinear dependence (powers, squares, atoms) are held fixed, so the objective and every row are affine in one block when the others are fixed, and a block step is an LP solved on the CPU: from an infeasible point the elastic LP

$$\min_{x_{B_k},\, s^\pm \ge 0} \sum_{r:\|A_r\|_\infty > 0} \frac{s_r^+ + s_r^-}{\|A_r\|_\infty} \quad \text{s.t.} \quad c_L \le A\, x_{B_k} + c + s^+ - s^- \le c_U,$$

then the objective LP $\min\{\nabla_{B_k} f_0^\top x_{B_k} : c_L \le A x_{B_k} + c \le c_U\}$, which does not increase f_0 from a feasible current point (in exact arithmetic; numerically the acceptance rule rejects worse incumbents), with inconsistent constant rows handled separately. Blocks cycle while they progress; then an LP on all rows linearized at the point, in a trust region, moves all blocks and block LPs restore feasibility. Seeds are the McCormick LP point in every block order, ratio fits and McCormick vertices of perturbed costs; feasible points also seed sparse IPOPT. The LP steps tend to return basic solutions, i.e. vertices (Figure 15a), where degenerate network optima lie; every candidate still passes the feasibility gate.

Symmetric incumbent sorting (CPU). For each chain of interchangeable blocks (Appendix B.1), the root incumbent and warm start become $\operatorname{sort}(\bar{x})$, the blocks permuted so that $x_{B_1[1]} \le \cdots \le x_{B_m[1]}$; block swaps preserve f_0 and the multiset of rows (checked exactly), so the point keeps its value and satisfies the ordering rows used by clipping (Figure 15b).

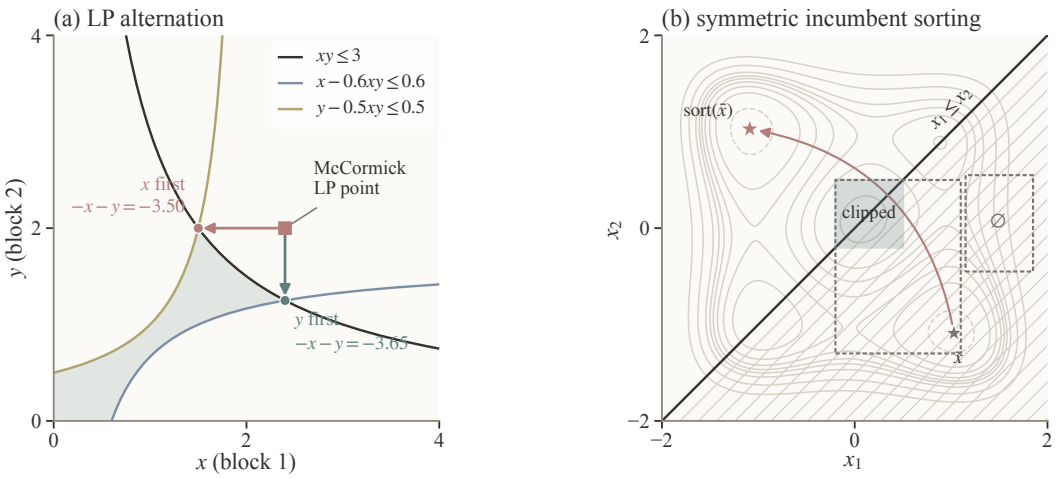

Figure 15: (a) **LP alternation** on $\min -x-y$ over $[0,4]^2$ with $xy \le 3$, $x-0.6xy \le 0.6$, $y-0.5xy \le 0.5$. From the same infeasible McCormick point $(2.4, 2)$, a run with x as the first block reaches the vertex $(1.5, 2)$ (-3.5) and a run with y first the global optimum $(2.4, 1.25)$ (-3.65); these are separate runs, and at both end points the trust-region LP predicts no decrease. (b) **Symmetric sorting**: the row $x_1 \le x_2$ removes the hatched half, $\bar{x}$ becomes $\mathrm{sort}(\bar{x})$ with the same value -0.540263, and clipping shrinks a box crossing the diagonal and empties one below it.

In the tree (GPU, host schedule). At scheduled iterations the ALM runs on every box of the batch before bounding, with about 1,000 starts in total, 20×200 steps, 10 s and the default parameters ($\eta_0 = 1$, $\rho_0 = 10$, gated multiplier updates, no row scaling). Start 0 of a box is the incumbent clamped into it (else the dual minimizer x^{best} of Equation (9)), $\mu^{(0)}$ is the batch mean of λ^{best}, and up to 10 inherited constraint planes that can be active join the rows. Restoration follows, plain and then with the steering row $f_0(x) \le P - 10^{-4}\max(1, |P|)$, and an SQP polish ends the call; points still pass the gate on the original rows. The period T starts at 5, doubles (up to 20) after three calls without improvement and resets on improvement; after four consecutive failures a call runs only if $(P - \min_{\mathcal{B}_t} \mathrm{LB})/(1 + |P|) < 0.01$. The lifted LP tree also proposes its leaf points (Appendix B.4).

B.7 Numerics

The numerical tensor kernels use double precision; symbolic coefficients use exact rationals and selected certificates use multiprecision interval arithmetic. Every LP-multiplier bound (the PDHG and CPU LP solver multipliers of the plane LP and of the lifted LP, the root dual LPs and the transferred certificates) and every LP infeasibility verdict is evaluated by one weak-duality routine with outward rounding, whereas interval propagation and constrained concretization use separate conservative tests. In that routine, under finite arithmetic and without underflow or overflow, library errors being assumed negligible: inner products carry the Higham bound γ_k of their summation error, a double-double evaluation is used when the error term could change a decision, and casts to lower precision round down. The lifted-LP rows carry outward cushions, and the posynomial and relative-entropy certificates are evaluated in interval arithmetic. The concretizations of the CROWN and interval bounds add a forward-error margin to their inner products and bias sums. The operator relaxations themselves, the separable floors and the projected Adam bound of the dual ascent are evaluated in ordinary double precision without outward rounding. Incumbents are verified on the original model; for scoring, each returned point is replayed on the exact double-precision model at 10^{-6}, and the 22 points of the instances that only GEAR certifies among GEAR, Gurobi and SCIP (Appendix D.2) were also re-evaluated in AMPL, with a largest violation of $9.9999994 \cdot 10^{-7}$.

B.8 Budgets

Several steps have wall-clock budgets: bound tightening (5 s on the McCormick LP and up to 10 s in total; in the tree, 5 s per call and 20 s in total), the LP branch-and-bound (30 s), the exact LPs of one GPU batch on the lifted relaxation (0.5 s), the quadratic-tangent KKT solve (10 s), each certificate LP

(20 s), the root ALM search (60 s) and IPOPT (30 s configured, raised to the shared 60 s root budget in parallel root mode, subject to the run deadline), the SQP polish (5 s at the root, 1 s in the tree), each in-tree ALM call (10 s) and its Gauss-Newton search (2 s). The solved set therefore depends on the speed of the host.

C Benchmark and Experimental Protocol

C.1 Source Pools

The 505-instance benchmark is built from two continuous NLP source pools: 265 `gams_world` instances and 240 `minlplib_linear` instances. GEAR and every baseline run on the same instance keys.

C.2 Canonicalization and Graph Construction

Each retained instance is converted to

$$\min_x f_0(x) \quad \text{s.t.} \quad f_i(x) \le 0,\ i = 1, \dots, m, \qquad x \in [\ell, u].$$

Maximization objectives are negated. Two-sided inequalities are split into paired one-sided inequalities. Equalities are encoded as paired inequalities with a fixed tolerance. Objective offsets, source variable bounds, and model scaling are preserved unless the source parser already normalizes them.

PyTorch graph. The graph uses one vector input $x \in \mathbb{R}^n$ and emits f_0 together with the shifted constraints, as in (2). Linear constraints are also graph outputs, reproduced exactly by LBP; when the parser exposes them as an explicit matrix, that matrix is also kept for clipping and the dual. Residual infeasibility uses individual constraint rows or their nonnegative mixtures and does not require a separate hard-max graph output. Shared algebraic subexpressions can be reused when exposed by the parser.

C.3 Inclusion and Exclusion Policy

An instance is retained only if it is continuous, has finite variable bounds after canonical preprocessing, uses supported graph primitives, has valid domain guards for singular operators, and can be parsed from its AMPL model into the GEAR graph representation.

Source / filter stage	Retained count
`gams_world` continuous computation-graph NLPs	265
`minlplib_linear` continuous computation-graph NLPs	240
Total	505

Table 9: Retained instances by source.

Exclusions. Excluded cases fall into structural categories: integer or discrete variables, missing finite bounds, unsupported operators, domain singularities that cannot be split or rewritten, parser failures, inconsistent source metadata, or root memory requirements beyond the configured GPU budget.

C.4 Translation Precision and Primal-Dual Soundness

GEAR parses each instance into a PyTorch computation graph and exact affine rows. Translation can introduce floating-point discrepancies relative to the original NLP model through evaluation order, intermediate precision, and transcendental implementations. These discrepancies must be accounted for before graph-level bounds are interpreted as original-model certificates.

On the primal side, the source-model validation protocol checks every accepted incumbent against the original constraints under their stated tolerances. The reported benchmark solutions were re-substituted into the original model and checked for feasibility with finite objective values. In

the notation of (50), those tolerances have already been absorbed into the constraint functions, so acceptance is against zero rather than an additional positive threshold. On the dual side, the conservative row and scalar evaluations described in Appendices A.3 and A.6 give original-model bounds only when their allowances also cover relevant translation and propagation errors. Under these conditions the corrected bounds err in the conservative direction.

C.5 Baseline Protocol

Every baseline in Section 4 is run on the same workstation and the same 505 instance keys as GEAR, with a 180 s limit.

Gurobi and Xpress parallelize the spatial branch-and-bound tree itself, so each instance is solved with 16 threads, one instance at a time.

SCIP is run in its concurrent mode with 16 threads. In this mode several differently configured solver instances race on the same problem and the first one to terminate determines the result, while each LP is solved single-threaded; instances are again processed one at a time.

The remaining solvers have no native parallel tree search on our benchmark. BARON's thread option parallelizes only its MIP subproblems, which do not arise in these purely continuous models. Couenne, IPOPT, SNOPT, MINOS, and LGO are single-threaded. LindoGlobal does provide a multi-threaded mode, but in our tests it returned objective values inconsistent with the returned points, so we run it single-threaded.

For all of these solvers we saturate the CPU at the level of throughput: 16 instances run concurrently, each with one thread, and each instance keeps its own 180 s wall-clock limit.

Knitro supports both layouts. On the tuning subset, parallel multistart with 16 threads and 16 concurrent single-thread runs with serial multistart solved the same instances, and we keep the latter.

For every baseline we select a single global configuration, without per-instance tuning. The candidates are drawn from each solver's documented options, covering presolve and bound-tightening intensity, cutting planes, branching, local-search heuristics, and, for the natively parallel solvers, the thread layouts 16×1, 8×2, and 4×4. We evaluate them on a stratified subset of 40 instances, half drawn from instances that are hard for at least one solver and half sampled uniformly at random, and keep the configuration with the most certified solutions. For SCIP we additionally keep powers of sums unexpanded (`expr/pow/expandmaxexponent` $= 1$); this setting was found on the full benchmark, where it certifies 8 more instances and loses none. All solvers use the same termination criteria: a feasibility tolerance of 10^{-6}, a relative optimality gap of 10^{-4}, and an absolute gap of 10^{-5}. Every returned point is re-evaluated by AMPL against all original constraints and variable bounds at 10^{-6}. A global solver is credited with an instance only if its own primal and dual bounds satisfy the gap criterion and no other run found a feasible point that contradicts its certificate. A certificate is also discarded when the solver itself flags it as unreliable (AMPL status `solved?`, e.g. BARON's "globality is not guaranteed"). A global solver's `infeasible` verdict counts as solved when the instance is confirmed infeasible: no run of any solver returned a point feasible at 10^{-6} and at least two global solvers report infeasibility within the limit (one instance, `mosel_start_folioqpgraph`, for which GEAR also certifies infeasibility). A local solver is credited only if its verified objective matches a bound certified by a global solver within the same tolerance.

C.6 GEAR Logs and Scoring

GEAR logs provide the last parsed primal incumbent P, valid lower bound D, final status, wall time, and node count. Missing final summary lines are counted as unsolved. A GEAR run is counted solved only when it satisfies the finite-gap test of Equation (13) or terminates after all active domains have been pruned. Baseline runs are scored as in Appendix C.5.

C.7 Released Tables

The release lists every instance with its source, name, numbers of variables and constraints, sense, and the functions it uses. For GEAR and each baseline it gives one row per instance with the final status, primal and dual values, and time; the baseline rows also give the largest constraint violation found when the returned point is re-evaluated in AMPL.

D Analysis of the Results and Limitations

D.1 Which Problems Benefit Most from the GPU

Problem class	n	GEAR	SCIP	Gurobi	Union	GEAR-only	Union-only
≤ 5 variables	183	**182**	171	177	182	0	0
6-20 variables	151	**145**	118	127	140	8	3
21-100 variables	80	66	58	66	**71**	3	8
101-1000 variables	65	28	17	29	**32**	5	9
> 1000 variables	26	0	0	2	**2**	0	2
box constraints only	95	**92**	81	81	89	3	0
1-20 constraints	235	**230**	197	217	226	6	2
> 20 constraints	175	99	86	103	**112**	7	20
transcendental terms	147	**131**	100	110	118	16	3
polynomial/rational only	358	290	264	291	**309**	0	19

Table 10: Solved instances by problem class. Union: the six global solvers. Transcendental: `exp`, `log`, `sqrt` or trigonometric terms.

Table 10 breaks the comparison of Section 4 down by problem class. Three patterns emerge.

GEAR leads on low- to medium-dimensional problems with few constraints and composite nonlinearities. On instances with at most 20 constraints, GEAR certifies 322/330 (97.6%), against 278 for SCIP and 298 for Gurobi. On instances with transcendental terms, it certifies 131/147 (89.1%), against 100 and 110; all 16 instances that GEAR solves and no global solver certifies belong to this class (e.g. `Paviani`, `hs101`, `hs102`, `hs110`, `schitt_s380`, `kriging_peaks`, `pricing050`, and `polygon25` to `100`, which a special-structure certificate closes at the root; Appendix B). On these problems, the difficulty is the nonconvexity of composed exponential, logarithmic and power terms over a small box. Factorable relaxations of such compositions tend to be loose, so a CPU spatial branch-and-bound can process many weak nodes and still end with a gap. GEAR closes most of these instances at the root (Appendices B.3 and D.2); on those that need a tree, it evaluates a clipped linear relaxation for many input domains in one GPU batch, which spreads the cost of a step over its boxes.

The GPU does not replace cutting planes on large structured models. On instances with more than 20 constraints, GEAR certifies 99/175 (56.6%) against 103 for Gurobi, and 20 of the 22 instances that only the global union solves fall into this class: pooling, wastewater and water-network models, and circle-packing (`kall`) models with tens to thousands of bilinear constraints. Their certificates rely on LP-based branch-and-bound with RLT and other cutting planes and on bound propagation at every node, and their dimension (up to 2,399 variables) makes splitting the input box expensive; no instance with more than 1,000 variables is solved by GEAR.

The benefit is coverage, not latency. On the 405 instances that both GEAR and at least one global solver certify, the fastest global solver needs a median of 0.03-0.06 s across size classes, against 4-29 s for GEAR, which is faster on only 8 of them. The GPU therefore extends the set of instances that can be certified within the budget; on instances that CPU solvers already close quickly, its fixed start-up cost dominates.

D.2 Where GEAR Wins and Where It Loses

We compare GEAR instance by instance with the two strongest baselines, Gurobi and SCIP (the run of Section 4). Of the 505 instances, 399 are certified by GEAR and by at least one of them, 22 only by GEAR, 18 only by Gurobi or SCIP, and 66 by none. For each of the 40 disagreements we read the logs of all three solvers and assign one cause (Table 11); the numbers below are read from those logs. "Required gap" is the tolerance of Equation (13).

Wins. Sixteen of the 22 wins come from a stronger bound. On the posynomial programs, the substitution $x = e^y$ makes the problem convex and GEAR's certificate is tight at the root, while the baselines relax each product of fractional powers separately and branch on the original variables. On the polygon instances, relaxing every term $r_i r_j \sin(\theta_i - \theta_j)$ separately leaves Gurobi with the

Cause	Instances	n	Evidence from the logs
GEAR *certifies, Gurobi and SCIP do not (22)*			
Posynomial objective: certificate in logarithmic coordinates	`hs101`, `hs102`, `cute_hs101`, `cute_hs102`, `schitt_s380`	5	baseline gaps end at 34 to $10^4\times$ the required gap after $8.7 \cdot 10^3$ to $2.7 \cdot 10^6$ nodes
Polygon area under a diameter bound: special-structure certificate	`polygon25`, `50`, `75`, `100`	4	baseline gaps end at $5.0{\cdot}10^3$ to $9.8{\cdot}10^3\times$ the required gap
Symmetric log-barrier objective: bound tightening collapses the box	`Paviani`, `hs110`, `cute_hs110`	3	baseline gaps end at $1.9{\cdot}10^3$ to $3.0{\cdot}10^3\times$ the required gap after $6.0{\cdot}10^4$ to $1.4{\cdot}10^6$ nodes
GEAR's own bounds where the baselines' relaxations stay loose	`pricing050`, `cute_spanhyd`, `cute_probpenl`, `Shekelfox10`	4	baseline gaps end at $1.7 \cdot 10^3$ to $10^4\times$ the required gap
A baseline's point fails the 10^{-6} recheck	`kriging_peaks-full050`, `100`, `200`, `ex8_4_4`, `elena_hs54`, `cute_avion2`	6	baseline violations $1.4 \cdot 10^{-6}$ to $1.7 \cdot 10^{-5}$, one of 0.16 (`elena_hs54`); GEAR's at most $9.9999994 \cdot 10^{-7}$
Gurobi or SCIP certifies, GEAR *does not (18)*			
Same root relaxation; the baselines' tree search is far faster	6 `wastewater`, 3 `waterund`, 5 `kall`, `pooling_digabel16`, `ex5_3_3`	16	Gurobi: $2.9 \cdot 10^3$ to $4.0 \cdot 10^4$ nodes/s; GEAR's CPU lifted-LP tree: 1.1-207 nodes/s (median 27)
No lifted LP applies; GEAR's box bounds stay loose	`eq6_1`	1	after $3.0 \cdot 10^6$ boxes $D = -165$ against $P = 670.69$; Gurobi closes in $8.3 \cdot 10^5$ nodes (70 s)
Beyond GEAR's lifted-LP size limit	`pooling_sppa9tp`	1	

Table 11: The 40 instances on which GEAR and the union of Gurobi and SCIP disagree, by cause.

trivial bound; GEAR recognizes the structure and verifies a bound with a dedicated certificate. On the symmetric log-barrier objectives, ordering the symmetric variables and tightening the bounds of the lifted relaxation against the incumbent collapses the box around the optimum. On the remaining six, a baseline's returned point violates a constraint by more than 10^{-6} when re-evaluated in AMPL: by at most $1.7 \cdot 10^{-5}$ on five, and by 0.16 on `elena_hs54` (Gurobi). GEAR solves its 22 wins in a median of 9.8 s (at most 114 s).

Losses. Sixteen of the 18 losses are polynomial QCQPs with many bilinear constraints: water and wastewater networks, pooling and circle packing. On each of them GEAR's lifted McCormick LP and Gurobi's root relaxation have the same or nearly the same value, so neither starts with a better relaxation; the difference is the tree search. Gurobi re-solves a warm-started LP with propagation and cuts at every node, at $2.9 \cdot 10^3$ to $4.0 \cdot 10^4$ nodes per second. GEAR's CPU lifted-LP tree processes 1.1-207 nodes per second, and after its open leaves are handed to the GPU, the GPU phase closes a median of 0.74% of the gap that remains (9 of 14 instances close less than 1%): the batched GPU bounds are weaker than the parent's exact LP. On `eq6_1` (square roots) no lifted LP applies, and the GPU bounds alone stay far from the incumbent. The baselines solve their 18 wins in a median of 1.2 s. This is the regime discussed in Appendix D.1, where per-node LP throughput, not batched bounding, decides the outcome.

D.3 Limitations

Scope. GEAR targets bounded continuous computation-graph NLPs with supported factorable operators and domain guards for singular primitives. It does not cover unbounded variables, unsupported operators, integer variables, or exact equality handling beyond tolerance-paired inequalities.

Large bilinear models. GEAR is weakest on large models with many bilinear constraints (Appendix D.2). On these models its lifted LP and Gurobi's root relaxation have the same or nearly the same value, its CPU tree processes far fewer nodes per second than Gurobi's, and the batched GPU bounds are weaker than the parent's exact LP.

Conditional convergence. The convergence result in Appendix A.13 is dimension-exponential and conditional. Theorem A.8 relies on Assumption A.7, including a δ-complete primal-recovery hypothesis on ALM-PGD and a small-scale infeasibility-pruning hypothesis. Neither hypothesis is needed for lower-bound or pruning soundness; both are needed for finite termination.

Operator domain restrictions. Fused division requires a denominator interval bounded away from zero. Non-integer powers require guarded nonnegative bases. Trigonometric relaxations require period localization before convex or concave envelopes are applied. Outside these domains, the implementation must split, rewrite, or reject the operation.

E More Related Works

Linear bound propagation and neural network verification. Linear bound propagation was introduced for ReLU networks by CROWN [70] and DeepPoly [53] and generalized to arbitrary computation graphs by auto_LiRPA [65]. α-CROWN [66] optimizes the slopes of the operator relaxations by gradient ascent, and β-CROWN [62] adds Lagrangian multipliers for split constraints inside bound propagation. GEAR differs in what it optimizes: it keeps the operator relaxations fixed (apart from the closed-form McCormick plane selection of Section 3.3) and optimizes only nonnegative multipliers λ on the constraints' linear bounds after propagation, so an ascent step costs $O(mn)$ per box instead of a new propagation. GCP-CROWN [69] and BICCOS [72] add cutting planes to the bound computation, and Clip-and-Verify [73] uses linear constraints to clip input and intermediate domains; GEAR adopts relaxed and complete clipping from the latter and replaces its coordinate descent by gradient steps when there are many constraints (Section 3.3). Other verifiers use MILP and SMT formulations [56, 30, 2], semidefinite relaxations [47, 18, 34, 17], and branch-and-bound over activations or inputs [13, 12, 20, 23, 51]. These methods certify a fixed property; none of them maintains a primal incumbent that moves the pruning threshold.

Relaxations of factorable programs. Factorable programming builds a relaxation of a nonlinear program from relaxations of its elementary operators [55, 36, 31]. McCormick envelopes [39] give the convex and concave envelopes of a bilinear term on a box, and RLT [50] multiplies constraints to tighten them further. An LP relaxation keeps all McCormick planes at once and couples them through auxiliary variables, while backward linear bound propagation uses one plane per term, chosen by the sign of its coefficient. GEAR's plane selection by concretized value and its direct quotient relaxation (Section 3.3) narrow this gap while keeping one linear bound per output.

Bound tightening and Lagrangian bounds. Optimization-based bound tightening solves one LP per variable bound; Gleixner et al. [26] derive cheaper Lagrangian variable bounds from LP duals and reuse them during the search. The Lagrangian bound of Equation (9) is the same weak-duality construction applied to linear bounds from propagation instead of an LP relaxation, clipping tightens variable and intermediate bounds without LP solves, and the infeasibility test of Equation (10) is a Farkas-type certificate over the same linear bounds.

Branch-and-bound search. Spatial branch-and-bound descends from the branch-and-bound method of Land and Doig [35]; Morrison et al. [41] survey strategies for searching, branching and pruning. Global NLP solvers usually branch on variables of nonconvex terms that are violated at the LP solution, while verification branch-and-bound scores splits by their estimated effect on the bound [12, 38]. λ-SB (Section 3.5) is closer to the latter: it weights the coefficient magnitudes of the linear bounds by box width and by the Lagrangian multipliers, without solving an LP.

Primal recovery and local NLP solvers. Augmented Lagrangian methods go back to Hestenes and Powell; we use the inequality form analyzed by Rockafellar [49, 48]. Local solvers such as IPOPT [59], SNOPT [25], MINOS [43], CONOPT [21] and Knitro [15] solve KKT systems with factorizations. GEAR's projected ALM uses only graph evaluations, automatic differentiation and Adam steps [32], so thousands of searches run in one batch, and the Lagrangian multipliers λ of the bound only initialize its own multipliers μ (Section 3.4).

Optimization over trained networks and GPU solvers. A related line of work optimizes over trained neural networks with mixed-integer formulations [24, 2, 29], bringing optimization techniques to verification; GEAR goes in the opposite direction and brings verification bounds to optimization. GPU LP solvers such as PDLP [3, 4] accelerate one large LP with first-order iterations, whereas GEAR parallelizes across many small bounding problems in the branch-and-bound tree.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and Section 1 state the bounded computation-graph NLP scope, the main contributions, and the comparison with six global solvers; Section 5 states the main limitations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Appendix D.3 discusses unsupported operators, bounded-domain scope, conditional convergence, and large bilinear models.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: Appendix A gives full proofs for the main theoretical claims.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: Section 4 specifies the 505 instances, solver criteria, hardware, time limit, and parsing policy; Appendix C gives canonicalization and filtering details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes].

Justification: The current draft describes the benchmark construction and audit fields and includes them in the supplementary files.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The experimental protocol, the GEAR and baseline configurations, time limits, and solved criteria are specified in Section 4 and Appendix C.5; dataset filtering is specified in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: The main experiment is a deterministic solver benchmark over the full retained instance set, so Figure 2 reports exact coverage counts rather than sampling-based error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: Section 4 reports the 180 s wall-clock limit, solver pairing, and the NVIDIA RTX 5090 GPU used for GEAR runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work is an algorithmic solver study using public benchmark instances and does not involve human subjects, private data, or deployed decision systems.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A].

    Justification: The contribution is a global optimization algorithm for computation-graph NLPs; the paper does not introduce a model or dataset with direct societal deployment claims.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release high-risk generative models, scraped datasets, or other assets requiring misuse safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The benchmark sources and solver families are cited, but the draft does not yet enumerate all source licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The paper introduces a 505-instance benchmark; Appendix C documents its construction, filters, and released tables.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve human subjects, so IRB or equivalent review is not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLMs are not an important, original, or non-standard component of the proposed solver methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.