CONFORMAL VARIABLE FIXING FOR MIXED-INTEGER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a method for accelerating repeated solves of mixed-integer optimization problems with fixed structure and varying data. A learned predictor maps problem data to binary variable assignments. We use conformal prediction to decide which assignments we can be confident to fix, leaving the remaining binary variables and all continuous variables to an optimization solver. This yields a reduced problem with fewer binary variables. We also introduce a nonconformity score that measures how far the predicted assignments must be relaxed for the reduced problem to contain a near-optimal solution. Under exact optimization, our method returns a feasible solution for every feasible instance. For calibration and test instances drawn independently from the same distribution, the returned solution is near-optimal with probability at least a user-specified level. Experiments on four benchmark families in control, robotics, and transportation show that the method can reduce solve times while maintaining high solution quality.

Keywords: Conformal prediction; mixed-integer optimization; learning to optimize; variable fixing; probabilistic guarantees.

1 INTRODUCTION

Mixed-integer optimization problems arise in many applications, including robotics (Shin et al., 2022; Shorinwa et al., 2024), transportation (Toth & Vigo, 2002; Bertsimas et al., 2019), power systems (Van Hentenryck, 2021; Sarkar et al., 2018), and control (Marcucci & Tedrake, 2021). Many applications require decisions in real time, yet mixed-integer programming is NP-hard (Garey & Johnson, 1979), and even modern solvers may be too slow for time-critical settings.

We are interested in repeatedly solving problems with the same structure but different parameter values, where the parameter could encode context or changing information. For example, a delivery company may solve routing problems repeatedly on the same road network as different demands for product deliveries arrive. In this context, we have access to a significant amount of data from previous parameter values, and this data can be used to train a learned optimizer. Learned optimizers have seen success in robotics (Cauligi et al., 2020; Srinivasan et al., 2021), control (Reiter et al., 2024; Russo et al., 2023), finance (Bertsimas & Stellato, 2022) and power systems (Park et al., 2024).

One direction uses learning to guide decisions within branch-and-bound (Khalil et al., 2016; Gupta et al., 2020; Lodi & Zarpellon, 2017; Bengio et al., 2021), including branching (Scavuzzo et al., 2024) and cutting-plane selection and generation (Deza & Khalil, 2023; Dragotto et al., 2023). These methods preserve the solver's optimality guarantees, but still require the search and certification performed by branch-and-bound. A second direction is to predict solutions or their key components directly (Bertsimas & Stellato, 2022). These predictions are fast but generally lack optimality guarantees. For mixed-integer model predictive control, Russo et al. (2023) obtain deterministic suboptimality certificates by solving a collection of linear programs online.

Probabilistic guarantees offer a tradeoff between computation and solution quality, such as returning an ϵ-optimal solution with probability at least $1-\alpha$. Conformal prediction provides a framework for calibrating such guarantees. Ovalle et al. (2025) use conformal prediction to provide probabilistic feasibility guarantees for optimization problems with learned constraints. Clarke & Stellato (2026)

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

learn stopping rules for branch-and-bound that retain ϵ-optimality with high probability. We give a detailed summary of related literature in Appendix A.

1.1 Contributions

We develop a framework for conformal variable fixing in parametric mixed-integer optimization. Conformal calibration determines which predicted binary assignments to fix to zero or one. The solver optimizes over the remaining binary variables and all continuous variables. Similarly to Ren et al. (2023), whose large language model planners ask for human help when in doubt, we leave uncertain binary assignments to the optimization solver. Our contributions are:

- *Calibration for variable fixing.* We introduce a nonconformity score measuring the relaxation of predicted assignments needed to retain an ϵ-optimal solution. Its calibration controls the probability of excluding every such solution.
- *Feasibility and optimality guarantees.* We prove that our framework returns a feasible solution for every feasible instance and an ϵ-optimal solution with probability at least $1-\alpha$. This guarantee is marginal over i.i.d. calibration and test instances, independent of the trained predictor.
- *Computational complexity.* We consider a collection of ways to fix binary variables so that every instance in a dataset retains an ϵ-optimal solution under at least one fixing. We prove that jointly minimizing the number of such fixings and the maximum number of variables left free is NP-hard.
- *Empirical evaluation.* Experiments on four benchmark families in control, robotics, and transportation show that our framework can achieve order-of-magnitude speedups while returning ϵ-optimal solutions with high probability.

2 Conformal prediction for learned mixed-integer optimizers

In this section, we formulate the parametric mixed-integer problem and describe how a learned predictor proposes binary variable assignments. We then use conformal calibration to select which assignments to fix and develop a repeated solve procedure with feasibility and probabilistic optimality guarantees.

2.1 Parametric mixed-integer programs

Let $\mathbf{Q}$ be a distribution supported on a set Θ. For problem data $\theta \in \Theta$ let $f_\theta : \mathbf{R}^n \to \mathbf{R}$ be an objective function. We make no assumptions on f_θ other than the existence of an algorithm to solve the optimization problem. In many examples we will work with linear MIPs where $f_\theta(x) = c(\theta)^T x$ for $c(\theta) \in \mathbf{R}^n$. For each $\theta \in \Theta$ let MIP_θ be the optimization problem

$$z^\star(\theta) = \begin{array}{ll} \text{minimize} & f_\theta(x) \\ \text{subject to} & x \in X(\theta), \end{array} \qquad (\mathrm{MIP}_\theta)$$

where $X(\theta) \subseteq \mathbf{R}^n$ is the feasible set, which is given by

$$X(\theta) = \{x \in \mathbf{R}^n \mid x \in \tilde{X}(\theta), \quad x_i \in \{0,1\}, \quad i = 1, \ldots, I\},$$

where $\tilde{X}(\theta) \subseteq \mathbf{R}^n$ is a convex set which may depend on θ. We assume that $\tilde{X}(\theta)$ includes the bounds $0 \le x_i \le 1$ for $i = 1, \ldots, I$, so that $\tilde{X}(\theta)$ is the continuous relaxation of the feasible set $X(\theta)$. Here I is the number of binary variables in the problem. We will often consider the feasible region $\tilde{X}(\theta) = \{x \in \mathbf{R}^n \mid A(\theta)x \le b(\theta)\}$ where $A(\theta) \in \mathbf{R}^{m \times n}$, $b(\theta) \in \mathbf{R}^m$, and m is the number of constraints. We consider repeated solves of (MIP_θ) for independent draws of $\theta \sim \mathbf{Q}$. We write MIP_Θ for the set of all optimization problems (MIP_θ), and $x^\star(\theta)$ for an optimal solution of (MIP_θ). When (MIP_θ) has several optimal solutions, $x^\star(\theta)$ may denote any. We write $x^\star_{1:I}(\theta) \in \{0,1\}^I$ for its binary part. We refer to the set of optimal solutions of (MIP_θ) as $X^\star(\theta)$ and the set of ϵ-optimal solutions as $X^\star_\epsilon(\theta)$, defined for $\epsilon \ge 0$ by

$$X^\star_\epsilon(\theta) = \{x \in X(\theta) \mid f_\theta(x) - z^\star(\theta) \le \epsilon\,(|z^\star(\theta)| + 1)\}\,.$$

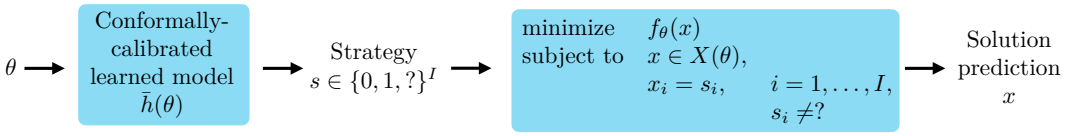

Figure 1: Overview of our method. A conformally calibrated predictor maps the context parameter θ to a strategy s, and a reduced problem is then solved in which the binary variables are fixed to the predicted values. The strategy may contain the symbol ?, which marks a binary variable the calibrated predictor leaves unassigned, to be found by the exact solver later.

Throughout, we assume that for every $\theta \in \Theta$ the feasible set $X(\theta)$ is nonempty, and the minimum in (MIP_θ) is attained. We also assume the user has access to a training dataset $\mathcal{D} = \{\theta_i\}_{i=1}^d$, a calibration dataset $\mathcal{C} = \{\theta_i\}_{i=1}^c$, and a test dataset $\mathcal{T} = \{\theta_i\}_{i=1}^l$ of parameter values drawn from $\mathbf{Q}$, and we assume for θ in each dataset, $z^\star(\theta)$ and $x^\star(\theta)$ have been computed and saved in advance.

2.2 Integer strategies

We learn a map from the context parameter θ to a *strategy*, following the terminology of Bertsimas & Stellato (2022). A strategy assigns each binary variable a value in $\{0, 1, ?\}$, where ? denotes an unassigned variable left to the solver. The notation ? represents the set $[0, 1]$. Our predictor can leave a binary variable unassigned when its value is uncertain. An illustration of our method is given in Figure 1.

Definition 2.1 (Strategy)**.** A strategy $s \in \{0, 1, ?\}^I$ is an assignment of integer variables in (MIP_θ) to members of $\{0, 1, ?\}$. We refer to the set of strategies for the parametric family (MIP_θ) as $\mathcal{S}$. If the number of entries of $s \in \mathcal{S}$ equal to ? is zero, we say s is a *pure integer strategy*. Otherwise, we say it is a *mixed strategy*. Given s, we write the set of indices of s which are ? as $\mathcal{F}(s)$, we write the number of ? entries in the strategy as $|s|_?$, and write $|\mathcal{S}|_? = \max_{s \in \mathcal{S}} |s|_?$.

A strategy $s \in \mathcal{S}$ for (MIP_θ) defines a restricted version of (MIP_θ)

$$\begin{aligned} z^\star(\theta, s) = \quad & \text{minimize} \quad && f_\theta(x) && (\text{MIP}_\theta(s)) \\ & \text{subject to} \quad && x \in X(\theta), \\ & && x_i = 0, \quad i = 1, \ldots, I \text{ with } s_i = 0, \\ & && x_i = 1, \quad i = 1, \ldots, I \text{ with } s_i = 1. \end{aligned}$$

The problem ($\text{MIP}_\theta(s)$) corresponds to the restricted optimizer of our method, which solves the small MIP referenced in Figure 1. If the unassigned set $\mathcal{F}(s)$ is small, solving ($\text{MIP}_\theta(s)$) is likely to be less expensive than solving (MIP_θ), because of the reduced number of integer variables. We therefore design a method which predicts a strategy s from parameter θ.

Definition 2.2 (Valid strategies)**.** We say that a strategy s is valid for a feasible point $x \in X(\theta)$ if $x_i = s_i$ for every $i = 1, \ldots, I$ such that $s_i \neq ?$.

It is helpful to also have a number representing the confidence we have in the value assigned by a strategy s to an index i.

Definition 2.3 (Confidence)**.** A *confidence* for a strategy s and context parameter θ is a vector $\kappa(s, \theta) \in [0, 1]^I$ where a higher value in index i means we are confident that the value s_i assigned by the strategy is likely to be correct for problem (MIP_θ) (with context parameter θ). A low value means that we are not confident, and we may need to relax it to solve the problem. We call $\kappa : \mathcal{S} \times \Theta \to [0, 1]^I$ a *confidence map*.

For the binary MIPs which we consider in this paper, a strategy and confidence can be defined together by a single vector and a single number. For $y \in [0, 1]$, we define $\lceil y \rfloor$ as the rounding of y, given by $\lceil y \rfloor = \mathbf{1}[y > 1/2]$.

Definition 2.4. A *soft strategy* is a vector $\sigma(\theta) \in [0, 1]^I$. It defines the pure integer strategy $s(\theta) = \lceil \sigma(\theta) \rfloor$ and confidence $\kappa(s, \theta) = 2|\sigma(\theta) - 1/2|$. A *soft strategy with rounding threshold* $(\sigma(\theta), \delta)$ for $\delta \in [0, 1/2]$ defines (elementwise, for $i = 1, \ldots, I$) the strategy

$$s_i(\theta) = \begin{cases} ? \text{ if } |\sigma_i(\theta) - 1/2| \leq \delta, \\ \lceil \sigma_i(\theta) \rfloor \text{ otherwise.} \end{cases}$$

2.2.1 FINDING INTEGER STRATEGY SETS

We first ask the question of whether it is possible to find a small set of strategies which cover our entire family of MIPs in the sense of the following definition.

Definition 2.5 (Covering)**.** We say that a strategy $s \in \mathcal{S}$ is ϵ-valid for (MIP_θ) if there is an ϵ-optimal solution $x^\epsilon \in X^\star_\epsilon(\theta)$ for (MIP_θ) such that $s_i = 1 \implies x^\epsilon_i = 1$, $s_i = 0 \implies x^\epsilon_i = 0$ for all $i \in \{1, \dots, I\}$. We say that a set of strategies $S \subseteq \mathcal{S}$ ϵ-covers the parametric family MIP_Θ if for every $\theta \in \Theta$ there is a strategy $s \in S$ such that s is ϵ-valid for (MIP_θ). We say that a set of strategies S ϵ-covers the dataset $\{\theta_j\}_{j=1}^d$ if for every $\theta \in \{\theta_j\}_{j=1}^d$ there is a strategy $s \in S$ such that s is ϵ-valid for (MIP_θ).

Clearly the strategy $\{?\}^I$ is 0-valid for (MIP_θ) for any $\theta \in \Theta$. Given a set of strategies S for (MIP_θ), we care in particular about three important numerical values: 1. The size of the strategy set $|S|$. 2. The infimum of the set of ϵ such that S is an ϵ-covering of MIP_Θ, which we write as $\epsilon^-(S)$. 3. The maximum number of entries equal to ? over all $s \in S$, which we write as $|S|_?$. If all three of these values are small for a given strategy set S, this means that the problem (MIP_θ) can be decomposed into a small number of subproblems covering an ϵ-optimal solution for each instance, meaning repeated parametric solves are likely to be easier. We are therefore interested in finding sets of strategies which simultaneously minimize $|S|$, $\epsilon^-(S)$, and $|S|_?$, subject to the constraint that S ϵ-covers MIP_Θ. We more frequently consider the data-driven problem for a given dataset $\mathcal{D} = \{\theta_1, \dots, \theta_d\}$, where we are searching for strategy sets S such that S ϵ-covers $\{(\text{MIP}_\theta) \mid \theta \in \mathcal{D}\}$. Minimizing any single term is easy, but minimizing any nontrivial, positive, nonzero weighted combination is NP-hard. We provide a proof of the following theorem in Appendix E.5.

Theorem 2.1. *Let $\alpha_1, \alpha_3 > 0$ and $\alpha_2 \geq 0$ be fixed rational constants. The problem of minimizing $\alpha_1|S| + \alpha_2\epsilon^-(S) + \alpha_3|S|_?$ subject to S being an ϵ-covering of $\{(\text{MIP}_\theta) \mid \theta \in \{\theta_1, \dots, \theta_d\}\}$ is NP-hard.*

2.3 PREDICTING INTEGER STRATEGIES

We learn a predictor $h : \Theta \to [0,1]^I$ that maps each context parameter to a soft strategy (Definition 2.4). We write h_i for the ith component of h. An output $h_i(\theta)$ near zero or one favors the corresponding assignment to x_i, while an output near $1/2$ indicates uncertainty.

We train h using the binary parts of the stored optimal solutions as targets, minimizing

$$\mathcal{L}(h) = \sum_{j=1}^{d} \ell\big(h(\theta_j), x^\star_{1:I}(\theta_j)\big),$$

where ℓ is the binary cross-entropy loss for a single training example, defined by

$$\ell(y_1, y_2) = -\sum_{i=1}^{I} \big[y_{2,i} \log y_{1,i} + (1 - y_{2,i}) \log(1 - y_{1,i})\big],$$

for a prediction $y_1 \in (0,1)^I$ and a target $y_2 \in \{0,1\}^I$. In our implementation, a sigmoid output layer keeps $h(\theta)$ in $(0,1)^I$, so the loss is well defined.

Rounding all entries of $h(\theta)$ gives a pure integer strategy. Even one incorrect binary assignment can make the restricted problem infeasible or exclude every ϵ-optimal solution. We therefore use a rounding threshold δ to leave uncertain variables unassigned, as in Definition 2.4. In Section 2.4, we calibrate this threshold so that the restricted problem retains an ϵ-optimal solution with probability at least $1 - \alpha$.

2.4 CONFORMAL CALIBRATION AND GUARANTEES

We use conformal prediction to choose a rounding threshold for the trained predictor h. This threshold determines which binary variables to fix and which to leave to the solver. Our goal is to retain an ϵ-optimal solution in the restricted problem with probability at least $1 - \alpha$.

In conformal prediction, a *nonconformity score* ϕ measures the quality of the prediction $h(\theta)$. A common choice is $\phi(\theta, h) = \|h(\theta) - x^\star_{1:I}(\theta)\|^2$ for a given norm (Angelopoulos & Bates, 2023). Our

nonconformity score is the smallest rounding threshold for which the restricted problem contains an ϵ-optimal solution. Define the relative suboptimality of a solution x by

$$r(x,\theta) = (f_\theta(x) - z^\star(\theta))/(|z^\star(\theta)| + 1). \tag{1}$$

The $+1$ in the denominator keeps the ratio well defined when $z^\star(\theta) = 0$, and is the same scaling used in the definition of $X_\epsilon^\star(\theta)$. For $\delta \in [0, 1/2]$ define the set-valued δ-rounding of a scalar $y \in [0,1]$ by

$$\lceil y \rfloor_\delta = \begin{cases} \{0\} & y < \frac{1}{2} - \delta \\ [0,1] & y \in [\frac{1}{2} - \delta, \frac{1}{2} + \delta] \\ \{1\} & y > \frac{1}{2} + \delta. \end{cases} \tag{2}$$

For the prediction $h(\theta) \in [0,1]^I$, define the rounding set in the full solution space by

$$\lceil h(\theta) \rfloor_\delta = \{x \in \mathbf{R}^n \mid x_i \in \lceil h_i(\theta) \rfloor_\delta, \quad i = 1, \ldots, I\}.$$

The continuous variables remain unrestricted by this rounding condition. We define the following nonconformity score

$$\phi(\theta, h) = \min\{\delta \in [0, 1/2] \mid X_\epsilon^\star(\theta) \cap \lceil h(\theta) \rfloor_\delta \neq \emptyset\}. \tag{3}$$

The score is the smallest threshold that leaves at least one ϵ-optimal solution compatible with the fixed assignments. A larger score requires leaving predictions farther from $1/2$ unassigned. At $\delta = 1/2$, all binary variables are unassigned, so the intersection in (3) is nonempty. The rounding set expands only at the finitely many thresholds $|h_i(\theta) - 1/2|$, where the corresponding variables become unassigned. Thus the minimum is attained at zero or one of these thresholds. We prove the following theorem in Appendix E.1, using a standard split-conformal argument (Shafer & Vovk, 2008; Gibbs et al., 2025).

Theorem 2.2. *Let $\theta_1, \ldots, \theta_c$ be i.i.d. draws from $\mathbf{Q}$, independent of h. For $i = 1, \ldots, c$, compute nonconformity scores $\phi_i = \phi(\theta_i, h)$ and let $\phi_{[1]} \leq \cdots \leq \phi_{[c]}$ denote their order statistics. Fix $\alpha \in [1/(c+1), 1)$ and choose $l = \lceil (1-\alpha)(c+1) \rceil$, which satisfies $l \leq c$, and set $\delta = \phi_{[l]}$. Let $\theta_{c+1} \sim \mathbf{Q}$ be independent of $\theta_1, \ldots, \theta_c$ and h. Then*

$$\mathbf{P}(X_\epsilon^\star(\theta_{c+1}) \cap \lceil h(\theta_{c+1}) \rfloor_\delta \neq \emptyset) \geq 1 - \alpha.$$

The probability is taken over the calibration sample and the independent test instance, with the trained predictor h held fixed.

Define the *prediction set* by $C(\theta_{c+1}) = \lceil h(\theta_{c+1}) \rfloor_{\phi_{[l]}}$. The theorem guarantees that the prediction set contains an ϵ-optimal solution with probability at least $1-\alpha$. On this event, solving the restricted problem to optimality returns an ϵ-optimal solution of the original problem. Using this theorem, we convert h from a predictor of soft strategies into a predictor of mixed strategies, by taking the rounding threshold to be $\delta = \phi_{[\lceil (1-\alpha)(c+1) \rceil]}$. This process is outlined in Algorithm 1.

The calibration score (3) can be computed for each instance by solving a single generic mixed-integer optimization problem. For a calibration instance θ_j, let $a_i = \max(0,\ 1/2 - h_i(\theta_j))$ and $b_i = \max(0,\ h_i(\theta_j) - 1/2)$. For each binary component x_i, the rounding condition $x_i \in \lceil h_i(\theta_j) \rfloor_\delta$ is equivalent to $\delta \geq a_i x_i + b_i(1 - x_i)$. The right-hand side is zero when x_i agrees with the rounded prediction and $|h_i(\theta_j) - 1/2|$ otherwise. Minimizing δ over the ϵ-optimal set therefore gives

$$\begin{array}{lll} \phi(\theta_j, h) = & \text{minimize} & \delta \\ & \text{subject to} & x \in X(\theta_j), \quad f_{\theta_j}(x) - z^\star(\theta_j) \leq \epsilon(|z^\star(\theta_j)| + 1), \\ & & \delta \geq a_i x_i + b_i(1 - x_i), \quad i = 1, \ldots, I, \\ & & 0 \leq \delta \leq 1/2, \end{array} \tag{4}$$

with variables $x \in \mathbf{R}^n$ and $\delta \in \mathbf{R}$. The score is the optimal *value* of (4), so it is a deterministic function of (θ_j, h). The optimal value $z^\star(\theta_j)$ is computed in advance and held fixed when computing the score. The process of computing the nonconformity score is outlined in Algorithm 1.

The calibration problems need not be solved to optimality. Any feasible pair (x, δ) gives an upper bound δ on the corresponding nonconformity score. Replacing the calibration scores with these upper bounds and sorting again cannot decrease the calibrated threshold. The resulting prediction sets contain those obtained using exact scores, so the conformal guarantee is preserved. In practice,

Algorithm 1 Conformal procedure

Require: $\alpha \in [1/(c+1), 1)$, $\epsilon \geq 0$, soft strategy predictor h, calibration set $\mathcal{C} = \{\theta_j\}_{j=1}^c$

Returns: Rounding threshold δ such that $\lceil h(\theta) \rfloor_\delta$ contains an ϵ-optimal solution with probability at least $1 - \alpha$

1: **for** $j = 1, \ldots, c$ **do**
2: Compute the nonconformity score $\phi_j \leftarrow \phi(\theta_j, h)$ by solving (4)
3: Sort scores to obtain order statistics $\phi_{[1]} \leq \cdots \leq \phi_{[c]}$
4: Choose $l = \lceil (1-\alpha)(c+1) \rceil$
5: **return** $\delta \leftarrow \phi_{[l]}$

we solve these problems heuristically. Using the calibrated threshold δ, we define the mixed strategy predictor $\bar{h}$ by

$$\bar{h}_i(\theta) = \begin{cases} 1 & h_i(\theta) > 1/2 + \delta \\ 0 & h_i(\theta) < 1/2 - \delta \\ ? & h_i(\theta) \in [1/2 - \delta,\ \frac{1}{2} + \delta], \end{cases}$$

for $i = 1, \ldots, I$. We prove the following corollary in Appendix E.2.

Corollary 2.2.1. *Let δ be the output of Algorithm 1 run on c i.i.d. draws $\mathcal{C} = \{\theta_j\}_{j=1}^c$ from $\mathbf{Q}$, independent of h. Let $\theta_{c+1} \sim \mathbf{Q}$ be an independent test instance. Then, with probability at least $1 - \alpha$ over the draws of $\mathcal{C}$ and θ_{c+1}, $X_\epsilon^\star(\theta_{c+1}) \cap \lceil h(\theta_{c+1}) \rfloor_\delta \neq \emptyset$. Equivalently, the calibrated strategy predictor $\bar{h}(\theta_{c+1})$ is consistent with some ϵ-optimal solution with probability at least $1 - \alpha$.*

Given a context parameter θ and a rounding threshold $\delta \in [0, 1/2]$, consider the restricted problem

$$\begin{array}{ll} \text{minimize} & f_\theta(x) \\ \text{subject to} & x \in X(\theta), \quad x \in \lceil h(\theta) \rfloor_\delta, \end{array} \tag{5}$$

with variable $x \in \mathbf{R}^n$. Here δ is the calibrated threshold from Algorithm 1. Problem (5) is equivalent to (MIP$_\theta(s)$) with strategy $\bar{h}(\theta)$, the mixed strategy obtained by applying rounding threshold δ to $h(\theta)$. When this problem is feasible, we denote an optimal solution by $\hat{x}(\theta, \delta)$. If the restricted problem is infeasible, we set $\hat{x}(\theta, \delta) = \emptyset$. If the problem (5) has multiple optima we allow $\hat{x}(\theta, \delta)$ to be any of them. We assume that, when feasible, problem (5) has an optimal solution, which is true whenever $\tilde{X}(\theta)$ is compact and f_θ is continuous, or when f_θ is linear and $\tilde{X}(\theta)$ is a rational polyhedron.

2.5 Ensuring feasibility

The restricted problem (5) may be infeasible because of the fixed binary assignments. To ensure feasibility, we repeatedly solve (5), increasing the rounding threshold whenever the restricted problem is infeasible. At $\delta = 1/2$, all binary variables are free and the restricted problem coincides with the original feasible problem. We calibrate the starting threshold for this procedure to retain the ϵ-optimality guarantee.

Fix an increasing sequence of threshold values in $[0, 1/2]$ given by $\bar{\delta}_1 < \cdots < \bar{\delta}_K$, with $\bar{\delta}_K = 1/2$. For $m = 1, \ldots, K$ let $L(\bar{\delta}_m) = \{\bar{\delta}_m, \bar{\delta}_{m+1}, \ldots, \bar{\delta}_K\}$ be the thresholds from $\bar{\delta}_m$ onwards. We write $\hat{x}_{\text{rep}}(\theta, \delta_0)$ for the solution returned by Algorithm 2. We choose the threshold sequence independently of the calibration and test instances and use the same sequence for all of them. For i.i.d. instances, the resulting scores are exchangeable, as required for the conformal guarantee. We give the proof of the following theorem in Appendix E.3.

Theorem 2.3. *For any $\delta_0 \in \{\bar{\delta}_1, \ldots, \bar{\delta}_K\}$, Algorithm 2 returns a solution feasible for (MIP$_\theta$). If $0 \leq \delta_1 \leq \delta_2 \leq 1/2$ and problem (5) is feasible at δ_1, then it is feasible at δ_2, and*

$$f_\theta(\hat{x}(\theta, \delta_2)) \leq f_\theta(\hat{x}(\theta, \delta_1)).$$

We now calibrate the starting threshold δ_0 using conformal prediction. For a calibration instance θ_j, define the *retry-aware nonconformity score*

$$\tilde{\delta}_j = \min\{\bar{\delta}_m \mid m = 1, \ldots, K, \quad \hat{x}_{\text{rep}}(\theta_j, \bar{\delta}_m) \in X_\epsilon^\star(\theta_j)\}. \tag{6}$$

Algorithm 2 Repeated solve algorithm

Require: Context parameter θ, starting threshold $\delta_0 \in \{\bar{\delta}_1, \ldots, \bar{\delta}_K\}$
Returns: The first feasible solution $\hat{x}(\theta, \delta)$ as δ ranges over $L(\delta_0)$ in increasing order

```
for δ ∈ L(δ_0) in increasing order do
    Solve (5) with threshold δ and set x ← x̂(θ, δ)
    if x ≠ ∅ then
        break
return x
```

The score $\tilde{\delta}_j$ is the smallest starting threshold in the sequence for which Algorithm 2 returns an ϵ-optimal solution. This minimum exists because the sequence is finite and starting at $\bar{\delta}_K = 1/2$ yields an optimal solution of the original problem. We prove the following corollary in Appendix E.4.

Corollary 2.3.1. *Let $\delta_0^\star$ be the output of Algorithm 3 run on c i.i.d. draws $\mathcal{C} = \{\theta_j\}_{j=1}^c$ from $\mathbf{Q}$, independent of h. Let $\theta_{c+1} \sim \mathbf{Q}$ be an independent test instance. Then $\mathbf{P}[\hat{x}_{\mathrm{rep}}(\theta_{c+1}, \delta_0^\star) \in X_\epsilon^\star(\theta_{c+1})] \geq 1 - \alpha$.*

The probability is taken over the calibration sample and the independent test instance, with the predictor and threshold sequence held fixed.

Algorithm 3 Retry-aware conformal calibration (RCP)

Require: $\alpha \in [1/(c+1), 1)$, $\epsilon \geq 0$, predictor h, calibration set $\mathcal{C} = \{\theta_j\}_{j=1}^c$, retry schedule $0 \leq \bar{\delta}_1 < \cdots < \bar{\delta}_K = 1/2$.
Returns: Starting threshold $\delta_0^\star$ such that $\hat{x}_{\mathrm{rep}}(\theta, \delta_0^\star) \in X_\epsilon^\star(\theta)$ with probability at least $1 - \alpha$.

```
for j = 1, ..., c do
    δ̃_j ← min{δ̄_m | m = 1, ..., K,  x̂_rep(θ_j, δ̄_m) ∈ X*_ε(θ_j)}.
Sort scores: δ̃_[1] ≤ ··· ≤ δ̃_[c].
Choose l = ⌈(1 − α)(c + 1)⌉.
return δ*_0 ← δ̃_[l].
```

In experiments we use the schedule $\{\bar{\delta}_1, \ldots, \bar{\delta}_K\} = \{1/2 - (1/2)^m \mid m = 1, 2, \ldots, 6\} \cup \{1/2 - 3 \cdot (1/2)^{m+2} \mid m = 1, 2, \ldots, 4\} \cup \{1/2\}$, so that $K = 11$.

An alternative always-feasible method. An alternative is to use the threshold from Algorithm 1 and solve the original problem whenever the restricted problem (5) is infeasible. This also ensures feasibility while retaining the probabilistic ϵ-optimality guarantee. By gradually relaxing the fixed assignments, our procedure can recover feasibility before all binary variables are free. We compare the solve times of these procedures in Section 3.

3 Experimental results

We test our methods on four benchmark problems from the literature. Detailed formulations and generation details are given in Appendix B. In the *obstacle avoidance* problem (Bertsimas & Stellato, 2022), a vehicle plans a path from a random initial position to a fixed goal while avoiding $n \in \{3, 5, 8, 10\}$ rectangular obstacles over $T = 60$ time steps, with big-M disjunctive constraints encoding obstacle avoidance. The family parameter is the initial vehicle position. In the *lane-change* problem (Frese & Beyerer, 2011), an autonomous vehicle overtakes $n \in \{2, 4, 6, 8, 12, 18\}$ moving cars while changing lanes over a discrete-time horizon. The family parameter encodes the agent initial state and the positions and velocities of the other cars. In the *helicopter path-planning* problem (Marcucci, 2025), a helicopter routes from a random start to a random destination through $n \in \{15, 30, 45, 60, 75\}$ islands subject to battery constraints. We formulate this as a shortest-path problem on a graph of convex sets (GCS). The family parameter is the pair of start and end positions. In the *school-bus routing* problem (Marcucci, 2025), a bus picks up a random subset of $n \in \{10, 12, 14, 16, 18\}$ children (each present with probability 0.8) and delivers them to school, jointly optimizing bus-stop locations and tour order via a GCS formulation of the traveling salesman

Table 1: Performance metrics (top) and baseline methods (bottom). $\tau_{\mathrm{GRB}}(\theta_i)$: solve time of GRB on instance θ_i. $\mathbf{1}[x(\theta_i) \neq \emptyset]$: 1 if the method returned a solution on θ_i, 0 otherwise; $\tilde{q} = \sum_i \mathbf{1}[x(\theta_i) \neq \emptyset]$. Mean suboptimality is averaged over instances where the method returns a solution. The mean is undefined when $\tilde{q} = 0$. Here $\mathrm{sgm}(\tau) = \exp\left(\frac{1}{q}\sum_{i=1}^{q} \log(\tau(\theta_i) + s)\right) - s$ is the shifted geometric mean of the solve times with shift $s = 1$ second.

Metric	Formula
Mean solve time	$(1/q)\sum_{i=1}^{q} \tau(\theta_i)$
Mean suboptimality %	$(100/\tilde{q})\sum_{\substack{1\leq i\leq q \\ x(\theta_i)\neq\emptyset}} r(x(\theta_i), \theta_i)$
Within ϵ-target (%)	$(100/q)\sum_{\substack{1\leq i\leq q \\ x(\theta_i)\neq\emptyset}} \mathbf{1}[r(x(\theta_i), \theta_i) \leq \epsilon]$
Infeasible (%)	$(100/q)\sum_{i=1}^{q} \mathbf{1}[x(\theta_i) \notin X(\theta_i)]$
Speedup (%)	$100\left(1 - \mathrm{sgm}(\tau)/\mathrm{sgm}(\tau_{\mathrm{GRB}})\right)$
Method	**Explanation**
GRB	Gurobi with relative MIP gap tolerance 0.01.
NEAR	Nearest neighbor: returns the stored solution for the training parameter closest to θ in ℓ_2 distance, projected onto $X(\theta)$ in ℓ_1 distance if necessary.
CONSTANT	FALLBACK with constant soft strategy predictor, namely the fraction of training optima in which each binary variable equals one (Appendix C).
FALLBACK	The conformal method, reoptimizing from scratch whenever it fails to return a feasible solution (end of Section 2.5).
GRB HEUR	Gurobi, terminating when the first feasible solution is found.

problem (TSP). The family parameter encodes the per-child presence pattern. Example solutions are shown in Figure 3, and binary variable counts for each problem are displayed in Figure 4.

Data. For each problem family we consider, we test on a handful of different problem sizes. We generate (using the loco-benchmarks Python package, see Appendix D) a train dataset of size $d = 7200$, $\mathcal{D} = \{\theta_i\}_{i=1}^{d}$, a calibration dataset of size $c = 800$, $\mathcal{C} = \{\theta_i\}_{i=1}^{c}$ and a test dataset of size $q = 1000$, $\mathcal{T} = \{\theta_i\}_{i=1}^{q}$. We discard infeasible instances without replacement. This means the actual datasets have slightly fewer instances than this, and dataset sizes are displayed in Table 2. The θ_i are i.i.d., and we explain what each θ represents for each problem below in Appendix B. All three datasets are therefore i.i.d. draws from $\mathbf{Q}$ conditioned on feasibility of the original problem. In data generation, we solve problems to the default 0.01% tolerance of Gurobi, with no time limit.

Methods and baselines. We evaluate CONFORMAL (Section 2.4) and RCP (retry-aware conformal prediction; Section 2.5). Both methods use a feedforward neural network h with three hidden layers of 512 units each. The hidden layers use ReLU activations. A sigmoid output layer ensures that $h(\theta) \in (0, 1)^I$. Table 1 defines the performance metrics and summarizes the baseline methods. We denote the reported solve time for instance θ_i by $\tau(\theta_i)$, measured in seconds. We sum the runtimes reported by Gurobi over all optimization solves. For learned methods, we also include neural network evaluation time.

Implementation details. We test our method using the Gurobi 13.0.1 (Gurobi Optimization, LLC, 2025) optimization solver. We use only a single thread. Presolve and other solve parameters are left untouched. During evaluation, we impose a 600-second solve time limit. None of the optimization solves performed by the learned methods reached this limit in our experiments. The theoretical ϵ-optimality guarantees for returned solutions assume that feasible restricted problems are solved to optimality. We set the suboptimality tolerance to $\epsilon = 0.01$ and the miscoverage level to $\alpha = 0.2$. Under the assumptions of Section 2, each method returns an ϵ-optimal solution with probability at least 0.8, over the calibration sample and an independent test instance. We train the networks in PyTorch (Paszke et al., 2019) using Adam (Kingma & Ba, 2015) to minimize the loss in Section 2.3. For each experiment, training runs for 2000 epochs on a single NVIDIA H100 GPU. Code and further experimental details are available in our public repository.

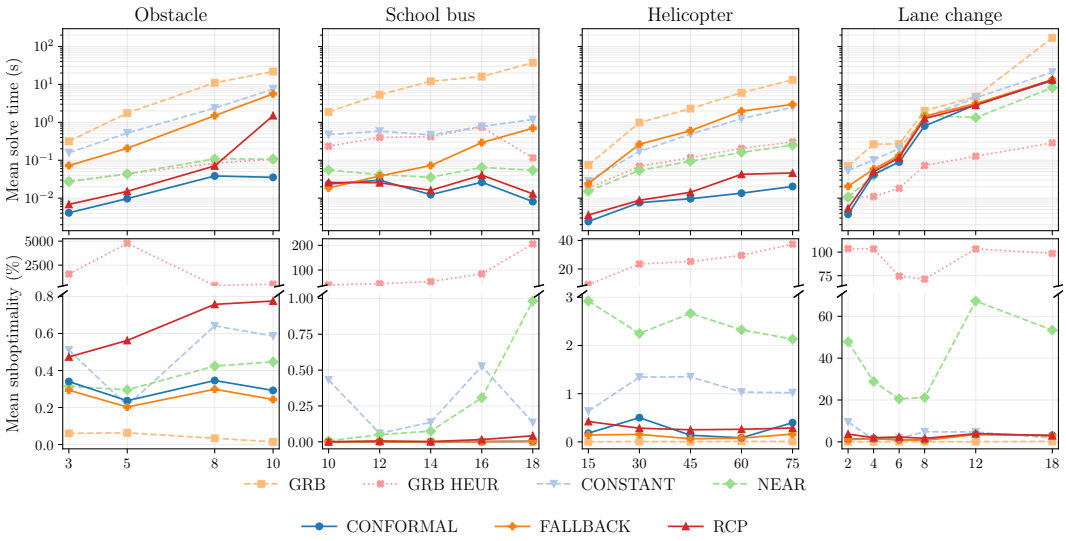

Figure 2: Results across all four benchmark problems at varying problem sizes. The problem size parameter is on the x axis: number of obstacles, children, islands, and cars respectively. Top row: mean solve time on a logarithmic scale. Bottom row: mean relative suboptimality, computed only over instances for which the method returned a solution (CONFORMAL returns none on up to 21% of instances; every other method always returns one). The vertical axis is broken so that the Gurobi heuristic, whose suboptimality is orders of magnitude larger, fits on the same panels.

Results. Results are displayed in Figure 2 and full numerical tables are in Appendix B. On obstacle avoidance, CONFORMAL achieves speedups of 98.5%–99.4% over GRB, with mean suboptimality below 0.4%. RCP achieves speedups of 97.7%–99.0%, with mean suboptimality below 0.8%. It returns a feasible solution on every test instance. On lane change, CONFORMAL and RCP achieve speedups of 19.1%–94.5% and 20.7%–92.3%, respectively. At $n = 18$, mean solve times are 169 seconds for GRB, 12.6 seconds for CONFORMAL, and 13.2 seconds for RCP. RCP returns a feasible solution on every test instance. On helicopter path planning, CONFORMAL and RCP achieve speedups of at least 95% over GRB at every problem size. RCP returns an ϵ-optimal solution on 93.7%–95.6% of test instances, with mean solve times of 3.6–46.1 milliseconds. On school-bus routing, CONFORMAL and RCP achieve speedups of at least 98.6% over GRB. At $n = 18$, mean solve times are 37.5 seconds for GRB, 8.1 milliseconds for CONFORMAL, and 13.0 milliseconds for RCP. RCP returns a feasible solution on every test instance, while CONFORMAL fails to return one on 1.5%–17.9% of instances across problem sizes. GRB HEUR has higher mean suboptimality than CONFORMAL and RCP at every tested problem size. On obstacle avoidance, NEAR returns an ϵ-optimal solution more often than either method. On the other three problem families, NEAR has higher mean suboptimality than both methods.

4 Conclusion

We presented a framework for accelerating parametric mixed-integer optimization using learned binary assignments and conformal calibration. Our nonconformity score identifies the smallest rounding threshold for which the restricted problem contains an ϵ-optimal solution. The calibrated repeated-solve procedure guarantees feasibility and returns an ϵ-optimal solution with probability at least $1 - \alpha$, under the stated assumptions. Experiments on four benchmark families show reductions in solve time relative to Gurobi.

AI USE STATEMENT

Large language models were used to edit prose for clarity and grammar, check bibliographic metadata against the original sources, assist with LaTeX formatting, support software development, and assist with experiment execution. They were not used to generate research ideas, interpret results, or write proofs. All technical content was checked by the authors. AI-assisted code was checked through manual review, AI-assisted review, tests, and comparisons against Gurobi. The authors are responsible for the final content.

ETHICS STATEMENT

This work proposes a method for accelerating mixed-integer optimization solvers and evaluates it on synthetic benchmark problems generated by our own code. The experiments involve no human subjects or personal data. The benchmark generators are described in Appendix B. Under the stated assumptions, the repeated-solve procedure guarantees feasibility and returns an ϵ-optimal solution with probability at least $1 - \alpha$. This probabilistic guarantee does not certify ϵ-optimality for each individual instance. Users deploying it in safety-critical settings should account for this probabilistic nature. We are not aware of any conflicts of interest.

REPRODUCIBILITY STATEMENT

The four benchmark families, their generators, the rejection rule for infeasible parameters, and all problem-size parameters are specified in Section 3 and Appendix B. Solver settings, the tolerance ϵ, the miscoverage level α, the data-split sizes, the network architecture, and the training budget are given in Section 3. All of our code is open-source, and we provide reproduction scripts for all experiments. The experiment code at `https://anonymous.4open.science/r/loco-2C47` and the LOCO package at `https://anonymous.4open.science/r/loco_benchmarks-4C21` are snapshots of commit `3e59245` of the experiment repository and commit `f9e0523` of the package repository, and the experiment requirements install the package from that snapshot. Proofs of all theorems and corollaries are provided in Appendix E.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023. doi: 10.1561/2200000101.

Davide Angioni, Claudia Archetti, and Maria Grazia Speranza. Neural combinatorial optimization: A tutorial. *Computers & Operations Research*, 182:107102, 2025.

Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290:405–421, 2021.

D. Bertsimas, P. Jaillet, and S. Martin. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, 67:143–162, 2019.

Dimitris Bertsimas and Bartolomeo Stellato. Online mixed-integer optimization in milliseconds. *INFORMS Journal on Computing*, 34(4):2229–2248, July 2022. ISSN 1526-5528. doi: 10.1287/ijoc.2022.1181.

Abhishek Cauligi, Preston Culbertson, Bartolomeo Stellato, Dimitris Bertsimas, Mac Schwager, and Marco Pavone. Learning mixed-integer convex optimization strategies for robot planning and control. *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1698–1705, 2020.

Wenbo Chen, Mathieu Tanneau, and Pascal Van Hentenryck. End-to-end feasible optimization proxies for large-scale economic dispatch. *IEEE Transactions on Power Systems*, 39(2):4723–4734, 2024. doi: 10.1109/TPWRS.2023.3317352.

Stefan Clarke and Bartolomeo Stellato. Conformal prediction for early stopping in mixed integer optimization. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Arnaud Deza and Elias B. Khalil. Machine learning for cutting planes in integer programming: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23), Survey Track*, pp. 6592–6600, 2023. doi: 10.24963/ijcai.2023/739.

Gabriele Dragotto, Stefan Clarke, Jaime Fernández Fisac, and Bartolomeo Stellato. Differentiable cutting-plane layers for mixed-integer linear optimization, 2023. arXiv preprint arXiv:2311.03350.

El Mehdi Er Raqabi and Pascal Van Hentenryck. Fusing backdoors, machine learning, and optimization for large-scale parametric mixed-integer programs, 2026. arXiv preprint arXiv:2606.21440.

Christian Frese and Jürgen Beyerer. A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1156–1162, 2011. doi: 10.1109/IVS.2011.5940489.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

M. Gasse, D. Chetelat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Isaac Gibbs, John J. Cherian, and Emmanuel J. Candès. Conformal prediction with conditional guarantees. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 87(4):1100–1126, 2025. doi: 10.1093/jrsssb/qkaf008.

Ambros M. Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff T. Linderoth, Marco E. Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13:443–490, 2021. doi: 10.1007/s12532-020-00194-3.

Prateek Gupta, Maxime Gasse, Elias B. Khalil, Pawan Kumar Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18087–18097, 2020.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, version 13.0, 2025. URL https://docs.gurobi.com/_/downloads/optimizer/en/13.0/pdf/.

Weimin Huang, Taoan Huang, Aaron M. Ferber, and Bistra Dilkina. Distributional MIPLIB: a multi-domain library for advancing ML-guided MILP methods, 2024. arXiv preprint arXiv:2406.06954.

Zeren Huang, Kerong Wang, Furui Liu, Hui-Ling Zhen, Weinan Zhang, Mingxuan Yuan, Jianye Hao, Yong Yu, and Jun Wang. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, 2022.

Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 724–731. AAAI Press, 2016. doi: 10.1609/aaai.v30i1.10080.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Chang Liu, Zhichen Dong, Haobo Ma, Weilin Luo, Xijun Li, Bowen Pang, Jia Zeng, and Junchi Yan. L2P-MIP: Learning to presolve for mixed integer programming. In *International Conference on Learning Representations*, 2024.

Andrea Lodi and Giulia Zarpellon. On learning and branching: a survey. *TOP*, 25(2):207–236, 2017.

T. Marcucci and R. Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 66(6):2433–2448, 2021. doi: 10.1109/TAC.2020.3007688.

Tobia Marcucci. A unified and scalable method for optimization over graphs of convex sets, 2025. arXiv preprint arXiv:2510.20184.

Tobia Marcucci, Jack Umenberger, Pablo A. Parrilo, and Russ Tedrake. Shortest paths in graphs of convex sets. *SIAM Journal on Optimization*, 34(1):507–532, 2024. doi: 10.1137/22M1523790.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks, 2020. arXiv preprint arXiv:2012.13349.

Daniel Ovalle, Lorenz T. Biegler, Ignacio E. Grossmann, Carl D. Laird, and Mateo Dulce Rubio. Conformal mixed-integer constraint learning with feasibility guarantees. In *Advances in Neural Information Processing Systems*, volume 38, pp. 89193–89229, 2025. doi: 10.52202/085713-2687.

Seonho Park, Wenbo Chen, Dahye Han, Mathieu Tanneau, and Pascal Van Hentenryck. Confidence-aware graph neural networks for learning reliability assessment commitments. *IEEE Transactions on Power Systems*, 39(2):3839–3850, 2024. doi: 10.1109/TPWRS.2023.3298735.

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*, volume 32, pp. 8024–8035. Curran Associates, Inc., 2019.

Krunal Patel. Progressively strengthening and tuning MIP solvers for reoptimization. *Mathematical Programming Computation*, 16:267–295, 2024. doi: 10.1007/s12532-024-00253-z.

Rudolf Reiter, Rien Quirynen, Moritz Diehl, and Stefano Di Cairano. Equivariant deep learning of mixed-integer optimal control solutions for vehicle decision making and motion planning. *IEEE Transactions on Control Systems Technology*, 33:1270–1284, 2024.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 661–682. PMLR, 2023.

Luigi Russo, Siddharth H. Nair, Luigi Glielmo, and Francesco Borrelli. Learning for online mixed-integer model predictive control with parametric optimality certificates. *IEEE Control Systems Letters*, 7:2215–2220, 2023.

M. N. I. Sarkar, Lasantha Gunaruwan Meegahapola, and Manoj Datta. Reactive power management in renewable rich power grids: A review of grid-codes, renewable generators, support devices, control strategies and optimization algorithms. *IEEE Access*, 6:41458–41489, 2018.

Lara Scavuzzo, Karen Aardal, Andrea Lodi, and Neil Yorke-Smith. Machine learning augmented branch and bound for mixed integer linear programming. *Mathematical Programming*, 2024. doi: 10.1007/s10107-024-02130-y.

G. Shafer and V. Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008.

Yunzhuang Shen, Yuan Sun, Andrew Craig Eberhard, and Xiaodong Li. Learning primal heuristics for mixed integer programs. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533651.

Young-Ha Shin, Seungwoo Hong, Sangyoung Woo, Jonghun Choe, Harim Son, Gijeong Kim, Joon-Ha Kim, Kang Kyu Lee, Jemin Hwangbo, and Hae-Won Park. Design of KAIST HOUND, a quadruped robot platform for fast and efficient locomotion with mixed-integer nonlinear optimization of a gear train. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 6614–6620, 2022. doi: 10.1109/ICRA46639.2022.9811755.

Olaoluwa Shorinwa, Trevor Halsted, Javier Yu, and Mac Schwager. Distributed optimization methods for multi-robot systems: Part 2—a survey. *IEEE Robotics & Automation Magazine*, 31(3): 154–169, 2024.

Mohit Srinivasan, Ankush Chakrabarty, Rien Quirynen, Nobuyuki Yoshikawa, Toshisada Mariyama, and Stefano Di Cairano. Fast multi-robot motion planning via imitation learning of mixed-integer programs. *IFAC-PapersOnLine*, 54(20):598–604, 2021. doi: 10.1016/j.ifacol.2021.11.237.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3706–3731, 2023. doi: 10.52202/075280-0164.

Paolo Toth and Daniele Vigo (eds.). *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM, 2002.

P. Van Hentenryck. Machine learning for optimal power flows. *INFORMS TutORials in Operations Research*, pp. 62–82, 2021. doi: 10.1287/educ.2021.0234.

Zhihai Wang, Xijun Li, Jie Wang, Yufei Kuang, Mingxuan Yuan, Jia Zeng, Yongdong Zhang, and Feng Wu. Learning cut selection for mixed-integer linear programming via hierarchical sequence model. In *International Conference on Learning Representations*, 2023.

Christopher Yeh, Nicolas Christianson, Alan Wu, Adam Wierman, and Yisong Yue. End-to-end conformal calibration for optimization under uncertainty. *Transactions on Machine Learning Research*, 2025.

Jiayi Zhang, Chang Liu, Xijun Li, Hui-Ling Zhen, Mingxuan Yuan, Yawen Li, and Junchi Yan. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519: 205–217, 2023. doi: 10.1016/j.neucom.2022.11.024.

A Related literature

Learning to accelerate mixed-integer optimization. Machine learning can guide decisions within mixed-integer optimization solvers, such as variable selection in branch-and-bound (Bengio et al., 2021; Zhang et al., 2023; Scavuzzo et al., 2024). Khalil et al. (2016) used imitation learning to approximate strong branching within a branch-and-bound solver. Gasse et al. (2019) learn branching policies using graph convolutional neural networks. The networks operate on a bipartite graph with nodes for variables and constraints. Lodi & Zarpellon (2017) survey learning to branch, and Bengio et al. (2021) give a methodological survey of machine learning for combinatorial optimization. Researchers have also applied learning to cutting planes, either selecting among the cuts a solver already generates (Wang et al., 2023; Huang et al., 2022) or generating new cuts directly (Dragotto et al., 2023). Deza & Khalil (2023) survey this line of work, covering cut selection by reinforcement and imitation learning, learned cut generation, and learning when to cut. Learning has also been used for solver configuration (Patel, 2024; Liu et al., 2024) and primal heuristics (Shen et al., 2021; Nair et al., 2020; Angioni et al., 2025). Another direction uses learned models to construct solutions. Sun & Yang (2023) combine diffusion models with decoding procedures tailored to each problem. For continuous economic dispatch, Chen et al. (2024) combine neural networks with differentiable repair layers to ensure feasibility. Learned variable fixing reduces a MIP's search space while leaving an optimization solver to complete the solution (Nair et al., 2020). Incorrect predictions can make the reduced problem infeasible or exclude all near-optimal solutions. Confidence estimates guide which predictions to enforce, and repair or relaxation procedures can restore feasibility or improve solution quality (Park et al., 2024; Er Raqabi & Van Hentenryck, 2026). Our method uses conformal calibration to choose which binary variables to fix and provides a marginal ϵ-optimality guarantee.

Conformal prediction. Conformal prediction constructs prediction sets with finite-sample marginal coverage guarantees under exchangeability (Angelopoulos & Bates, 2023). Ren et al. (2023) use conformal prediction to construct sets of candidate actions for an LLM planner. The planner requests human help unless its prediction set contains exactly one action. Similarly, our method uses conformal calibration to select which predicted binary assignments to enforce and leaves the remaining variables to an optimization solver. In robust optimization, conformal prediction calibrates uncertainty sets to provide marginal coverage guarantees (Yeh et al., 2025). For mixed-integer solvers, it can calibrate early stopping in branch-and-bound to provide probabilistic guarantees on suboptimality (Clarke & Stellato, 2026). Our method obtains such guarantees by calibrating which predicted variable assignments to enforce.

B Problem descriptions

This appendix presents the mixed-integer formulations and data generation procedures for the four benchmark families. Figure 3 illustrates example solutions, and Figure 4 shows how the number of binary variables varies with problem size. Table 2 reports the numbers of sampled, discarded, and retained instances and the sizes of the training, calibration, and test sets. The dataset and results tables are collected in Appendix F.

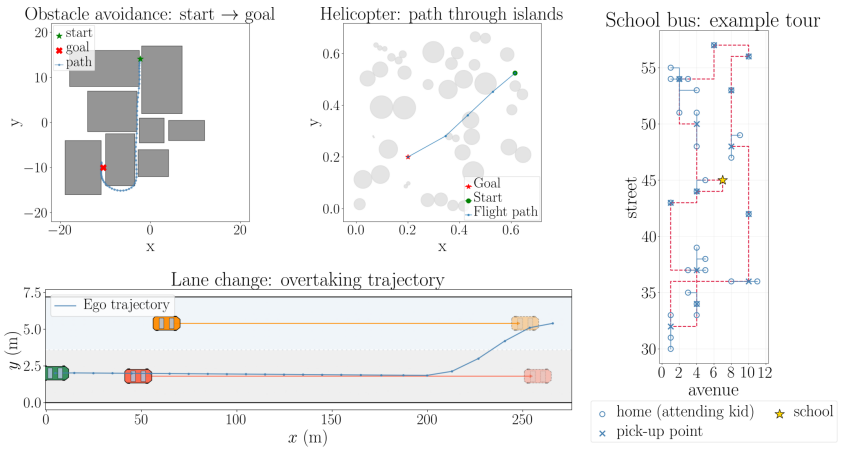

Figure 3: Example solutions to the four benchmark problems. The obstacle avoidance plot shows a trajectory (blue) from the start to the goal. The helicopter plot shows a route from the start to the goal, landing on islands where necessary to recharge. The lane change plot shows a trajectory taken by the agent to overtake both other cars. The school bus tour is shown by the red dashed line. Circles mark the homes of attending children, blue crosses mark pickup points, and the yellow star marks the school.

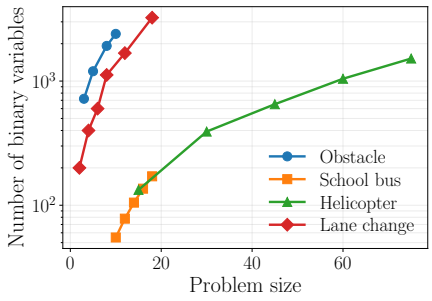

Figure 4: Number of binary variables versus problem size n. The size n counts obstacles for obstacle avoidance, islands for helicopter path planning, and other cars for lane change. In school bus routing, n counts all children, regardless of attendance.

B.1 Obstacle avoidance

We consider a vehicle moving among fixed rectangular obstacles over T discrete time steps. The parameter $\theta \in \mathbf{R}^2$ determines its initial position $p_0(\theta)$, as described below, and the objective penalizes distance to a fixed goal. We represent the trajectory by $T+1$ positions and enforce obstacle avoidance at each position after the initial one using big-M constraints.

Let $p_t \in \mathbf{R}^2$ and $v_t \in \mathbf{R}^2$ denote the position and velocity variables for $t = 0, \ldots, T$. The acceleration variables $a_t \in \mathbf{R}^2$ are defined for $t = 0, \ldots, T-1$. Obstacle $o = 1, \ldots, n$ is an axis-aligned box with lower and upper corners $l_o, u_o \in \mathbf{R}^2$. For each obstacle and time step $t = 1, \ldots, T$, we introduce binary vectors $z^-_{o,t}, z^+_{o,t} \in \{0,1\}^2$ to activate separation constraints along

each coordinate. We write $z_{o,t} = (z^-_{o,t}, z^+_{o,t}) \in \{0,1\}^4$. The optimization problem reads

$$\begin{aligned}
\text{minimize} \quad & \gamma \textstyle\sum_{t=0}^{T-1} \|a_t\|_2^2 + \sum_{t=0}^{T} \|p_t - p_{\text{goal}}\|_2^2 + \lambda \sum_{t=1}^{T-1} \sum_{o=1}^{n} \|z_{o,t+1} - z_{o,t}\|_1 && \text{(7a)}\\
\text{subject to} \quad & p_{t+1} = p_t + \Delta t\, v_t, \quad v_{t+1} = v_t + \Delta t\, a_t, \quad t = 0, \ldots, T-1, && \text{(7b)}\\
& \|v_t\|_\infty \le v_{\max}, \quad \|p_t\|_\infty \le p_{\max}, \quad t = 0, \ldots, T, && \text{(7c)}\\
& \|a_t\|_\infty \le a_{\max}, \quad t = 0, \ldots, T-1, && \text{(7d)}\\
& p_t \le l_o + M(\mathbf{1} - z^-_{o,t}), \quad p_t \ge u_o - M(\mathbf{1} - z^+_{o,t}), && \text{(7e)}\\
& \qquad o = 1, \ldots, n, \quad t = 1, \ldots, T,\\
& \mathbf{1}^T \left(z^-_{o,t} + z^+_{o,t}\right) \ge 1, \quad o = 1, \ldots, n, \quad t = 1, \ldots, T, && \text{(7f)}\\
& p_0 = p_0(\theta), \quad v_0 = 0, \quad z^-_{o,t}, z^+_{o,t} \in \{0,1\}^2. && \text{(7g)}
\end{aligned}$$

Here M is chosen large enough that a separation inequality is redundant when its indicator is zero. When an indicator equals one, (7e) enforces the corresponding coordinate bound. Constraint (7f) requires at least one indicator to equal one, so each trajectory point $p_1, \ldots, p_T$ lies outside the obstacle's interior. The remaining constraints impose the discrete-time dynamics, bounds on position, velocity, and acceleration, and the initial conditions $p_0 = p_0(\theta)$ and $v_0 = 0$. The objective penalizes acceleration, distance to the goal, and changes in the binary separation indicators. We use $\gamma = 0.1$ for acceleration and $\lambda = 0.01$ for indicator changes. The discretization time step is $\Delta t = 0.1\,\text{s}$. We add a tiny switching penalty to the obstacle-avoidance formulation of Bertsimas & Stellato (2022), given in (7a), to slightly simplify the classification problem by making it so that the obstacle indicators are somewhat consistent.

Problem generation. We use a horizon of $T = 60$ steps and consider $n \in \{3, 5, 8, 10\}$ obstacles. For each problem size, the obstacle positions are fixed across all instances. The specific obstacle map can be found in our public code, and is the same as the one used by Bertsimas & Stellato (2022). The velocity and acceleration bounds are $v_{\max} = a_{\max} = 10$. The maximum allowed position is taken to be $p_{\max} = 25$, and we take $M = 50$, which is twice the position bound $p_{\max}$ and therefore does not affect the problem. 9,000 problem instances are generated by drawing $\theta \in \mathbf{R}^2$ with independent standard logistic components. The initial position is $p_0(\theta) = \Pi\big(-18 + 36\ \text{sig}(\theta)\big)$, where sig is the componentwise logistic function and Π projects a point outside the obstacles when it lands inside one. Before projection, $-18 + 36\ \text{sig}(\theta)$ is uniformly distributed on $[-18, 18]^2$. After discarding infeasible instances, we use between 6,230 and 7,057 instances for training, between 692 and 784 for calibration, and between 866 and 981 for testing, depending on the size (Table 2).

Results. Table 3 reports the results. The conformal method has a mean solve time below 0.1 seconds at every size, with speedups between 98.5% and 99.4% and mean suboptimality below 0.4%. RCP returns a feasible solution on every test instance and achieves speedups between 97.7% and 99.0%. FALLBACK also returns a feasible solution on every test instance and attains ϵ-optimality on 92.2% to 93.9% of instances, but has a higher mean solve time than RCP at every size. NEAR attains ϵ-optimality on 89.3% to 95.2% of instances, exceeding the conformal method at every size, while achieving smaller speedups. The Gurobi heuristic is ϵ-optimal on at most 1% of instances.

B.2 Lane change

We consider a vehicle changing from the right lane to the left lane among n other cars over T discrete time steps. The objective penalizes deviation from a target velocity and acceleration, and rewards being ahead of the other cars. The mixed-integer linear formulation is adapted from Frese & Beyerer (2011).

Let $p_t \in \mathbf{R}^2$ and $v_t \in \mathbf{R}^2$ denote the agent position and velocity at step $t = 0, \ldots, T-1$, and let $a_t \in \mathbf{R}^2$ be the acceleration input for $t = 0, \ldots, T-2$. Coordinates 1 and 2 denote longitudinal and lateral position, respectively. Each car $i = 1, \ldots, n$ travels at a constant velocity $v^i(\theta)$ with $[v^i(\theta)]_2 = 0$. Its initial lateral position is the center $y_{\ell_i(\theta)}$ of its lane $\ell_i(\theta) \in \{0, 1\}$, and its position at step t is $p^i_t(\theta) = p^i_0(\theta) + t\,\Delta t\, v^i(\theta)$. For each car and time step, the binary vectors $z^-_{t,i}, z^+_{t,i} \in \{0,1\}^2$ activate separation constraints below and above the car in each coordinate. The optimization

problem reads

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{t=0}^{T-1} \left\|\omega_v(v_t - v^{\text{ref}})\right\|_1 + \sum_{t=0}^{T-2} \left\|\omega_a a_t\right\|_1 - w_{\text{ov}} \sum_{t=0}^{T-1} \sum_i [z_{t,i}^+]_1 && \text{(8a)}\\
& \quad - \lambda \textstyle\sum_{t=0}^{T-1} \sum_{i=1}^{n} \mathbf{1}^T \left(z_{t,i}^- + z_{t,i}^+\right) \\
\text{subject to} \quad & p_{t+1} = p_t + \Delta t\, v_t + \tfrac{\Delta t^2}{2}\, a_t, \qquad v_{t+1} = v_t + \Delta t\, a_t, \quad t = 0, \ldots, T-2, && \text{(8b)}\\
& p_t \le p_t^i(\theta) - h + M(\mathbf{1} - z_{t,i}^-), \quad t = 0, \ldots, T-1, \quad i = 1, \ldots, n, && \text{(8c)}\\
& p_t \ge p_t^i(\theta) + h - M(\mathbf{1} - z_{t,i}^+), \quad t = 0, \ldots, T-1, \quad i = 1, \ldots, n, && \text{(8d)}\\
& \mathbf{1}^T \left(z_{t,i}^- + z_{t,i}^+\right) \ge 1, \quad t = 0, \ldots, T-1, \quad i = 1, \ldots, n, && \text{(8e)}\\
& p_0 = p_0(\theta), \quad v_0 = v_0(\theta), && \text{(8f)}\\
& [p_{T-1}]_2 = y_{\text{left}}, \quad [v_{T-1}]_2 = 0, && \text{(8g)}\\
& \underline{v} \le v_t \le \overline{v}, \quad \underline{a} \le a_t \le \overline{a}, \quad -10 \le [p_t]_1 \le 600, \quad 1.0 \le [p_t]_2 \le 6.2, && \text{(8h)}\\
& z_{t,i}^-,\, z_{t,i}^+ \in \{0, 1\}^2. && \text{(8i)}
\end{aligned}$$

The products involving M, ω_v, and ω_a are elementwise, and $[\,\cdot\,]_j$ denotes the j-th entry of a vector. The position and velocity bounds apply for $t = 0, \ldots, T-1$, and the acceleration bounds apply for $t = 0, \ldots, T-2$. The vector data are the velocity target $v^{\text{ref}} = (v^\star, 0)$, the objective weights ω_v and ω_a, the separation half-widths h, and the big-M constants M, whose values are given below, together with the box bounds $\underline{v} = (0, -3.5)$, $\overline{v} = (40, 3.5)$, $\underline{a} = (-4, -2.5)$, and $\overline{a} = (3, 2.5)$. The optimization variables are the position, velocity and acceleration, as well as the binary indicators $z_{t,i}^-$ and $z_{t,i}^+$.

The target speed is $v^\star = 20\,\text{m/s}$, and the weights are $\omega_v = (0.4, 0.3)$ and $\omega_a = (0.08, 0.08)$. The overtake reward w_{ov} is chosen for each problem size, as specified below. The discretization time step is $\Delta t = 0.5\,\text{s}$. The half-widths are $[h]_1 = \frac{1}{2}(L_{\text{agent}} + L_{\text{car}}) + d_{\text{safe}} = 5.0\,\text{m}$ and $[h]_2 = \frac{1}{2}(W_{\text{agent}} + W_{\text{car}}) + d_{\text{safe}} = 2.4\,\text{m}$, using the agent dimensions $L_{\text{agent}} = 4.5\,\text{m}$, $W_{\text{agent}} = 1.8\,\text{m}$, obstacle-car dimensions $L_{\text{car}} = 4.5\,\text{m}$, $W_{\text{car}} = 2.0\,\text{m}$, and safety margin $d_{\text{safe}} = 0.5\,\text{m}$. The road consists of two lanes each of width 3.6 m (total width 7.2 m). The lateral bounds $[1.0,\ 6.2]$ of (8h) keep the agent within the road, accounting for the agent half-width (0.9 m) and a road margin of 0.1 m. The big-M constants are $[M]_1 = 800$ and $[M]_2 = 15$. Constraint (8b) is the double-integrator dynamics, (8c) and (8d) are the big-M separation constraints on the low and high side of each coordinate, and (8h) bounds the velocity, acceleration and position. The sum constraint (8e) requires the agent vehicle to be separated from car i by at least one half-plane at each step. The terminal constraints (8g) require the agent to finish centered in the target (left) lane y_{left} with zero lateral velocity, and (8f) fixes the initial state from the family parameter. The family parameter θ encodes the agent initial state $(p_0(\theta), v_0(\theta))$ and the obstacle configuration $(p_0^i(\theta), \ell_i(\theta), v^i(\theta))$ for each car i. The last term of (8a), with coefficient $\lambda = 10^{-3}$, rewards every satisfied separation indicator, so that the indicators are determined by the trajectory rather than chosen arbitrarily among the valid ones. When $w_{\text{ov}} > 0$, the objective rewards each time step spent ahead of each car by at least the longitudinal separation distance $[h]_1$. This favors overtaking early and staying ahead.

Problem generation. We generate 9,000 instances by sampling θ from a correlated parametric distribution (using the loco-benchmarks generator, see Appendix D). The generator's correlation parameter, which mixes each draw with a fixed base draw so that larger values make instances more alike, is 0.5 for $n = 2$ and 0.7 for the other sizes. The parameter θ determines the vehicle's initial state and the initial positions, lane assignments, and longitudinal velocities of the other cars. Each car's trajectory then follows from its constant velocity. After discarding infeasible instances, we use between 7,181 and 7,200 instances for training, between 797 and 800 for calibration, and between 998 and 1,000 for testing, depending on the size (Table 2). We report results for $n \in \{2, 4, 6, 8, 12, 18\}$ other cars. The horizon and overtake reward scale with the number of cars: $T = 25$ and $w_{\text{ov}} \in \{0,\ 2,\ 8\}$ for $n \in \{2, 4, 6\}$; $T = 35$ and $w_{\text{ov}} \in \{5,\ 12\}$ for $n \in \{8, 12\}$; $T = 45$ and $w_{\text{ov}} = 15$ for $n = 18$. For $n = 2$, $w_{\text{ov}} = 0$ activates a variant with a hard terminal constraint requiring the agent to have overtaken every car, $[p_{T-1}]_1 \ge [p_{T-1}^i(\theta)]_1 + [h]_1$ for $i = 1, \ldots, n$ (which would drive the problem infeasible for larger n, hence the switch to the soft reward-based formulation for $n \ge 4$). This yields $4nT$ binary variables per instance, ranging from 200 (for $n = 2$) to 3,240 (for $n = 18$).

Results. Table 4 reports the results. For $n \leq 8$, the conformal method achieves speedups between 51.4% and 94.5%, compared with 37.2% to 92.3% for RCP. Both RCP and FALLBACK return a feasible solution on every test instance. FALLBACK attains ϵ-optimality on 82.6% to 92.5% of instances, with a higher mean solve time than RCP at every size. At $n = 18$, Gurobi takes 168.6 seconds on average, compared with 12.6 seconds for the conformal method and 13.2 seconds for RCP. Their speedups at this size are 92.3% and 92.1%, respectively. For $n \geq 12$, NEAR is ϵ-optimal on only 2.7% to 3.8% of instances and has mean suboptimality above 50%. The Gurobi heuristic is ϵ-optimal on at most 4% of instances across all sizes.

B.3 Helicopter path planning

We consider a helicopter flying from a start position p_{start} to a destination p_{end} among n non-overlapping circular islands. It starts with a full battery and may land on intermediate islands to recharge. The battery depletes during flight at rate $d > 0$, and each flight must end before its charge becomes negative. Neither endpoint needs to lie on an island. The problem is adapted from Marcucci (2025). The parameter θ determines the two endpoints, as described below.

We formulate the problem as a shortest path on a *graph of convex sets* (GCS) (Marcucci et al., 2024), yielding a mixed-integer second-order cone program. Let $G = (V, E)$ be a directed graph whose vertices are the n islands, a source s, and a destination g. The edge set E contains the allowed flights between distinct vertices after the preprocessing described below. For each vertex $i \in V$, let $c_i \in \mathbf{R}^2$ be the set center and $r_i \geq 0$ its radius. The endpoints are represented by $c_s = p_{\text{start}}(\theta)$, $c_g = p_{\text{end}}(\theta)$, and $r_s = r_g = 0$. The flight speed is $v > 0$ and the charge rate is $k > 0$.

Let $A, B \in \{0,1\}^{|V|\times|E|}$ be the outgoing and incoming incidence matrices. For each edge $e = (i, j)$, column e of A has its only nonzero entry at row i, and column e of B has its only nonzero entry at row j. The submatrices $A_0, B_0 \in \{0,1\}^{n\times|E|}$ contain the island rows, and $e_s, e_g \in \mathbf{R}^{|V|}$ are the standard unit vectors corresponding to the endpoints.

We use the perspective (convex-hull) reformulation of Marcucci et al. (2024). The binary variables $u \in \{0,1\}^{|V|}$ indicate visited vertices, and $z \in \{0,1\}^{|E|}$ indicate traversed edges. For each edge $e = (i, j)$, the position copies $w_e, y_e \in \mathbf{R}^2$ represent z_e times the departure and arrival positions. We stack their transposes as the rows of $W, Y \in \mathbf{R}^{|E|\times 2}$. The vectors $b, a \in \mathbf{R}^{|E|}$ contain the corresponding battery levels on departure and arrival, also scaled by z_e. The vectors $t \in \mathbf{R}^{|E|}$ and $q \in \mathbf{R}^n$ contain the flight times and island recharge times, respectively. The entries of q follow the island ordering in A_0 and B_0. The optimization problem reads

$$
\begin{array}{ll}
\text{minimize} & \mathbf{1}^T q + \mathbf{1}^T t + \lambda \mathbf{1}^T z \\
\text{subject to} & Az = u - e_g, \quad Bz = u - e_s, \\
& u_s = u_g = 1, \\
& A_0 W = B_0 Y, \\
& A_0 b - B_0 a = kq, \quad e_s^T A b = 1, \\
& a = b - dt, \\
& 0 \leq a \leq z, \quad 0 \leq b \leq z, \\
& \|w_e - z_e c_i\|_2 \leq z_e r_i, \quad e = (i,j) \in E, \\
& \|y_e - z_e c_j\|_2 \leq z_e r_j, \quad e = (i,j) \in E, \\
& \|w_e - y_e\|_2 \leq v t_e, \quad e \in E, \\
& q \geq 0, \quad t \geq 0, \\
& u \in \{0,1\}^{|V|}, \quad z \in \{0,1\}^{|E|},
\end{array}
\tag{9}
$$

where the optimization variables are u, z, W, Y, a, b, t, and q, and the vector inequalities hold componentwise. The objective is the total recharge and flight time plus a penalty of $\lambda = 10^{-4}$ for each traversed edge.

The degree constraints give every visited island one incoming and one outgoing edge, with one departure from s, one arrival at g, and no edges entering s or leaving g. The position balance $A_0 W = B_0 Y$ makes the departure and arrival positions agree at each visited island. The battery balances account for discharge during flight and recharge on islands, and $e_s^T A b = 1$ requires departure from s with a full battery. The norm constraints keep both endpoints of each active edge in their respective

sets and bound its flight time below by distance divided by speed. Together with the battery bounds and discharge equation, they force the position copies, battery levels, and flight time to zero on inactive edges. The recharge balance then forces the recharge time to zero at unvisited islands. The degree constraints admit cycles disjoint from the path from s to g. Each cycle adds flight time, recharge time, and a positive edge penalty, so no optimal solution of the full problem contains one. If fixed binary variables force a cycle in a restricted problem, the returned point is evaluated as a feasible solution of (9), including the cycle's cost. The route is the component containing s and g.

We use flight speed $v = 1$, charge rate $k = 1$, and discharge rate $d = 6$.

Problem generation. The flight domain is the square $[0, 2/3]^2$. The parameter $\theta \in \mathbf{R}^4$ is what the predictor receives, and the endpoints are $p_{\text{start}}(\theta) = \frac{2}{3}\,\text{sig}(\theta_{1:2})$ and $p_{\text{end}}(\theta) = \frac{2}{3}\,\text{sig}(\theta_{3:4})$, where sig is the componentwise logistic function. The generator draws θ as the logit of a Gaussian-copula sample with correlation 0.5 around a base vector drawn once per family, so instances are i.i.d. conditional on that base vector, as the guarantees require, but the endpoint coordinates are not uniform on the domain. To keep the MIP tractable, we omit edges between two islands whose minimum distance, the center distance minus both radii, exceeds the maximum flight distance $v/d = 1/6$ (they are infeasible to activate given the battery, so optimality is preserved). Edges incident to the start or destination are always kept, so every instance has the same binary variables. Start and destination pairs with no battery-feasible route are discarded, as described in Section 3. We generate 9,000 instances per size and, after discarding infeasible instances, use between 6,951 and 7,200 for training, between 772 and 800 for calibration, and between 966 and 1,000 for testing, depending on the size (Table 2). We consider five sizes indexed by the number of islands $n \in \{15, 30, 45, 60, 75\}$. We fix a pattern of islands across all instances of a given size, sampled by generating island centers uniformly on the $[0, 2/3]^2$ map and radii uniformly in $[0.02, 0.1]$, then clipping each radius to half the distance to the nearest other center so that islands do not intersect. For full details we direct the reader to our public code.

Results. Table 5 reports the results. The conformal method and RCP achieve speedups of at least 95% at every size. RCP returns a feasible solution on every test instance and attains ϵ-optimality on 93.7% to 95.6% of instances, with mean solve times between 3.6 and 46.1 milliseconds. FALLBACK also returns a feasible solution on every test instance and attains ϵ-optimality on 96.6% to 98.6% of instances. Its speedups range from 68.4% to 93.5%, and its mean solve time exceeds that of RCP at every size.

B.4 School bus routing

We consider the school-bus routing problem of Marcucci (2025). A bus collects the attending children from a population of n children and delivers them to a fixed school. Each child may walk at most a distance $d_{\max} > 0$, measured in the ℓ_1 norm, from home to a pickup point. We jointly choose the pickup points and a tour through them and the school to minimize total bus travel and walking distance. The parameter θ encodes the attendance pattern. We formulate the problem as a traveling salesman problem on a graph of convex sets (Marcucci et al., 2024).

Let $G = (V, E)$ be a complete undirected graph with school vertex $s = 0$ and child vertices $1, \ldots, n$. We label each edge by its ordered endpoints, so $E = \{(i, j) \mid i, j \in V,\ i < j\}$. For each child i, let $c_i \in \mathbf{R}^2$ be their home position, and let $c_s \in \mathbf{R}^2$ be the school position. The attendance vector $p = p(\theta) \in \{0, 1\}^{|V|}$ is problem data, with $p_s = 1$ and $p_i = 1$ exactly when child i is present. Let $V_p = \{i \in V \setminus \{s\} \mid p_i = 1\}$ be the set of present children. The endpoint incidence matrices $A, B \in \{0, 1\}^{|V| \times |E|}$ use the same convention as in the helicopter formulation: column $e = (i, j)$ has its only nonzero entry at row i in A and row j in B.

For each child $i = 1, \ldots, n$, the allowed pickup positions and walking slacks form the bounded polyhedron

$$\mathcal{X}_i = \left\{(q, d) \in \mathbf{R}^2 \times \mathbf{R}^2 \;\middle|\; d \geq \pm(q - c_i), \quad \mathbf{1}^T d \leq d_{\max}\right\},$$

where the inequalities hold componentwise and $\pm$ denotes both signs. For the school, set $\mathcal{X}_s = \{(c_s, 0)\}$, fixing its position and giving it zero walking slack. For $a \geq 0$, we write $a\mathcal{X}_i = \{ax \mid x \in \mathcal{X}_i\}$; in particular, $0\mathcal{X}_i = \{0\}$.

We use the perspective construction of Marcucci et al. (2024). The binary variables $z \in \{0,1\}^{|E|}$ indicate traversed edges. For each vertex, the variables $q_i, d_i \in \mathbf{R}^2$ represent its pickup position and walking slack, with $d_s = 0$ for the school. For each edge $e = (i,j)$, the variables $w_e, y_e \in \mathbf{R}^2$ are its endpoint position copies, and $m_e, r_e \in \mathbf{R}^2$ are the corresponding walking-slack copies. These copies represent z_e times the vertex variables, and the school's walking-slack copies are zero. The variable $u_e \in \mathbf{R}^2$ bounds the absolute difference between the position copies in each coordinate. We stack the transposes of q_i and d_i as the rows of $Q, D \in \mathbf{R}^{|V|\times 2}$, and those of w_e, y_e, m_e, r_e, and u_e as the rows of $W, Y, M, R, U \in \mathbf{R}^{|E|\times 2}$, respectively. The optimization problem reads

$$
\begin{array}{ll}
\text{minimize} & \mathbf{1}^T U \mathbf{1} + \mathbf{1}^T D \mathbf{1} \\
\text{subject to} & (A+B)z = 2p, \\
& AW + BY = 2Q, \quad AM + BR = 2D, \\
& U \geq W - Y, \quad U \geq Y - W, \\
& (w_e, m_e) \in z_e \mathcal{X}_i, \quad e = (i,j) \in E, \\
& (y_e, r_e) \in z_e \mathcal{X}_j, \quad e = (i,j) \in E, \\
& (q_i - w_e, d_i - m_e) \in (p_i - z_e)\mathcal{X}_i, \quad e = (i,j) \in E, \\
& (q_j - y_e, d_j - r_e) \in (p_j - z_e)\mathcal{X}_j, \quad e = (i,j) \in E, \\
& \mathbf{1}^T z_{\partial S} \geq 2, \quad \emptyset \neq S \subseteq V_p, \\
& z \in \{0,1\}^{|E|},
\end{array}
\tag{10}
$$

where ∂S contains the edges with exactly one endpoint in S, and $z_{\partial S}$ is the corresponding subvector of z. All matrix inequalities hold componentwise.

The constraint $(A+B)z = 2p$ gives the school and each present child two incident edges and excludes absent children from the tour. The cut constraints eliminate disconnected subtours, so the active edges form a single tour through the school and the present children. These cuts are enforced lazily within a single branch-and-bound solve using solver callbacks. Restricting the cuts to subsets of V_p is sufficient because every edge incident to an absent child is zero.

On an inactive edge, the scaled-set constraints force the position and walking-slack copies to zero. On an active edge, both endpoints are present, and the complementary constraints force each copy to equal its vertex variable. The matrix balances link the two incident copies to each visited vertex, so the objective counts its walking slack once. For an absent child, all incident copies and its vertex variables are zero, and the complementary constraints reduce to membership in $0\mathcal{X}_i$. This explains the factors $p_i - z_e$ and $p_j - z_e$, which equal $1 - z_e$ at present vertices and zero at absent ones. At an optimum, $\mathbf{1}^T u_e = \|w_e - y_e\|_1$, so the edge cost equals the distance between its stops when active and is zero when inactive. Expanding the scaled-set constraints gives linear inequalities, and replacing the binary domain by $[0,1]^{|E|}$ gives a linear programming relaxation.

Problem generation. For each size, the home positions are drawn once, uniformly from the integer grid $[30, 60) \times [1, 12)$, and remain fixed across instances. The school is at $c_s = (45, 7)$, and the maximum walking distance is $d_{\max} = 3$. Before infeasible patterns are discarded, each child is present with marginal probability $p_{\text{kid}} = 0.8$. Presence indicators are obtained by thresholding correlated uniform draws at p_{kid}. With probability equal to the correlation parameter, all children share one uniform draw; otherwise, the draws are independent. Presence patterns with fewer than two present children, for which the degree constraints cannot be satisfied, are rejected and redrawn, so every generated instance is feasible; about 12% of raw draws are rejected, almost all with no child present. The implementation omits edges whose endpoint home-to-home ℓ_1 distance exceeds 100. Under the stated position bounds, this distance is at most 39, so the filter removes no edges. We consider five sizes indexed by $n \in \{10, 12, 14, 16, 18\}$ children. The model has $n(n+1)/2$ binary edge variables, giving counts of 55, 78, 105, 136, and 171, respectively. We generate 9,000 instances per size and use 7,200 for training, 800 for calibration, and 1,000 for testing (Table 2).

Results. Table 6 reports the results. The conformal method and RCP have mean solve times below 0.05 seconds at every size, with speedups ranging from 98.6% to over 99.9%. At $n = 18$, Gurobi takes 37.5 seconds on average, compared with 8.1 milliseconds for the conformal method and 13.0 milliseconds for RCP. RCP returns a feasible solution on every test instance. FALLBACK also returns a feasible solution on every test instance and attains ϵ-optimality on 99.8% to 100% of

instances, but has a higher mean solve time than RCP for $n \geq 12$. The Gurobi heuristic is ϵ-optimal on at most 1.8% of instances, with mean suboptimality between 39.6% and 205.9%.

C THE CONSTANT BASELINE

CONSTANT uses the fixing and fallback procedure of FALLBACK with a predictor that does not depend on θ. For each binary variable, its prediction is the fraction of training solutions in which that variable equals one:

$$\nu_i = \frac{1}{d} \sum_{j=1}^{d} x_i^\star(\theta_j), \qquad i = 1, \ldots, I.$$

We set $h(\theta) \equiv \nu$ and calibrate the rounding threshold using Algorithm 1, with $\epsilon = 0.01$ and $\alpha = 0.2$. The threshold is the $\lceil (1-\alpha)(c+1) \rceil$-th smallest calibration score, where c denotes the calibration-set size.

At test time, variables satisfying $|\nu_i - 1/2| > \delta$ are fixed to the rounded prediction, and the remaining binary variables are left free. If the restricted solve returns no feasible solution, CONSTANT solves the original problem. The reported solve time includes both solves when the fallback is used. Under exact solution of the subproblems, the fallback ensures feasibility as in Section 2.5. CONSTANT returned a feasible solution on every reported test instance. Its comparison with FALLBACK evaluates the effect of replacing the constant predictor with a predictor that depends on the instance.

D LOCO BENCHMARKS PACKAGE

LOCO (Learning of Combinatorial Optimization) provides configurable generators for parametric mixed-integer optimization problems. Users choose a problem family, its size, and the parameters of its sampling distribution. The package generates the benchmark families used in this paper and complements collections of benchmark instances (Gleixner et al., 2021) and problem distributions (Huang et al., 2024). For the conformal guarantees, calibration and test instances must be i.i.d. from a fixed family distribution and independent of the trained predictor. In every family the fresh randomness is independent across draws, and the training, calibration, and test sets are consecutive slices of one seeded stream, so all instances of a family are i.i.d. from the distribution conditional on the quantities fixed at construction: the obstacle map, the island map, the home positions, and, for lane change and helicopter, the base vector that the correlated sampler mixes with fresh noise. The package is available at

https://anonymous.4open.science/r/loco_benchmarks-4C21.

E PROOFS

E.1 THEOREM 2.2

Proof. Define $\phi_{c+1} = \phi(\theta_{c+1}, h)$. Since $\theta_1, \dots, \theta_{c+1}$ are i.i.d. and h is independent of all of them, the scores $\phi_1, \dots, \phi_{c+1}$ are exchangeable.

Let r denote the rank of ϕ_{c+1} among $\phi_1, \dots, \phi_{c+1}$, with ties broken uniformly at random. By exchangeability, r is uniformly distributed on $\{1, \dots, c+1\}$. If $\phi_{c+1} > \phi_{[l]}$, then at least l of the calibration scores are strictly smaller than ϕ_{c+1}, so $r > l$ under any tie-breaking. With $l = \lceil (1-\alpha)(c+1) \rceil$, we have $l \geq (1-\alpha)(c+1)$, so

$$\mathbf{P}\big(\phi_{c+1} > \phi_{[l]}\big) \;\leq\; \mathbf{P}(r > l) \;=\; \frac{c+1-l}{c+1} \;\leq\; \alpha.$$

Thus $\mathbf{P}(\phi_{c+1} \leq \delta) \geq 1-\alpha$. The rounding sets are nested in δ, and the minimum in (3) is attained. Hence $\phi(\theta, h) \leq \delta$ if and only if $X_\epsilon^\star(\theta) \cap \lceil h(\theta) \rfloor_\delta \neq \emptyset$, which gives the conclusion. ■

E.2 COROLLARY 2.2.1

Proof. For each j, Algorithm 1 computes the nonconformity score $\phi_j = \phi(\theta_j, h)$ by solving (4). It then selects $\delta = \phi_{[l]}$, with $l = \lceil (1-\alpha)(c+1) \rceil$. The claim follows directly from Theorem 2.2. ■

E.3 THEOREM 2.3

Proof. *Feasibility.* The set $L(\delta_0)$ contains $\bar{\delta}_K = 1/2$ as its final element. At this threshold, $|h_i(\theta) - 1/2| \leq 1/2$ for every i, so no binary variable is fixed. Problem (5) therefore coincides with (MIP$_\theta$). Since (MIP$_\theta$) is assumed feasible, $\hat{x}(\theta, 1/2) \neq \emptyset$. Algorithm 2 thus returns a feasible solution no later than this final step.

Monotonicity. Let $0 \leq \delta_1 \leq \delta_2 \leq 1/2$. Every assignment imposed at threshold δ_2 is also imposed at threshold δ_1, so

$$\lceil h(\theta) \rfloor_{\delta_1} \subseteq \lceil h(\theta) \rfloor_{\delta_2}.$$

Intersecting both sets with $X(\theta)$ preserves this inclusion. Thus feasibility at δ_1 implies feasibility at δ_2, and minimizing the same objective over the larger feasible set gives

$$f_\theta(\hat{x}(\theta, \delta_2)) \leq f_\theta(\hat{x}(\theta, \delta_1)).$$

■

E.4 COROLLARY 2.3.1

Proof. Algorithm 3 computes the scores $\tilde{\delta}_1, \dots, \tilde{\delta}_c$ of (6). Define $\tilde{\delta}_{c+1}$ by the same rule for the test instance. With the predictor and threshold sequence held fixed, these scores are exchangeable because the instances are i.i.d. and independent of both. The rank argument in the proof of Theorem 2.2

therefore gives

$$\mathbf{P}\big(\tilde{\delta}_{c+1} \le \delta_0^\star\big) \ge 1 - \alpha.$$

On the event $\tilde{\delta}_{c+1} \le \delta_0^\star$, let δ' be the first feasible threshold in $L(\tilde{\delta}_{c+1})$ and $\delta^\dagger$ the first feasible threshold in $L(\delta_0^\star)$. Both exist by Theorem 2.3. Since $L(\delta_0^\star) \subseteq L(\tilde{\delta}_{c+1})$, we have $\delta^\dagger \ge \delta'$. The definition of the score implies that the solution returned at δ' is ϵ-optimal. By the monotonicity in Theorem 2.3,

$$\begin{aligned} f_{\theta_{c+1}}(\hat{x}(\theta_{c+1}, \delta^\dagger)) &\le f_{\theta_{c+1}}(\hat{x}(\theta_{c+1}, \delta')) \\ &\le z^\star(\theta_{c+1}) + \epsilon\big(|z^\star(\theta_{c+1})| + 1\big). \end{aligned}$$

Thus the solution returned from starting threshold $\delta_0^\star$ is feasible and ϵ-optimal on an event of probability at least $1 - \alpha$. ■

E.5 Theorem 2.1

Proof. We reduce set cover to the problem in the theorem. Let $U = \{1, \dots, M\}$ and $A_1, \dots, A_N \subseteq U$. The set-cover decision problem asks whether at most K of these sets have union U. It suffices to consider nonempty U, $1 \le K \le N$, and $\bigcup_{j=1}^N A_j = U$, since all other cases can be decided directly. Choose

$$R = \left\lceil \frac{\alpha_1 N}{2\alpha_3} \right\rceil.$$

The weights α_1 and α_3 are fixed positive rational constants, so $R \le \alpha_1 N/(2\alpha_3) + 1 = O(N)$ as an integer and $2\alpha_3(R+1) > \alpha_1 N \ge \alpha_1 K$. For each $u \in U$, construct the binary optimization problem

$$\begin{aligned} \text{minimize} \quad & 1 \\ \text{subject to} \quad & \sum_{j : u \in A_j} x_{j,0} = 1, \\ & x_{j,0} = 0, \quad j \in \{1, \dots, N\} \text{ with } u \notin A_j, \\ & x_{j,r} = x_{j,0}, \quad j = 1, \dots, N, \quad r = 1, \dots, R, \end{aligned} \tag{11}$$

with variable $x \in \{0,1\}^{N \times (R+1)}$. Each problem has $N(R+1) = O(N^2)$ binary variables and is feasible because some A_j contains u. The entire family of M problems can therefore be constructed in polynomial time. For $j = 1, \dots, N$, define $\bar{x}^j \in \{0,1\}^{N \times (R+1)}$ by

$$\bar{x}^j_{k,r} = \begin{cases} 1 & k = j, \\ 0 & k \ne j, \end{cases} \qquad k = 1, \dots, N, \quad r = 0, \dots, R.$$

The feasible points of (11) are exactly the vectors $\bar{x}^j$ for which $u \in A_j$. All are optimal because the objective is constant. Let $B = \alpha_1 K$. We show that a set cover of size at most K exists if and only if this family admits a covering strategy set S with

$$\alpha_1 |S| + \alpha_2 \epsilon^-(S) + \alpha_3 |S|_? \le B.$$

Suppose first that $A_{j_1}, \dots, A_{j_k}$ cover U, with $k \le K$. Choose the pure strategies $S = \{\bar{x}^{j_1}, \dots, \bar{x}^{j_k}\}$. Every problem has an optimal solution in S, so $\epsilon^-(S) = |S|_? = 0$. The strategy objective is therefore $\alpha_1 |S| \le \alpha_1 K = B$.

Conversely, suppose that S covers the family and has strategy objective at most B. All three terms are nonnegative and $\alpha_1, \alpha_3 > 0$, so $|S| \le K$ and $|S|_? \le B/\alpha_3$. Any two distinct vectors $\bar{x}^i$ and $\bar{x}^j$ differ in exactly $2(R+1)$ binary coordinates. A strategy valid for both must leave all of those coordinates unassigned, which would imply

$$\alpha_3 |S|_? \ge 2\alpha_3 (R+1) > B.$$

Thus each strategy in S is valid for at most one feasible vector in the entire family. Remove strategies valid for no feasible vector and replace every remaining strategy by its unique compatible feasible vector. These replacements preserve the covering and cannot increase its cardinality, its suboptimality, or its maximum number of unassigned variables. The resulting pure strategies correspond to at most K sets A_j. Since every problem is still covered and $\bar{x}^j$ is feasible for parameter u exactly when $u \in A_j$, these sets cover U. This proves the reduction. ■

F EXPERIMENTAL TABLES

Table 2 reports dataset sizes. Tables 3–6 give the detailed results for the benchmark families described in Appendix B.

Table 2: Dataset sizes for each problem family and size n. Drawn: parameters sampled from the generator. Discarded: infeasible instances, removed without replacement. Retained: instances kept. Train, Calibration, Test: sizes of the three splits of the retained instances.

Family	n	Drawn	Discarded	Retained	Train	Calibration	Test
Obstacle avoidance	3	9000	178	8822	7057	784	981
	5	9000	870	8130	6504	722	904
	8	9000	1058	7942	6354	705	883
	10	9000	1212	7788	6230	692	866
School bus	10	9000	0	9000	7200	800	1000
	12	9000	0	9000	7200	800	1000
	14	9000	0	9000	7200	800	1000
	16	9000	0	9000	7200	800	1000
	18	9000	0	9000	7200	800	1000
Helicopter	15	9000	311	8689	6951	772	966
	30	9000	0	9000	7200	800	1000
	45	9000	90	8910	7128	791	991
	60	9000	0	9000	7200	800	1000
	75	9000	0	9000	7200	800	1000
Lane change	2	9000	16	8984	7187	798	999
	4	9000	0	9000	7200	800	1000
	6	9000	0	9000	7200	800	1000
	8	9000	0	9000	7200	800	1000
	12	9000	0	9000	7200	800	1000
	18	9000	24	8976	7181	797	998

Table 3: Obstacle avoidance results. Time: mean solve time (s), with one standard deviation in parentheses. Subopt: mean relative suboptimality (%), with one standard deviation in parentheses. ϵ-opt: % of test instances solved to ϵ-optimality. Infeas: % of test instances for which the method fails to return a feasible solution. Speedup: the shifted geometric mean speedup relative to GRB, as defined in Table 1. Shaded rows: our methods.

n	Method	Time (s)	Subopt (%)	ϵ-opt (%)	Infeas (%)	Speedup (%)
3	GRB	0.315 (±0.234)	0.06 (±0.128)	100	0	0
	GRB HEUR	0.028 (±0.002)	1,575.35 (±13,398.7)	0.5	0	90.5
	CONFORMAL	0.004 (±0.002)	0.34 (±1.186)	78.5	13.7	98.5
	FALLBACK	0.072 (±0.198)	0.294 (±1.108)	92.2	0	91
	RCP	0.007 (±0.017)	0.473 (±1.565)	88.5	0	98
	CONSTANT	0.157 (±0.228)	0.51 (±3.389)	95.4	0	51.9
	NEAR	0.028 (±0.005)	0.314 (±3.844)	95.2	0	90.6
5	GRB	1.759 (±2.003)	0.063 (±0.146)	100	0	0
	GRB HEUR	0.044 (±0.005)	4,773.57 (±48,944.6)	1	0	96.6
	CONFORMAL	0.01 (±0.008)	0.237 (±0.877)	79.2	14.4	99.3
	FALLBACK	0.206 (±0.655)	0.203 (±0.816)	93.6	0	90.6
	RCP	0.015 (±0.062)	0.562 (±1.725)	86.5	0	99
	CONSTANT	0.522 (±0.986)	0.21 (±1.695)	97.6	0	72.6
	NEAR	0.044 (±0.015)	0.296 (±1.125)	92.3	0	96.8
8	GRB	11.031 (±19.034)	0.034 (±0.106)	100	0	0
	GRB HEUR	0.084 (±0.112)	379.2 (±2,893.74)	0.9	0	97.9
	CONFORMAL	0.039 (±0.052)	0.346 (±1.538)	79.4	13.6	99
	FALLBACK	1.502 (±6.544)	0.299 (±1.435)	93	0	92.2
	RCP	0.071 (±0.72)	0.758 (±2.227)	83.4	0	99
	CONSTANT	2.378 (±9.019)	0.641 (±4.069)	96.1	0	75.6
	NEAR	0.109 (±0.075)	0.424 (±1.405)	89.4	0	97.2
10	GRB	21.854 (±42.044)	0.015 (±0.067)	100	0	0
	GRB HEUR	0.107 (±0.123)	524.189 (±4,558.26)	0.5	0	98.3
	CONFORMAL	0.035 (±0.029)	0.292 (±1.145)	77.1	16.7	99.4
	FALLBACK	5.648 (±24.61)	0.243 (±1.051)	93.9	0	90.9
	RCP	1.499 (±13.103)	0.775 (±1.975)	81.8	0	97.7
	CONSTANT	7.493 (±29.434)	0.586 (±3.275)	95.7	0	76.3
	NEAR	0.107 (±0.102)	0.447 (±1.69)	89.3	0	98.3

Table 4: Lane-change results. Time: mean solve time (s), with one standard deviation in parentheses. Subopt: mean relative suboptimality (%), with one standard deviation in parentheses. ϵ-opt: % of test instances solved to ϵ-optimality. Infeas: % of test instances for which the method fails to return a feasible solution. Speedup: the shifted geometric mean speedup relative to GRB, as defined in Table 1. Shaded rows: our methods.

n	Method	Time (s)	Subopt (%)	ϵ-opt (%)	Infeas (%)	Speedup (%)
2	GRB	0.07 (±0.048)	0.067 (±0.152)	100	0	0
	GRB HEUR	0.011 (±0.018)	103.717 (±145.868)	1	0	85.3
	CONFORMAL	0.004 (±0.002)	1.25 (±5.801)	78.3	14.2	94.5
	FALLBACK	0.021 (±0.045)	1.072 (±5.39)	92.5	0	71.6
	RCP	0.005 (±0.011)	3.692 (±18.564)	84.1	0	92.3
	CONSTANT	0.054 (±0.034)	9.452 (±61.13)	83.1	0	23.2
	NEAR	0.011 (±0.007)	47.85 (±135.462)	37.1	0	84.6
4	GRB	0.265 (±0.116)	0.018 (±0.09)	100	0	0
	GRB HEUR	0.011 (±0.007)	103.381 (±63.674)	0.5	0	95.7
	CONFORMAL	0.042 (±0.045)	1.542 (±11.516)	80.9	5.3	84.3
	FALLBACK	0.058 (±0.085)	1.46 (±11.212)	86.2	0	78.7
	RCP	0.05 (±0.066)	1.991 (±12.773)	83.4	0	81.5
	CONSTANT	0.101 (±0.075)	1.144 (±4.591)	88.5	0	61.9
	NEAR	0.039 (±0.066)	28.84 (±82.599)	4.7	0	85.7
6	GRB	0.27 (±0.211)	0.043 (±0.104)	100	0	0
	GRB HEUR	0.018 (±0.019)	74.521 (±35.678)	2	0	92.2
	CONFORMAL	0.09 (±0.103)	1.065 (±5.451)	79.9	11.1	66.4
	FALLBACK	0.134 (±0.176)	0.947 (±5.15)	91	0	51.9
	RCP	0.118 (±0.187)	2.331 (±14.463)	84.7	0	58.1
	CONSTANT	0.203 (±0.157)	1.725 (±8.769)	85.6	0	23.8
	NEAR	0.124 (±0.152)	20.516 (±30.913)	12.8	0	54.8
8	GRB	2.024 (±7.285)	0.066 (±0.134)	100	0	0
	GRB HEUR	0.072 (±0.086)	71.314 (±40.921)	4	0	93.4
	CONFORMAL	0.803 (±2.636)	0.867 (±5.016)	80	11.9	51.4
	FALLBACK	1.449 (±5.338)	0.764 (±4.716)	91.9	0	31.8
	RCP	1.259 (±4.083)	1.648 (±6.784)	84.3	0	37.2
	CONSTANT	1.346 (±2.882)	4.821 (±57.862)	78.4	0	13
	NEAR	1.54 (±14.51)	21.211 (±40.806)	14	0	38.4
12	GRB	4.747 (±17.028)	0.053 (±0.144)	100	0	0
	GRB HEUR	0.128 (±0.133)	103.386 (±72.925)	1	0	94.9
	CONFORMAL	2.917 (±6.037)	3.599 (±12.96)	81.4	1.6	19.1
	FALLBACK	3.132 (±8.073)	3.542 (±12.864)	83	0	17.7
	RCP	2.814 (±5.505)	3.917 (±13.224)	80.8	0	20.7
	CONSTANT	4.453 (±6.884)	4.699 (±32.623)	81.8	0	−23.6
	NEAR	1.331 (±2.799)	67.319 (±88.603)	3.8	0	65.9
18	GRB	168.606 (±186.626)	0.218 (±2.494)	98.7	0	0
	GRB HEUR	0.288 (±0.307)	98.581 (±17.437)	0	0	99.7
	CONFORMAL	12.569 (±29.655)	3.125 (±10.63)	80.3	2	92.3
	FALLBACK	13.252 (±29.055)	2.99 (±10.438)	82.6	0	92
	RCP	13.229 (±30.312)	2.955 (±10.402)	82.8	0	92.1
	CONSTANT	20.979 (±49.839)	1.839 (±7.664)	84.2	0	89.5
	NEAR	8.166 (±25.592)	53.408 (±31.001)	2.7	0	97

Table 5: Helicopter path-planning results. Time: mean solve time (s), with one standard deviation in parentheses. Subopt: mean relative suboptimality (%), with one standard deviation in parentheses. ϵ-opt: % of test instances solved to ϵ-optimality. Infeas: % of test instances for which the method fails to return a feasible solution. Speedup: the shifted geometric mean speedup relative to GRB, as defined in Table 1. Shaded rows: our methods.

n	Method	Time (s)	Subopt (%)	ϵ-opt (%)	Infeas (%)	Speedup (%)
15	GRB	0.074 (±0.052)	0.002 (±0.021)	100	0	0
	GRB HEUR	0.018 (±0.011)	9.082 (±22.623)	71	0	75.2
	CONFORMAL	0.002 (±0.002)	0.18 (±1.38)	77	20.9	96.7
	FALLBACK	0.024 (±0.049)	0.142 (±1.229)	97.9	0	68.4
	RCP	0.004 (±0.008)	0.422 (±3.952)	95.4	0	95.1
	CONSTANT	0.028 (±0.053)	0.637 (±4.259)	96.1	0	63.2
	NEAR	0.015 (±0.004)	2.923 (±17.267)	87.2	0	79.2
30	GRB	0.985 (±0.727)	0.008 (±0.055)	100	0	0
	GRB HEUR	0.069 (±0.056)	23.483 (±58.641)	53.4	0	92.2
	CONFORMAL	0.008 (±0.005)	0.503 (±3.174)	82.8	12.1	99.1
	FALLBACK	0.264 (±0.705)	0.156 (±1.154)	96.9	0	81.4
	RCP	0.009 (±0.025)	0.284 (±2.104)	95.6	0	99
	CONSTANT	0.17 (±0.548)	1.344 (±5.951)	89.9	0	87.5
	NEAR	0.054 (±0.029)	2.25 (±7.726)	82.6	0	93.9
45	GRB	2.297 (±1.628)	0.005 (±0.039)	100	0	0
	GRB HEUR	0.117 (±0.089)	25.242 (±76.722)	49.6	0	94
	CONFORMAL	0.01 (±0.005)	0.137 (±0.978)	79.6	17.5	99.5
	FALLBACK	0.601 (±1.411)	0.065 (±0.494)	97.9	0	84.3
	RCP	0.014 (±0.01)	0.251 (±1.512)	94.6	0	99.3
	CONSTANT	0.486 (±1.4)	1.35 (±7.182)	88.3	0	88
	NEAR	0.094 (±0.06)	2.665 (±8.872)	79.1	0	95.2
60	GRB	6.016 (±4.827)	0.012 (±0.066)	100	0	0
	GRB HEUR	0.206 (±0.143)	29.553 (±87.861)	44.3	0	95.6
	CONFORMAL	0.014 (±0.005)	0.084 (±0.853)	79.1	19.7	99.7
	FALLBACK	1.971 (±4.431)	0.078 (±0.824)	98.6	0	86.2
	RCP	0.043 (±0.474)	0.261 (±1.766)	95.6	0	99.4
	CONSTANT	1.249 (±4.32)	1.033 (±5.335)	90.7	0	92.4
	NEAR	0.161 (±0.101)	2.324 (±8.247)	81	0	96.5
75	GRB	13.228 (±10.792)	0.007 (±0.041)	100	0	0
	GRB HEUR	0.303 (±0.176)	37.407 (±94.591)	38.1	0	96.8
	CONFORMAL	0.02 (±0.008)	0.399 (±2.253)	82.7	11.1	99.8
	FALLBACK	2.965 (±8.179)	0.162 (±0.962)	96.6	0	93.5
	RCP	0.046 (±0.526)	0.287 (±1.215)	93.7	0	99.6
	CONSTANT	2.501 (±8.458)	1.02 (±4.516)	89.2	0	95.1
	NEAR	0.25 (±0.133)	2.132 (±6.31)	78.2	0	97.3

Table 6: School-bus routing results. n: number of children (yielding $55, 78, 105, 136, 171$ binary edge activations for $n = 10, 12, 14, 16, 18$ respectively). Time: mean solve time (s), with one standard deviation in parentheses. Subopt: mean relative suboptimality (%), with one standard deviation in parentheses. ϵ-opt: % of test instances solved to ϵ-optimality. Infeas: % of test instances for which the method fails to return a feasible solution. Speedup: the shifted geometric mean speedup relative to GRB, as defined in Table 1. Shaded rows: our methods.

n	Method	Time (s)	Subopt (%)	ϵ-opt (%)	Infeas (%)	Speedup (%)
10	GRB	1.887 (±1.972)	0 (±0)	100	0	0
	GRB HEUR	0.234 (±0.413)	39.603 (±34.816)	0.5	0	87.5
	CONFORMAL	0.024 (±0.126)	0 (±0)	98.5	1.5	98.7
	FALLBACK	0.019 (±0.104)	0 (±0)	100	0	99
	RCP	0.026 (±0.135)	0 (±0)	100	0	98.6
	CONSTANT	0.476 (±0.576)	0.432 (±3.139)	97.6	0	73.9
	NEAR	0.055 (±0.191)	0.007 (±0.152)	99.8	0	97
12	GRB	5.351 (±3.618)	0 (±0)	100	0	0
	GRB HEUR	0.399 (±0.587)	44.618 (±36.737)	0.4	0	92.4
	CONFORMAL	0.03 (±0.18)	0 (±0)	95.2	4.8	99.5
	FALLBACK	0.038 (±0.202)	0 (±0)	100	0	99.3
	RCP	0.026 (±0.154)	0.006 (±0.186)	99.9	0	99.5
	CONSTANT	0.584 (±0.679)	0.059 (±0.481)	98.3	0	88.4
	NEAR	0.042 (±0.187)	0.05 (±0.672)	98.9	0	99.2
14	GRB	12.067 (±7.467)	0 (±0)	100	0	0
	GRB HEUR	0.419 (±0.562)	53.305 (±34.605)	1.2	0	96.2
	CONFORMAL	0.012 (±0.085)	0 (±0)	93.9	6.1	99.9
	FALLBACK	0.073 (±0.347)	0 (±0)	100	0	99.5
	RCP	0.016 (±0.105)	0.002 (±0.048)	99.9	0	99.9
	CONSTANT	0.467 (±0.853)	0.136 (±0.828)	96.1	0	96.1
	NEAR	0.035 (±0.156)	0.074 (±0.641)	98.2	0	99.7
16	GRB	16.055 (±12.15)	0 (±0)	100	0	0
	GRB HEUR	0.75 (±1.006)	84.641 (±50.41)	1.8	0	94.5
	CONFORMAL	0.026 (±0.166)	0.002 (±0.056)	85.1	14.8	99.8
	FALLBACK	0.292 (±0.921)	0.002 (±0.052)	99.9	0	98.4
	RCP	0.041 (±0.249)	0.016 (±0.168)	99.1	0	99.7
	CONSTANT	0.776 (±1.305)	0.527 (±2.891)	94.5	0	94.6
	NEAR	0.065 (±0.27)	0.307 (±2.607)	96.5	0	99.5
18	GRB	37.478 (±26.349)	0 (±0)	100	0	0
	GRB HEUR	0.116 (±0.42)	205.944 (±104.521)	1.6	0	99.7
	CONFORMAL	0.008 (±0.004)	0.003 (±0.068)	81.9	17.9	100
	FALLBACK	0.698 (±2.497)	0.003 (±0.061)	99.8	0	98.9
	RCP	0.013 (±0.021)	0.042 (±0.641)	98.8	0	99.9
	CONSTANT	1.187 (±1.337)	0.135 (±1.175)	96.1	0	96
	NEAR	0.054 (±0.165)	0.982 (±4.584)	90.9	0	99.8