AUTOREGRESSIVE DIFFERENTIABLE METHOD FOR INTEGER PROGRAMMING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce an autoregressive differentiable method to solve 0-1 integer programs. We fix an arbitrary order of the binary variables and we train a transformer to predict the next bit while remaining in the feasible set. Our method is first trained on feasible incumbents provided by any solver, thus allowing us to initialize the transformer in the feasible set. Our procedure then implements a Lagrangian penalty to penalize infeasible solutions, and the transformer is further trained to explore the feasible set using Gumbel-softmax activations on the relaxed objective. We have tested our method on non-convex instances of quadratic knapsack problem and demonstrated consistent improvement upon state-of-the-art open-source solvers for dense problems up to 10,000 binary variables. In particular, we empirically demonstrate a phenomenon akin to a tunneling effect where the effective change of variables from binary variable to the continuous weights of the transformer that the method implements enables crossing barriers in the relaxed objective landscape.

1 INTRODUCTION

Binary integer programming is a central modeling language for combinatorial optimization (Nemhauser & Wolsey, 1988). Problems including knapsack (Martello & Toth, 1990), maximum cut (Goemans & Williamson, 1995), facility location (Cornuéjols et al., 1990), sparse regression (Bertsimas et al., 2016), and scheduling (Pinedo, 2016) can be formulated over binary vectors, yet many practically relevant instances remain computationally challenging (Belotti et al., 2013; Bertsimas et al., 2016; Bixby, 2012; Lodi, 2010; Gleixner et al., 2021). Modern branch-and-cut solvers and primal heuristics are extremely powerful, but can still spend substantial time improving a good feasible incumbent (Berthold, 2013; Fischetti & Lodi, 2003; Rothberg, 2007).

We ask whether autoregressive differentiable models can serve as enhanced primal mechanisms for binary integer programs (Vinyals et al., 2015; Bello et al., 2017; Geng et al., 2025). Rather than predicting a solution from scratch, we initialize a neural model at a feasible incumbent obtained from a standard solver (Nair et al., 2020; Khalil et al., 2022; Han et al., 2023), then optimize it directly against a differentiable relaxation of the original objective with a Lagrangian penalty for constraint violation (Di Pillo & Grippo, 1989). The resulting method searches in neural network weight space rather than directly in the decision variables. This formulation is intended to complement conventional integer programming machinery: the solver supplies a feasible point and the neural reparameterization is used only for subsequent primal improvement. The central question is whether the nonlinear, shared autoregressive parametrization exposes useful optimization directions that differ from direct coordinate-wise movement in the binary decision space.

Our starting point is deliberately instance-specific. The neural network is not trained to generalize across a dataset of optimization instances; instead, it is initialized at a solution already found by a solver and used as a local improvement mechanism. This separates feasibility discovery from neural improvement and lets the transformer exploit a high-quality incumbent without learning the combinatorial problem from scratch.

We focus on binary quadratic integer programs

$$\max_{x\in\{0,1\}^n} x^\top Qx \qquad \text{s.t.} \qquad Ax \le b, \tag{1}$$

where $Q \in \mathbb{R}^{n\times n}$, $A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$. Our experimental testbed is the quadratic knapsack problem (QKP), following and extending Lopez Piqueres & Chen (2025). QKP is known to remain

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.

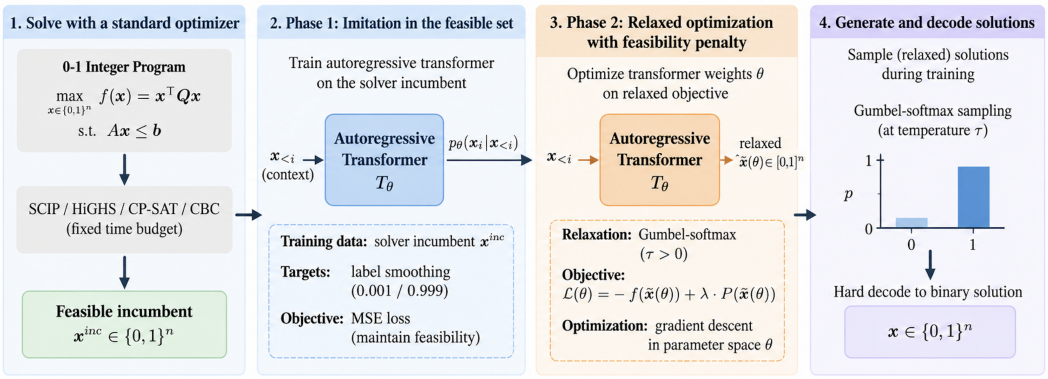

Figure 1: **Overview of ADIP.** Starting from a feasible incumbent x^{inc} obtained from a standard integer-programming solver, Phase 1 trains an autoregressive transformer to reproduce the incumbent using smoothed binary targets. Phase 2 optimizes the transformer parameters directly against the relaxed integer-program objective with a Lagrangian feasibility penalty, using Gumbel-softmax to maintain near-discrete outputs. Hard candidates are decoded throughout training, and the best feasible solution encountered is returned.

strongly NP-hard even with small integer weights (Gallo et al., 1980; Pisinger et al., 2007), and arbitrary Q yields a non-convex relaxed landscape. Although we experiment with non-convex quadratic objectives, the method applies whenever the binary objective admits a differentiable continuous relaxation.

Our method, which we call Autoregressive Differentiable Integer Progamming (ADIP), has two phases. We fix an arbitrary variable ordering and use a causal transformer to learn $p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \mid x_{<i})$. For each instance, an open-source solver such as SCIP (Hojny et al., 2025), HiGHS (Huangfu & Hall, 2018), CP-SAT (Perron et al., 2023), or CBC (Forrest & Lougee-Heimer, 2005; COIN-OR, 2026) first produces a feasible incumbent x^{inc}. In Phase 1, the transformer is trained to reproduce x^{inc} using targets ε and $1-\varepsilon$ (instead of 0 and 1), preventing softmax saturation and preserving gradients. In Phase 2, the same weights are optimized against

$$\mathcal{J}_\lambda(\theta) = \tilde{x}_\theta^\top Q \tilde{x}_\theta - \lambda \sum_{j=1}^{m} \mathrm{ReLU}((A\tilde{x}_\theta - b)_j), \tag{2}$$

where $\tilde{x}_\theta \in [0,1]^n$ is generated autoregressively. Gumbel-softmax activations (Jang et al., 2017; Maddison et al., 2017) encourage near-discrete exploration, while the best feasible hard solution encountered is retained throughout training.

For the purpose of advancing open-source software and open science, we deliberately chose to compare against state-of-the-art open-source solvers, and not against their closed-source competitors. Our contributions are fourfold. First, we introduce an autoregressive nonlinear reparametrization of relaxed binary decision variables for integer programs, in which the shared parameters of the autoregressive model themselves become the optimization variables. Second, we present our specific method ADIP combining transformer-based incumbent imitation, finite-logit targets, Gumbel-softmax exploration, and a Lagrangian feasibility penalty in a two-phase primal-improvement procedure. Third, we provide theoretical results showing that incumbent improvement remains NP-hard, that the transformer reparametrization retains the binary search space, and that parameter-space gradient flow induces a state-dependent geometry in decision space. Fourth, we evaluate ADIP on random QKP instances under matched computational budgets and study whether its trajectories reach attraction basins different from those reached by combinatorial optimization or by direct relaxed optimization, uncovering a "tunneling-like" phenomenon.

2 Related Work

Our method is closely related in motivation to Differentiable Integer Linear Programming (DiffILO) (Geng et al., 2025), which constructs a probabilistic continuous reformulation of an ILP. The two

approaches differ in how differentiability is obtained and in the role of the neural parametrization. DiffILO associates independent Bernoulli parameters with binary decisions and derives a stochastic constraint surrogate. Sornwanee (2026) shows that its sample-wise loss contains a discontinuous indicator, so the differentiation-expectation interchange used to justify its stochastic gradient is generally invalid and omits boundary terms that are not scalable for general constraints. ADIP instead differentiates an explicit continuous Gumbel-softmax autoregressive relaxation (Jang et al., 2017; Maddison et al., 2017).

Autoregressive models have previously been used for combinatorial optimization and mixed-integer programming. Tang et al. (2025) use autoregressive reinforcement learning to sequentially fix MILP variables before a conventional solver completes the solution, while variational neural annealing uses autoregressive distributions for Ising-type combinatorial optimization (Hibat-Allah et al., 2021). In contrast, ADIP first fits an autoregressive model to a feasible solver incumbent and then makes its shared parameters the optimization variables themselves: the original relaxed objective is back-propagated through the nonlinear map $\theta \mapsto \tilde{x}_\theta$. To the best of our knowledge, this combination of incumbent initialization, autoregressive reparametrization, and direct differentiable optimization of the integer-program objective has not previously been proposed for general binary integer programming.

ADIP is also related to quantum-inspired generative optimization. Constrained tensor networks (Lopez Piqueres & Chen, 2025) encode linear constraints exactly into matrix product states and optimize a distribution supported only on feasible configurations; their QKP benchmark directly motivates ours. In contrast, ADIP does not encode the feasible set architecturally, but initializes from a feasible incumbent and optimizes through a Lagrangian penalty. The Generative Quantum Eigensolver (Nakaji et al., 2024) likewise places the trainable parameters in a classical autoregressive model, but uses the transformer to generate quantum circuits rather than the decision variables themselves. Quantum Hamiltonian Descent (Leng et al., 2023) provides a complementary quantum mechanism for traversing non-convex landscapes via Hamiltonian dynamics and motivates our tunneling analogy.

3 PRELIMINARIES

We consider Eq. 1 with feasible set $\mathcal{F} = \{x \in \{0,1\}^n : Ax \leq b\}$ and continuous relaxation $\tilde{x} \in [0,1]^n$. Under a fixed variable ordering, a causal transformer T_θ (Vaswani et al., 2017) factorizes

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \mid x_{<i}). \tag{3}$$

We use a causal transformer with alphabet $\{"0", "1"\}$, and for differentiable self-generation, a relaxed bit is embedded via token embeddings e_0, e_1 as

$$e(\tilde{x}_i) = (1 - \tilde{x}_i)e_0 + \tilde{x}_i e_1, \tag{4}$$

which allows gradients to propagate through the generated prefix $(\tilde{x}_\theta)_{<i}$.

For logits $\ell_{i,0}, \ell_{i,1} \in \mathbb{R}$, independent Gumbel noises $g_{i,0}, g_{i,1}$, and temperature $\tau > 0$, Gumbel–softmax (Jang et al., 2017) gives

$$\pi_{i,k} = \frac{\exp((\ell_{i,k} + g_{i,k})/\tau)}{\sum_{r\in\{0,1\}} \exp((\ell_{i,r} + g_{i,r})/\tau)}, \qquad \tilde{x}_i = \pi_{i,1}. \tag{5}$$

In our experiments we use the straight-through variant: the forward pass uses $\hat{x}_i = \mathbf{1}\{\pi_{i,1} \geq \pi_{i,0}\}$, while the backward pass differentiates the continuous relaxation in Eq. 5.

4 METHODOLOGY

We now describe ADIP, the proposed autoregressive differentiable method for binary integer programming. The method has two phases. The first phase initializes a transformer at a feasible incumbent found by an external solver. The second phase optimizes the transformer weights using a differentiable relaxation of the objective with a feasibility penalty.

Algorithm 1 ADIP: Autoregressive Differentiable Integer Programming

Require: (Q, A, b), solver $\mathcal{S}$, T_{seed}, K_1, K_2, penalty λ, temperature τ
1: Run $\mathcal{S}$ for T_{seed}; let x^{inc} be its best feasible incumbent
2: Initialize transformer parameters θ
3: **for** $k = 1, \ldots, K_1$ **do**
4: Update θ to minimize $\mathcal{L}_{\text{init}}$ in Eq. 7 under teacher forcing
5: **end for**
6: $x^\star \leftarrow x^{\text{inc}}$
7: **for** $k = 1, \ldots, K_2$ **do**
8: Generate $\tilde{x}_\theta^{(k)}$ autoregressively using Gumbel–softmax temperature τ
9: Ascend $\nabla_\theta \mathcal{J}_\lambda(\theta)$ from Eq. 8
10: Decode $\hat{x}^{(k)}$ and update $x^\star$ if it is feasible and improves the objective
11: **end for**
12: **return** $x^\star$

4.1 Overview

Given an instance of Eq. 1, ADIP first runs a conventional solver for T_{seed} and extracts its best feasible incumbent x^{inc}. A causal transformer is then trained under teacher forcing to reproduce this incumbent. Starting from those trained weights, Phase 2 generates a relaxed sequence autoregressively with Gumbel–softmax and ascends the penalized objective, while every hard-decoded candidate is checked against the original constraints and the best feasible solution is retained. Algorithm 1 summarizes the procedure.

4.2 Complexity of incumbent improvement

It is worth noting that starting from a feasible incumbent does not make the residual search problem easy.

Theorem 1 (Hardness of incumbent improvement)**.** *For QKP with integer-valued objective coefficients and strictly positive integer weights, given a feasible incumbent x^{inc}, deciding whether there exists a feasible x with $x^\top Q x > (x^{\text{inc}})^\top Q x^{\text{inc}}$ is NP-complete.*

The proof, in Appendix A.3, reduces QKP decision to incumbent improvement by adding a capacity-saturating item of value $K - 1$. Thus ADIP addresses an NP-hard residual search problem even after a feasible point is provided.

4.3 Expressivity of the autoregressive parametrization

Universal approximation results for transformers (Yun et al., 2020) imply that the neural change of variables does not discard binary solutions.

Proposition 1 (Combinatorial expressivity)**.** *Fix sequence length n and a transformer class satisfying the universal approximation property of Yun et al. (2020). For every $x^\star \in \{0,1\}^n$ and $\varepsilon \in (0, 1/2)$, there exist parameters θ such that $\|\tilde{x}_\theta - x^\star\|_\infty < \varepsilon$. Hence threshold decoding at $1/2$ is surjective onto $\{0,1\}^n$.*

Corollary 1 (Hardness under autoregressive reparametrization)**.** *Under Proposition 1 and given $M \in \mathbb{R}$, deciding whether there exists θ such that $\hat{x}_\theta^\top Q \hat{x}_\theta > M$ subject to $A\hat{x}_\theta \leq b$ [1] is NP-hard.*

Proofs are in Appendices A.4-A.5. The parametrization therefore retains the combinatorial search space and its worst-case hardness. This separates expressivity from optimization: the role of the transformer reparameterization is to provide a different continuous geometry through which the discrete search space is explored (see Paragraph 4.8).

[1] This is the decision version of globally optimizing the objective via the hard-decoded transformer.

4.4 PHASE 1: FEASIBLE INCUMBENT IMITATION

Let x^{inc} be the best feasible solver incumbent. For each position i, the transformer receives $x^{\mathrm{inc}}_{<i}$ and predicts x^{inc}_i. Exact targets in $\{0,1\}$ can drive the logits toward saturation, so we use

$$y_i^{\mathrm{inc}} = \begin{cases} 1-\varepsilon, & x_i^{\mathrm{inc}} = 1, \\ \varepsilon, & x_i^{\mathrm{inc}} = 0, \end{cases} \tag{6}$$

$$\mathcal{L}_{\mathrm{init}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i^{\mathrm{inc}}\log p_\theta(x_i = 1 \mid x^{\mathrm{inc}}_{<i}) + (1-y_i^{\mathrm{inc}})\log\big(1 - p_\theta(x_i = 1 \mid x^{\mathrm{inc}}_{<i})\big)\right], \tag{7}$$

with $\varepsilon = 0.001$ in our experiments. Here target smoothing enforces a finite logit gap $\log((1-\varepsilon)/\varepsilon)$, whereas exact targets are reached only in the infinite-logit limit. Thus Phase 1 yields a model whose hard decoding reproduces the feasible incumbent while retaining usable gradients for Phase 2. The decoded hard sequence agrees with x^{inc} whenever the predicted probabilities are on the correct side of $1/2$. At the same time, the relaxed output remains a finite-logit representation near the incumbent, so Phase 2 begins from a feasible discrete anchor without having destroyed the gradient signal needed to move away from it.

4.5 PHASE 2: DIFFERENTIABLE OPTIMIZATION IN TRANSFORMER WEIGHT SPACE

Let $\tilde{x}_\theta \in [0,1]^n$ denote the relaxed autoregressive output and define

$$\Phi(\tilde{x}) = \sum_{j=1}^{m}\mathrm{ReLU}((A\tilde{x}-b)_j), \qquad \mathcal{J}_\lambda(\theta) = \tilde{x}_\theta^\top Q\tilde{x}_\theta - \lambda\Phi(\tilde{x}_\theta). \tag{8}$$

We maximize $\mathcal{J}_\lambda$ with respect to θ. During Phase 2 we monitor hard decoded feasibility $A\hat{x}_\theta \le b$; the returned solution is always the best feasible hard candidate encountered.

For integer-valued constraints, the incumbent admits a conservative finite penalty scale for binary solutions.

Proposition 2 (Sufficient penalty for binary feasibility)**.** *Let $f_{\mathrm{inc}} = (x^{\mathrm{inc}})^\top Q x^{\mathrm{inc}}$ and $U_Q = \sum_{i,j}\max\{Q_{ij}, 0\}$. If A, b are integer-valued and $\lambda > U_Q - f_{\mathrm{inc}}$, every infeasible binary x satisfies $x^\top Qx - \lambda\Phi(x) < f_{\mathrm{inc}}$; hence no global maximizer of the penalized binary objective is infeasible.*

The proof is in Appendix A.6. The guarantee does not extend to arbitrary relaxed points, whose violations can be arbitrarily small. Proposition 2 cannot be interpreted as proving that the complete relaxed trajectory remains feasible, however, in practice, we will see in the subsequent paragraph 4.7 that Gumbel-softmax activations often yield near-discrete bits, and this proposition thus explains why near-discrete solutions encountered on the trajectory remain feasible thanks to the Lagrangian penalty.

4.6 AUTOREGRESSIVE RELAXED ROLLOUT

Phase 2 differentiates through the complete generated sequence. Using Eq. 4, at step i the model computes $\ell_i = T_\theta(e(\tilde{x}_1), \ldots, e(\tilde{x}_{i-1}))$ and $\tilde{x}_i = \pi_{i,1}$ from Eq. 5. This recursion defines a differentiable map $\theta \mapsto \tilde{x}_\theta$, so the relaxed objective can be backpropagated through the entire autoregressive computational graph. This dependence across positions is essential: changing the parameters that affect an early relaxed token also changes the prefix consumed at all subsequent generation steps. The resulting Jacobian therefore couples coordinates through the shared autoregressive computation and does not treat the n relaxed variables as independent optimization parameters.

4.7 NEAR-DISCRETE EXPLORATION WITH GUMBEL-SOFTMAX

Standard softmax optimization can traverse highly fractional vectors with no combinatorial meaning: a fractional point may have high relaxed objective value while corresponding to no good feasible binary solution. Gumbel–softmax biases Phase 2 toward near-binary configurations. This has two practical implications. First, objective values encountered during training are evaluated near the

combinatorial vertices. Second, the trajectory can make relatively sharp moves between near-discrete configurations rather than slowly traversing the interior of $[0,1]^n$. In the binary case, with logit difference $\Delta = \ell_1 - \ell_0$ and Gumbel noise difference $G = g_1 - g_0$, we have $\tilde{x}_\tau = \sigma((\Delta + G)/\tau)$ where σ is the logistic function.

Proposition 3 (Low-temperature Gumbel–softmax)**.** *Let $D_\tau = \partial \tilde{x}_\tau / \partial \Delta$ and let q be the standard logistic density. For finite Δ, as $\tau \to 0$,*

$$\tilde{x}_\tau \to \mathbf{1}\{\Delta + G > 0\} \text{ almost surely}, \qquad \mathbb{E}[D_\tau] \to q(-\Delta), \qquad \mathrm{Var}(D_\tau) = \Theta(\tau^{-1}), \tag{9}$$

with $\mathbb{E}[D_\tau^2] = q(-\Delta)/(6\tau) + o(\tau^{-1})$.

Thus decreasing τ confines the forward transition to an $O(\tau)$-wide region while preserving a finite expected gradient through increasingly rare $O(\tau)$ switching events carrying $O(\tau^{-1})$ gradients; the diverging variance motivates keeping τ bounded away from zero. In our autoregressive model, switching an early coordinate also changes the conditioning prefix of all later decisions, thus flipping multiple bits in one gradient step. The proof is in Appendix A.7.

4.8 Why optimize in transformer space?

The central motivation for ADIP is the nonlinear change of variables $\theta \mapsto \tilde{x}_\theta \mapsto F_\lambda(\tilde{x}_\theta)$, where $F_\lambda(\tilde{x}) = \tilde{x}^\top Q \tilde{x} - \lambda \Phi(\tilde{x})$. Let $J_\theta = \partial \tilde{x}_\theta / \partial \theta$.

Theorem 2 (Induced decision-space dynamics)**.** *For continuous-time gradient ascent $\dot{\theta} = \nabla_\theta F_\lambda(\tilde{x}_\theta)$,*

$$\dot{\tilde{x}} = K_\theta \nabla_{\tilde{x}} F_\lambda(\tilde{x}), \qquad K_\theta := J_\theta J_\theta^\top \succeq 0. \tag{10}$$

Proof. By the chain rule, $\nabla_\theta F_\lambda = J_\theta^\top \nabla_{\tilde{x}} F_\lambda$ and $\dot{\tilde{x}} = J_\theta \dot{\theta}$, giving Eq. 10. Moreover $v^\top K_\theta v = \|J_\theta^\top v\|_2^2 \geq 0$. □

Direct relaxed gradient ascent follows $\dot{\tilde{x}}_{\mathrm{direct}} = \nabla_{\tilde{x}} F_\lambda(\tilde{x})$, corresponding to $K = I$, whereas ADIP can rotate, suppress, or amplify directions through the state-dependent kernel K_θ, analogous to parameter-space kernel dynamics in neural networks (Jacot et al., 2018). Theorem 2 therefore makes precise why the transformer is more than a high-dimensional encoding of $\tilde{x}$. Direct optimization follows the Euclidean gradient of F_λ, whereas ADIP follows that gradient after multiplication by a positive-semidefinite matrix determined by the current network Jacobian. Because J_θ changes during training, the induced kernel is itself state-dependent.

The induced geometry also depends on the specific parameters, not only on the represented solution. For one attention head with input representation H, let $Q = HW_Q$ (queries), $K = HW_K$ (keys), and $V = HW_V$ (values). Causally masked attention is $\mathrm{softmax}(QK^\top/\sqrt{d_k} + M)V$, so the forward map depends on W_Q, W_K through their product while gradient descent acts on the matrices separately.

Theorem 3 ((informal) Path-dependent induced geometry)**.** *Assume an attention head affects the relaxed rollout nontrivially. For $c > 0$, replace W_Q by cW_Q and W_K by $c^{-1}W_K$, leaving all other parameters fixed. The resulting transformer defines exactly the same autoregressive map for every prefix, yet for some $c \neq 1$ its kernel $J_\theta J_\theta^\top$ differs. Moreover, there exists a quadratic objective for which the two functionally identical transformers induce different instantaneous velocities in relaxed decision space.*

In Appendix A.8, we state a formal version of the above theorem and provide a full proof. The key symmetry is that $W_Q W_K^\top$ is invariant under this rescaling, while the corresponding Jacobian blocks scale as $J_Q \mapsto c^{-1} J_Q$ and $J_K \mapsto c J_K$. Thus the weights retain geometric information invisible in $\tilde{x}$ itself. Theorem 3 establishes that one can construct rescaled models that are functionally indistinguishable, producing the same conditional distributions at every autoregressive step, yet gradient ascent can evolve differently because it acts on the parameterization rather than on the represented function alone. In particular, the ADIP trajectory cannot in general be described by an autonomous vector field depending only on the current relaxed solution: two parameterizations can represent exactly the same autoregressive map and nevertheless induce different local optimization dynamics. Motivated by this result, we call a trajectory "tunneling-like" if it reaches a point whose endpoint under direct relaxed gradient ascent belongs to a different attraction basin from that of the initial incumbent. This operational notion is tested in Section 5.3.

Table 1: Matched-budget results on random QKP instances. Means and standard deviations (between parentheses) are computed across 100 runs per experiment for size $n = 400$ and 10 runs per experiment for sizes $n \in \{1000, 10000\}$.

n	Base solver	Matched solver obj.	ADIP obj.	Improvement (%)
400	SCIP	2080.51 (339.71)	2145.45 (338.41)	5.71 (4.57)
1000	SCIP	5081.30 (562.45)	5710.90 (620.19)	12.78 (10.29)
10000	SCIP	0.00 (0.00)	60.21 (26.01)	∞
400	HiGHS	226.00 (240.42)	893.00 (796.20)	378.48 (156.70)
1000	HiGHS	0.00 (0.00)	300.00 (323.85)	∞
400	CP-SAT	726.00 (234.76)	1277.50 (355.67)	77.31 (8.35)
1000	CP-SAT	1317.50 (106.77)	2687.50 (584.78)	102.85 (27.95)
400	CBC	19.50 (13.44)	133.50 (21.92)	848.45 (765.87)
1000	CBC	0.00 (0.00)	300.00 (323.85)	∞

5 Experiments

We evaluate whether ADIP improves solver incumbents under matched wall-clock budgets and whether its weight-space trajectory accesses different basins of the relaxed objective. The experiments therefore address two complementary questions: whether the method provides practical primal improvement over state-of-the-art open-source solvers given the same total time, and whether the proposed reparametrization changes the local optimization behavior in the relaxed landscape.

5.1 Quadratic knapsack benchmark and protocol

Following Lopez Piqueres & Chen (2025), we consider

$$\max_{x \in \{0,1\}^n} x^\top Q x \qquad \text{s.t.} \qquad w^\top x \leq C, \tag{11}$$

where $w \in \mathbb{R}^n_+$ are item weights and $C > 0$ is the capacity. In our experiments, Q is a random non-symmetric integer-valued matrix, $Q_{ij} \sim \mathrm{Unif}\{-5, \ldots, 5\}$, $w_i \sim \mathrm{Unif}\{0, \ldots, 5\}$, $C = \frac{1}{4}\sum_i w_i$, $n \in \{400, 1000, 10000\}$, and we generate 100 instances per experiment for size $n = 400$, and 10 instances per experiment for sizes $n \in \{1000, 10000\}$. Our baselines are SCIP (Achterberg, 2009; Hojny et al., 2025), HiGHS (Huangfu & Hall, 2018), CP-SAT (Perron et al., 2023), and CBC (Forrest & Lougee-Heimer, 2005; COIN-OR, 2026). Only SCIP can natively run quadratic integer programs, the other solvers require performing Fortet linearization (Fortet, 1960) leading to integer linear programs with $O(n^2)$ variables, which may become impractical already at sizes $n = 1000$. Each solver runs for $T_{\text{seed}} = 500$ seconds to provide x^{inc}; ADIP then trains for $T_{\text{train}} = 500$ seconds. The matched baseline runs the solver alone for $T_{\text{total}} = T_{\text{seed}} + T_{\text{train}}$, and we report

$$\mathrm{Imp}_{\text{time}}(\%) = 100 \frac{f(x^{\text{ADIP}}) - f(x^{\text{OSS}}_{T_{\text{total}}})}{|f(x^{\text{OSS}}_{T_{\text{total}}})|}. \tag{12}$$

The default model is a two-layer causal transformer with embedding dimension 64, four heads, and feed-forward dimension 256. Phase 1 uses $K_1 = 150$, $\eta_1 = 10^{-3}$, and $\varepsilon = 0.001$; Phase 2 uses $K_2 = 2000$, $\eta_2 = 10^{-3}$ for size $n = 400$ (respectively $\eta_2 = 10^{-4}$ for sizes $n \in \{1000, 10000\}$), $\lambda = 100$ for size $n = 400$ (respectively $\lambda = 500$ for sizes $n \in \{1000, 10000\}$) and $\tau = 0.5$. Full implementation details are given in Appendix A.2. The code to reproduce the experiments is available at `https://anonymous.4open.science/r/adip-38DE`.

5.2 Main results

Table 1 reports the matched-budget comparison. We find that ADIP outperforms SCIP in 58% of the runs across the $n = 400$ experiments, 90% of the runs across the $n = 1000$ experiments, and 100% of the runs across the $n = 10000$ experiments, suggesting that ADIP is stronger when the solvers leave greater room for improvement (at large problem sizes).

Figure 2 shows a representative objective-value trajectory. Starting from a SCIP incumbent, the best feasible decoded solution improves during transformer training. The staircase structure reflects

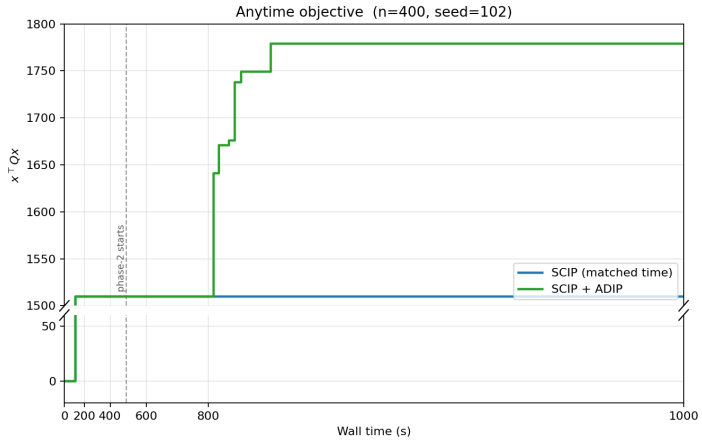

Figure 2: Representative objective-value trajectory: initial incumbent search and hard-decoded objective over ADIP training.

improvements of the discrete incumbent rather than the continuous relaxed training objective. The improvement also correlates with the SCIP gap at T_{seed}; this secondary diagnostic is shown in Appendix A.2. This observation is consistent with the intuition that the neural improvement procedure is most useful when substantial optimization remains after the seed phase, although the gap alone should not be interpreted as determining ADIP performance.

5.3 Barrier-crossing and tunneling-like behavior

We next investigate whether the transformer reparametrization accesses regions of the relaxed landscape that are difficult to reach by combinatorial optimization or by direct optimization in $\tilde{x}$-space. Let $E(\tilde{x}) = -\tilde{x}^\top Q\tilde{x}$ denote the energy associated with the maximization objective. For an initial relaxed point $\tilde{x}^{(0)}$, we run direct gradient descent on E over the continuous feasible relaxation,

$$\tilde{x}^{(t+1)} = \tilde{x}^{(t)} - \eta\nabla_{\tilde{x}}E(\tilde{x}^{(t)}), \tag{13}$$

checking $A\tilde{x} \leq b$ at each step. We apply this procedure to two types of initial points: the original solver incumbent x^{inc}, viewed as a point in $[0,1]^n$, and relaxed transformer outputs $\tilde{x}_\theta^{(k)}$ encountered during Phase 2. We iterate until convergence or until the feasible boundary prevents further improvement. If these initial points converge to different endpoints, we can infer that they belong to different attraction basins of the direct relaxed dynamics. The comparison therefore asks whether the transformer trajectory has entered a different region of the non-convex relaxed landscape.

Empirically, ADIP visits points whose relaxed-gradient endpoints differ from that of the initial incumbent, indicating that optimization in transformer parameter space can cross basin boundaries not crossed by direct local optimization from the incumbent. This experiment directly probes the geometric claim of Theorems 2–3: if the transformer merely reproduced ordinary Euclidean optimization in $\tilde{x}$-space, points generated from the incumbent would not be expected to systematically acquire different direct-gradient endpoints. We interpret this as a tunneling-like effect induced by the autoregressive reparametrization. Figure 3 visualizes the trajectories; t-SNE is used only for visualization, while basin identity is determined from endpoints in the original high-dimensional relaxed space.

Overall, the experiments show that ADIP can improve solver incumbents under matched computational budgets and that optimization in transformer parameter space can reach regions of the relaxed landscape that are not reached by combinatorial solvers or direct optimization from the solver incumbent. We therefore view ADIP as a complementary primal-improvement mechanism.

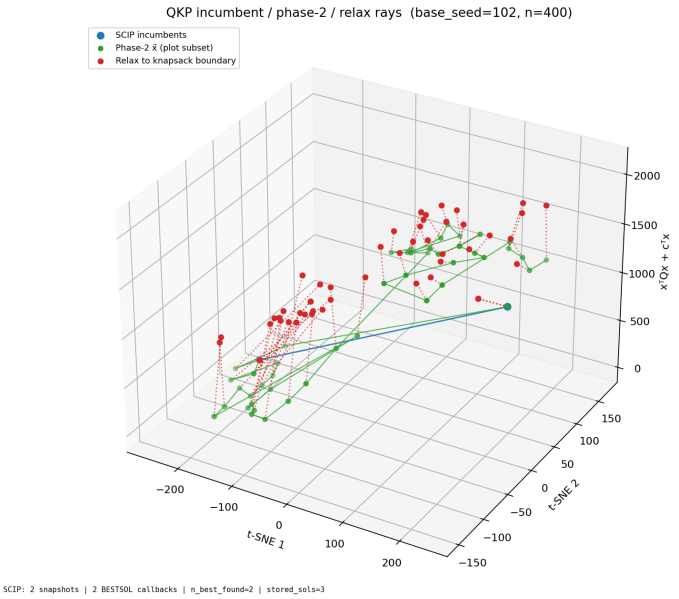

Figure 3: Visualization of the tunneling-like experiment. The solver incumbents, relaxed ADIP trajectory, and direct-relaxation endpoints are projected with t-SNE; the vertical axis shows objective value. Basin identity is determined in the original space.

Table 2: Representative basin analysis of the relaxed landscape.

n	Trajectory points	Distinct endpoints	Incumbent endpoint	Best endpoint
400	73	45	1931.3	1971.8
400	72	52	1842.5	2020.4
1000	90	35	4281.0	5083.0
1000	88	29	5248.0	5874.1

6 Limitations

ADIP is trained separately for each instance and requires a feasible incumbent from an external solver. The Lagrangian penalty does not guarantee feasibility of every relaxed intermediate point, although Gumbel-softmax exploration remains close to the original combinatorial structure and the algorithm always returns the best feasible hard solution encountered. Our tunneling analogy is geometric and empirical and does not imply global optimization guarantees. Finally, our experiments focus on random QKP instances; evaluating the method on larger and more diverse integer-programming benchmarks, as well as amortizing it across related instances, are important directions for future work.

7 Conclusion

We introduced ADIP, which uses an autoregressive transformer as a nonlinear reparametrization of relaxed binary decision variables for primal improvement in integer programming. Optimizing through the transformer induces a state-dependent geometry in decision space, providing a search mechanism that differs from combinatorial optimization and from direct optimization of the relaxed variables. On random QKP instances, ADIP improves state-of-the-art solver incumbents under matched computational budgets and accesses distinct attraction basins of the relaxed landscape. Together, these results suggest that autoregressive neural reparametrization can provide a complementary search mechanism for hard discrete optimization.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and editing portions of the manuscript, including language polishing, improving clarity and presentation, and reducing the length of the paper to satisfy formatting constraints. Generative AI tools were also used to assist with software development, including drafting, debugging, and refining parts of the experimental code. The central research ideas, problem formulation, methodological contributions, and key scientific innovations of this work were developed by the authors and did not originate from generative AI tools.

All AI-assisted text, mathematical arguments, references, and code were reviewed and verified by the authors. In particular, literature and citations suggested with AI assistance were checked against the original sources, theoretical arguments were independently verified by the authors, and AI-assisted code was inspected and tested as part of the experimental implementation. The authors take full responsibility for the final content of the paper, including all text, claims, proofs, code, and experimental results produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the complete implementation of ADIP and the experimental pipeline at `https://anonymous.4open.science/r/adip-38DE`. The repository includes instructions for reproducing the experiments reported in the paper, including instance generation, solver baselines, model training, and evaluation. The model architecture, optimization procedure, and principal experimental settings are described in the main text, with additional implementation and hyperparameter details provided in the appendix. Complete proofs of the theoretical results are also provided in the appendix. Together, the released code, experimental specifications, and theoretical appendices are intended to enable independent reproduction of the results reported in this work.

REFERENCES

Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. doi: 10.1007/s12532-008-0001-1.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *International Conference on Learning Representations Workshop*, 2017. URL https://arxiv.org/abs/1611.09940.

Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013. doi: 10.1017/S0962492913000032.

Timo Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6): 611–614, 2013. doi: 10.1016/j.orl.2013.08.007.

Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016. doi: 10.1214/15-AOS1388.

Robert E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, Extra Volume ISMP:107–121, 2012. doi: 10.4171/DMS/6/16.

Shreya Chaudhary, Jinglei Cheng, Samuel Kushnir, Jiaqi Leng, Pengyu Liu, Yuxiang Peng, Hanrui Wang, and Xiaodi Wu. Quantum-inspired hamiltonian descent for mixed-integer quadratic programming. In *NeurIPS 2025 Workshop on GPU-Accelerated and Scalable Optimization (ScaleOPT)*, 2025.

COIN-OR. CBC: Coin-or branch-and-cut solver, 2026. URL https://github.com/coin-or/Cbc. Open-source mixed-integer programming solver.

Gérard Cornuéjols, George L. Nemhauser, and Laurence A. Wolsey. The uncapacitated facility location problem. In P. B. Mirchandani and R. L. Francis (eds.), *Discrete Location Theory*, pp. 119–171. John Wiley & Sons, New York, 1990.

Gianni Di Pillo and Luigi Grippo. Exact penalty functions in constrained optimization. *SIAM Journal on Control and Optimization*, 27(6):1333–1360, 1989.

Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical Programming*, 98(1–3):23–47, 2003. doi: 10.1007/s10107-003-0395-5.

John Forrest and Robin Lougee-Heimer. CBC user guide. In *TutORials in Operations Research: Emerging Theory, Methods, and Applications*, pp. 257–277. INFORMS, 2005. doi: 10.1287/educ.1053.0020.

Robert Fortet. Applications de l'algèbre de boole en recherche opérationnelle. *Revue Française de Recherche Opérationnelle*, 4(14):17–26, 1960.

Giorgio Gallo, Peter L. Hammer, and Bruno Simeone. Quadratic knapsack problems. In Manfred W. Padberg (ed.), *Combinatorial Optimization*, volume 12 of *Mathematical Programming Studies*, pp. 132–149. Springer, Berlin, Heidelberg, 1980. doi: 10.1007/BFb0120892.

Zijie Geng, Jie Wang, Xijun Li, Fangzhou Zhu, Jianye Hao, Bin Li, and Feng Wu. Differentiable integer linear programming. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FPfCUJTsCn.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff 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(3):443–490, 2021. doi: 10.1007/s12532-020-00194-3.

Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. doi: 10.1145/227683.227684.

Qingyu Han, Linxin Yang, Qian Chen, Xiang Zhou, Dong Zhang, Akang Wang, Ruoyu Sun, and Xiaodong Luo. A GNN-guided predict-and-search framework for mixed-integer linear programming. In *The Eleventh International Conference on Learning Representations*, 2023.

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, and Juan Carrasquilla. Variational neural annealing. *Nature Machine Intelligence*, 3:952–961, 2021.

Christopher Hojny, Mathieu Besançon, Ksenia Bestuzheva, Sander Borst, João Dionísio, Johannes Ehls, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Adrian Göß, Alexander Hoen, Jacob von Holly-Ponientzietz, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Marco Lübbecke, Stephen J. Maher, Paul Matti Meinhold, Gioni Mexi, Til Mohr, Erik Mühmer, Krunal Kishor Patel, Marc E. Pfetsch, Sebastian Pokutta, Chantal Reinartz Groba, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Matthias Walter, Dieter Weninger, and Liding Xu. The SCIP optimization suite 10.0, 2025.

Qi Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01144.

Elias B. Khalil, Christopher Morris, and Andrea Lodi. MIP-GNN: A data-driven framework for guiding combinatorial solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10219–10227, 2022.

Jiaqi Leng, Ethan Hickman, Joseph Li, and Xiaodi Wu. Quantum hamiltonian descent, 2023.

Andrea Lodi. Mixed integer programming computation. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey (eds.), *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*, pp. 619–645. Springer, Berlin, Heidelberg, 2010. doi: 10.1007/978-3-540-68279-0_16.

Javier Lopez Piqueres and Jing Chen. Cons-training tensor networks: Embedding and optimization over discrete linear constraints. *SciPost Physics*, 18:192, 2025. doi: 10.21468/SciPostPhys.18.6.192.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.00712.

Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, 1990.

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. *arXiv preprint arXiv:2012.13349*, 2020. URL https://arxiv.org/abs/2012.13349.

Kouhei Nakaji, Lasse Bjørn Kristensen, Jorge A. Campos-Gonzalez-Angulo, Mohammad Ghazi Vakili, Haozhe Huang, Mohsen Bagherimehrab, Christoph Gorgulla, FuTe Wong, Alex McCaskey, Jin-Sung Kim, Thien Nguyen, Pooja Rao, et al. The generative quantum eigensolver (GQE) and its application for ground state search, 2024.

George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, NY, 1988. ISBN 9780471828198. doi: 10.1002/9781118627372.

Laurent Perron, Frédéric Didier, and Steven Gay. The CP-SAT-LP solver. In *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*, volume 280 of *Leibniz International Proceedings in Informatics*, pp. 3:1–3:2. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPIcs.CP.2023.3.

Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 5 edition, 2016. doi: 10.1007/978-3-319-26580-3.

W. David Pisinger, Anders Bo Rasmussen, and Rune Sandvik. Solution of large quadratic knapsack problems through aggressive reduction. *INFORMS Journal on Computing*, 19(2):280–290, 2007. doi: 10.1287/ijoc.1050.0172.

Edward Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007. doi: 10.1287/ijoc.1060.0189.

Thanawat Sornwanee. Differentiable integer linear programming is not differentiable & it's not a mere technical problem, 2026. URL https://arxiv.org/abs/2601.17800.

Paul Mingzheng Tang, Moses Hong De Lee, and Hoong Chuin Lau. Autoregressive rl approach for mixed-integer linear programs. In *Learning and Intelligent Optimization*, volume 15745 of *Lecture Notes in Computer Science*, pp. 1–17. Springer, 2025. doi: 10.1007/978-3-032-09192-5_1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.

A Appendix

A.1 Additional details on related works

Our experimental setup is directly inspired by the constrained tensor-network approach of Lopez Piqueres & Chen (2025), which introduces constrained matrix product states (MPS) whose block structure exactly restricts their support to binary configurations satisfying prescribed linear constraints. The authors optimize the resulting generative model using an unsupervised annealing procedure and evaluate it on the QKP, where they report improvements over SCIP on random instances. In fact, their QKP experiments form the basis of our experimental benchmark: following Lopez Piqueres & Chen (2025), we consider randomly generated integer-valued quadratic objectives and knapsack constraints and compare the quality of solutions obtained against SCIP under controlled computational budgets. The two approaches differ fundamentally, however, in how feasibility and optimization are represented. Constrained MPS encode the linear constraints exactly into the tensor-network architecture, so that samples have support only on the feasible set. This is a serious limitation of their method, as in general, exact encoding of linear constraints over binary variables via MPS requires exponentially many parameters. Our transformer does not hard-code the feasible set into its architecture; instead, it is first initialized on a feasible incumbent returned by SCIP and subsequently optimized using a differentiable objective together with a Lagrangian feasibility penalty. Moreover, while the constrained-MPS algorithm updates a probability distribution over feasible configurations through an annealed sample-and-train procedure, our method differentiates the relaxed QKP objective end-to-end through an autoregressive map $\theta \mapsto \tilde{x}_\theta$ and thereby performs optimization directly in the shared parameter space of the transformer. Thus, while our QKP benchmark and its motivation are explicitly inherited from Lopez Piqueres & Chen (2025), our method investigates a different mechanism for navigating the combinatorial landscape: differentiable autoregressive reparametrization, which has better scalability potential than exact constraint encoding via MPS.

The second conceptual connection is the Generative Quantum Eigensolver (GQE) of Nakaji et al. (2024). GQE departs from conventional variational quantum eigensolvers by placing the trainable parameters in a classical generative model rather than directly in a parameterized quantum circuit. In its GPT-QE realization, a transformer autoregressively generates sequences of quantum-circuit components, and the generative model is iteratively trained so that the circuits it produces attain lower energies. This perspective is closely related to the motivation of our method: in both cases, a difficult optimization problem is lifted from its original representation into the parameter space of a classical autoregressive generative model, whose shared parameters induce correlated changes in the generated object. The crucial difference is that GQE generates descriptions of quantum circuits whose energies must subsequently be evaluated, whereas our transformer directly generates the binary decision variables of the integer program through a differentiable relaxation. Consequently, our objective can be backpropagated through the generated variables to the transformer weights, making $\theta \mapsto \tilde{x}_\theta \mapsto f(\tilde{x}_\theta)$ an explicit differentiable reparametrization of the original optimization landscape. Our method therefore shares GQE's broader quantum-inspired idea of optimizing through a classical generative representation, but applies it entirely classically and directly to constrained discrete optimization, with the additional use of solver-incumbent initialization and Gumbel–softmax relaxation to obtain a differentiable, near-discrete optimization trajectory.

A complementary connection is Quantum Hamiltonian Descent (QHD) (Leng et al., 2023), which formulates continuous non-convex optimization as a time-dependent quantum Hamiltonian evolution. QHD is motivated in part by the ability of quantum dynamics to exploit trajectories that are inaccessible to classical local descent, including tunneling across barriers in the objective landscape. This motivation is closely related to our empirical tunneling analogy, but the underlying mechanisms are fundamentally different. QHD requires a quantum Hamiltonian evolution over the original continuous variables, whereas ADIP is entirely classical: it changes the optimization geometry by lifting the relaxed decision vector to the parameter space of an autoregressive transformer. In particular, our induced dynamics $\dot{\tilde{x}} = J_\theta J_\theta^\top \nabla_{\tilde{x}} F$ provide a classical, state-dependent reparametrization of the gradient field, while the observed basin crossing arises from optimization through this neural representation together with stochastic near-discrete exploration.

Recent concurrent work on Quantum-Inspired Hamiltonian Descent (QIHD) (Chaudhary et al., 2025) similarly transfers Hamiltonian intuition to a classical optimizer for mixed-integer quadratic programs. QIHD explicitly simulates momentum-position Hamiltonian dynamics with constraint

penalties, whereas ADIP obtains non-Euclidean decision-space dynamics implicitly through the Jacobian of an autoregressive neural reparametrization.

A.2 Experimental details

Compute environment. All reported runs were executed on a single Apple MacBook Pro with an Apple M4 Max CPU (14 cores: 10 performance + 4 efficiency) and 36 GB unified memory. Training used CPU-only PyTorch; no GPU acceleration was used as transformers used are small and GPU overheads dominate here. Warm-start solvers were restricted to a single thread. BLAS/OpenMP thread counts were likewise set to 1 for reproducibility. Wall-clock timing was used for SCIP, HiGHs, CP-SAT and CBC with fixed random seeds.

Instance generation. As in the main text, for each size $n \in \{400, 1000, 10000\}$, for each experiment, we sample $Q_{ij} \sim \mathrm{Unif}\{-5, \dots, 5\}$ (integer, nonsymmetric), $w_i \sim \mathrm{Unif}\{0, \dots, 5\}$, and capacity $C = \frac{1}{4}\sum_i w_i$. Instances are identified by a base seed. The generation code is released in the repository.

Warm-start models. SCIP solves the native nonconvex quadratic integer program (quadratic epigraph). HiGHS and CBC receive a Fortet/McCormick linearization of the binary quadratic, introducing an auxiliary binary y_{ij} for each nonzero product term together with $y_{ij} \le x_i$, $y_{ij} \le x_j$, $y_{ij} \ge x_i + x_j - 1$. CP-SAT uses the same auxiliaries but encodes $y_{ij} = x_i x_j$ via OR-Tools `AddMultiplicationEquality` (equivalent on $\{0,1\}^n$). For dense indefinite Q, the linearized models have $O(n^2)$ binaries and constraints, which becomes severe already at $n = 1000$ for LP-based MIP solvers; CP-SAT's SAT/CP search often returns a feasible incumbent sooner under the same 500 s budget.

Matched-budget protocol. Each base solver is allotted $T_{\mathrm{seed}} = 500\,\mathrm{s}$ to produce x^{inc}. ADIP then runs with a Phase-2 wall-clock cap $T_{\mathrm{train}} = 500\,\mathrm{s}$ (and at most $K_2 = 2000$ Adam steps, with early stopping on plateau). The matched baseline continues the same solver alone for $T_{\mathrm{total}} = T_{\mathrm{seed}} + T_{\mathrm{train}} = 1000\,\mathrm{s}$. We report relative improvement

$$\mathrm{Imp}_{\mathrm{time}}(\%) = 100 \cdot \frac{f(x^{\mathrm{ADIP}}) - f(x^{\mathrm{OSS}}_{T_{\mathrm{total}}})}{|f(x^{\mathrm{OSS}}_{T_{\mathrm{total}}})|},$$

and write ∞ when the matched solver objective is identically zero while ADIP returns a strictly positive feasible objective (common for linearized solvers at large n, and for SCIP at $n = 10000$ under our budgets).

Architecture and optimization. The model is a two-layer causal transformer over the binary alphabet with embedding dimension $d = 64$, 4 heads, feed-forward width $d_{\mathrm{ff}} = 256$, and banded sparse attention retaining a fraction 0.1 of previous positions. Phase 1 minimizes the smoothed teacher-forcing CE equation 7 with Adam, $K_1 = 150$ maximum steps, learning rate $\eta_1 = 10^{-3}$, plateau patience $P_1 = 12$ and minimum delta $\delta_1 = 10^{-5}$, and target smoothing $\varepsilon = 0.001$. Phase 2 maximizes the soft Lagrangian equation 8 with Adam, Gumbel–softmax straight-through estimation at temperature $\tau = 0.5$, soft weight $w_{\mathrm{soft}} = 1$, and plateau patience $P_2 = 45$ (minimum delta $\delta_2 = 10^{-5}$). Size-dependent Phase 2 knobs are:

n	Phase-2 LR η_2	knapsack penalty λ
400	10^{-3} (same as η_1)	100
1000, 10000	10^{-4}	500

At every Phase 2 step we hard-decode $\hat{x}$, test knapsack feasibility, and retain the best feasible objective encountered. Phase-1 CE uses a one-pass implementation; Phase-2 rollouts use KV caching with chunked gradient checkpointing to keep long sequences tractable.

Objective-value trajectory (Fig. 2). Figure 2 compares the matched SCIP-alone curve over 1000 s with the SCIP+T_{train} ADIP curve. The plot uses a broken y-axis (near-zero band and high-objective zoom) and a late-time expanding x-scale $-\log(T - t)$ to emphasize improvements near the end of the budget.

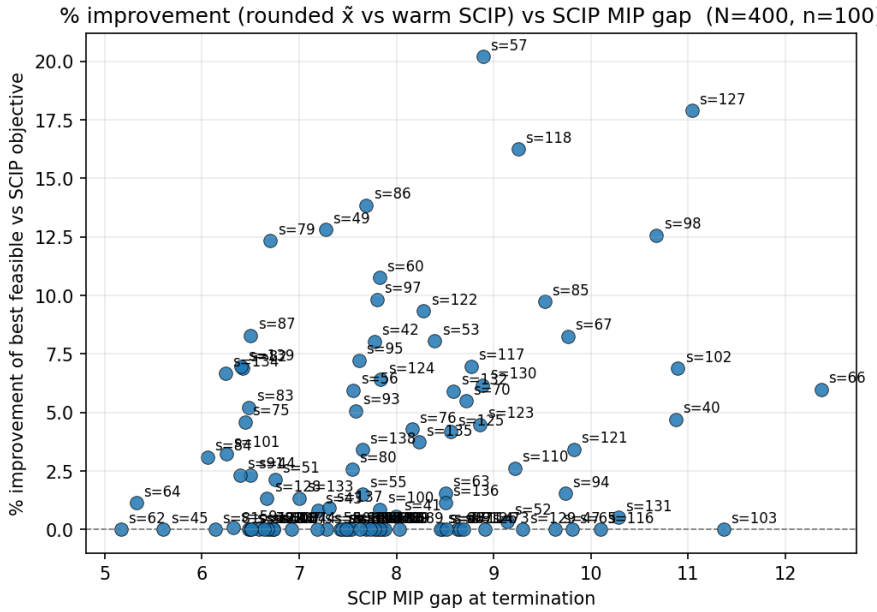

Figure 4: Percentage improvement of SCIP+ADIP vs. matched SCIP, plotted against the SCIP MIP gap after $T_{\text{seed}} = 500\,\text{s}$ ($n = 400$, 100 seeds).

Basin / tunneling diagnostic (Sec. 5.3, Fig. 3). Along a recorded ADIP trajectory, from each point (and from SCIP incumbent snapshots), we run projected gradient ascent of the continuous objective subject to the knapsack boundary ($w^\top \tilde{x} \le C$), with step size 0.05, at most 3000 iterations, and tolerances 10^{-6} (capacity) / 10^{-8} (projected gradient). Two endpoints are counted as distinct if their ℓ_2 distance exceeds 10^{-3}. Table 2 reports trajectory length, the number of distinct endpoints, and the objective at the incumbent-started vs. best endpoint. The 3D visualization in Fig. 3 is a t-SNE projection of the same points (perplexity 30) with objective on the vertical axis; basin identity is always decided in the original $[0, 1]^n$ space, not in the t-SNE embedding.

Software. Experiments used Python ≥ 3.10, NumPy < 2, PyTorch ≥ 2, `pyscipopt` / SCIP 10, HiGHS (`highspy`), and OR-Tools (CBC + CP-SAT). Pinned source hashes and reproduction commands are included in the repository.

Additional experimental figure. Figure 4 plots, for the SCIP $n = 400$ matched-budget sweep (100 seeds), the percentage improvement of SCIP+ADIP over the matched SCIP objective against the SCIP MIP gap at T_{seed}. Larger residual gaps after the seed phase tend to leave more room for ADIP improvement, consistent with Sec. 5.2, but the gap is not a deterministic predictor of ADIP success.

A.3 Proof of Theorem 1

Membership in NP is immediate: given a candidate $x \in 0, 1^n$, one can verify both feasibility and strict improvement in polynomial time.

For NP-hardness, consider the decision version of QKP. Given $Q \in \mathbb{Z}^{n\times n}$, strictly positive weights $w \in \mathbb{Z}^n_{>0}$, capacity $C \in \mathbb{Z}_{>0}$, and an integer threshold $K \ge 1$, the problem asks whether there exists $x \in 0, 1^n$ such that

$$w^\top x \le C, \qquad x^\top Q x \ge K. \tag{14}$$

This decision problem is NP-hard (Gallo et al., 1980; Pisinger et al., 2007).

Construct an augmented instance by introducing one additional binary variable z. Give this variable weight C and define

$$Q' = \begin{pmatrix} Q & 0 \\ 0 & K-1 \end{pmatrix}, \qquad w' = (w\ C)\,. \tag{15}$$

The capacity remains C. Consider the explicitly known incumbent

$$x^{\mathrm{inc}} = (0, \ldots, 0, 1). \tag{16}$$

It is feasible and has objective value $K - 1$.

Because all original weights are strictly positive, any feasible solution with $x_{n+1} = 1$ must set every original variable to zero: the additional variable alone exhausts the entire capacity. Hence no feasible solution with $x_{n+1} = 1$ can strictly improve upon x^{inc}. A strictly improving solution must therefore have $x_{n+1} = 0$, in which case its objective is precisely $x^\top Q x$. Since the objective is integer-valued,

$$x^\top Q x > K - 1 \quad \Longleftrightarrow \quad x^\top Q x \geq K. \tag{17}$$

Thus the augmented instance admits a feasible solution strictly improving upon x^{inc} if and only if the original quadratic knapsack decision instance is a YES instance. The construction is polynomial, which proves NP-hardness.

A.4 Proof of Proposition 1

Fix an arbitrary $x^\star \in \{0, 1\}^n$. Consider the constant sequence-to-sequence map

$$f_{x^\star}(z) = x^\star \tag{18}$$

on any compact input domain $\mathcal{K}$. This map is continuous. By the universal approximation theorem for transformers with positional encodings (Yun et al., 2020), for every $\varepsilon > 0$ there exists a transformer $T_\theta \in \mathcal{T}$ such that

$$\sup_{z \in \mathcal{K}} \|T_\theta(z) - f_{x^\star}(z)\|_\infty < \varepsilon. \tag{19}$$

In particular, there is an input initiating the autoregressive rollout yielding,

$$\|\tilde{x}_\theta - x^\star\|_\infty < \varepsilon. \tag{20}$$

Choose $\varepsilon < 1/2$. If $x_i^\star = 0$, then

$$\tilde{x}_{\theta,i} < \varepsilon < 1/2, \tag{21}$$

whereas if $x_i^\star = 1$, then

$$\tilde{x}_{\theta,i} > 1 - \varepsilon > 1/2. \tag{22}$$

Thresholding each coordinate at $1/2$ therefore yields

$$\hat{x}_{\theta,i} = x_i^\star \tag{23}$$

for every i. Since $x^\star$ was arbitrary, the hard-decoded transformer parametrization covers the entire binary hypercube $\{0, 1\}^n$.

A.5 Proof of Corollary 1

By Proposition 1,

$$\{\hat{x}_\theta : \theta \in \Theta\} = \{0, 1\}^n. \tag{24}$$

Therefore optimization over the hard-decoded transformer contains the original binary quadratic program exactly:

$$\max_{\theta:, A\hat{x}_\theta \leq b} \hat{x}_\theta^\top Q \hat{x}_\theta = \max_{x \in 0,1^n:, Ax \leq b} x^\top Q x. \tag{25}$$

Since the decision version of the problem on the right-hand side is NP-hard, so is global optimization over θ.

A.6 Proof of Proposition 2

Because A, b, and x are integer-valued, $Ax - b$ is integer-valued. Hence if a binary vector x is infeasible, at least one constraint is violated by at least one unit, and therefore

$$\Phi(x) \geq 1. \tag{26}$$

Furthermore, since $x_i x_j \in 0, 1$,

$$x^\top Q x = \sum_{i,j} Q_{ij} x_i x_j \leq \sum_{i,j} \max Q_{ij}, 0 = U_Q. \tag{27}$$

Thus every infeasible binary x satisfies

$$x^\top Q x - \lambda \Phi(x) \leq U_Q - \lambda < f_{\text{inc}}, \tag{28}$$

where the final inequality follows from the condition in Proposition 2.

Since x^{inc} is feasible, $\Phi(x^{\text{inc}}) = 0$, and its penalized objective is exactly f_{inc}. Therefore no infeasible binary point can globally maximize the penalized objective.

A.7 Proof of Proposition 3

The almost-sure convergence follows directly from the zero-temperature limit of the sigmoid:

$$\sigma\left(\frac{v}{\tau}\right) \longrightarrow \mathbf{1}_{v>0} \tag{29}$$

for every $v \neq 0$.

Let $s'(u) = \sigma(u)(1 - \sigma(u))$. Since $V = \Delta + G$ has density $q(v - \Delta)$,

$$\mathbb{E}[D_\tau] = \int_{\mathbb{R}} \frac{1}{\tau} s'\left(\frac{v}{\tau}\right) q(v - \Delta), dv. \tag{30}$$

Using the substitution $v = \tau u$ gives

$$\mathbb{E}[D_\tau] = \int_{\mathbb{R}} s'(u) q(\tau u - \Delta), du. \tag{31}$$

By dominated convergence and $\int_{\mathbb{R}} s'(u), du = 1$,

$$\mathbb{E}[D_\tau] \longrightarrow q(-\Delta). \tag{32}$$

Similarly,

$$\mathbb{E}[D_\tau^2] = \int_{\mathbb{R}} \frac{1}{\tau^2} s'\left(\frac{v}{\tau}\right)^2 q(v - \Delta), dv \qquad = \frac{1}{\tau} \int_{\mathbb{R}} s'(u)^2 q(\tau u - \Delta), du. \tag{33}$$

Since

$$\int_{\mathbb{R}} s'(u)^2, du = \int_0^1 t(1 - t), dt = \frac{1}{6}, \tag{34}$$

we obtain

$$\mathbb{E}[D_\tau^2] = \frac{q(-\Delta)}{6\tau} + o(\tau^{-1}). \tag{35}$$

Because $\mathbb{E}[D_\tau]$ converges to the finite value $q(-\Delta)$, the second moment dominates asymptotically, yielding $\text{Var}(D_\tau) = \Theta(\tau^{-1})$.

A.8 Proof of Theorem 4

We start by stating a formal version of the theorem.

Consider one attention head with input representation H, query, key, and value matrices

$$Q = HW_Q, \qquad K = HW_K, \qquad V = HW_V, \tag{36}$$

and causally masked attention

$$\text{Attn}(H) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right) V, \tag{37}$$

where M is the causal mask.

Theorem 4 (Path-dependent induced geometry)**.** *Let θ be the parameters of a causal transformer and suppose that, for some attention head, the output of the relaxed autoregressive rollout depends nontrivially on at least one of its query or key matrices, i.e.*

$$J_Q := \frac{\partial \tilde{x}_\theta}{\partial W_Q} \neq 0 \qquad \textit{or} \qquad J_K := \frac{\partial \tilde{x}_\theta}{\partial W_K} \neq 0. \tag{38}$$

For every $c > 0$, define a reparametrized transformer $\theta^{(c)}$ by

$$W_Q^{(c)} = cW_Q, \qquad W_K^{(c)} = c^{-1}W_K, \tag{39}$$

with every other parameter unchanged.

Then:

1. *the two transformers define exactly the same autoregressive map,*
$$\tilde{x}_{\theta^{(c)}}(z) = \tilde{x}_\theta(z) \qquad \textit{for every input or autoregressive prefix } z; \tag{40}$$
2. *nevertheless, except in the degenerate case $J_Q = J_K = 0$, there exists $c \neq 1$ for which*
$$J_{\theta^{(c)}} J_{\theta^{(c)}}^\top \neq J_\theta J_\theta^\top. \tag{41}$$
3. *there exists a QKP objective for which the two transformers induce different instantaneous velocities in relaxed decision space*
$$\dot{\tilde{x}}_\theta \neq \dot{\tilde{x}}_{\theta^{(c)}}. \tag{42}$$

In other words there exist two transformers representing exactly the same binary solution but inducing different local optimization geometries in relaxed decision space.

Proof. Under the rescaling stated in Theorem 4, the query and key matrices become

$$Q^{(c)} = H(cW_Q) = cQ \tag{43}$$

and

$$K^{(c)} = H(c^{-1}W_K) = c^{-1}K. \tag{44}$$

Their product is therefore invariant:

$$Q^{(c)}(K^{(c)})^\top = (cQ)(c^{-1}K)^\top = QK^\top. \tag{45}$$

Hence the matrix entering the causally masked attention softmax is unchanged. Since W_V and every subsequent model parameter are also unchanged, the output of this attention head is identical. Applying the same argument throughout the autoregressive rollout shows that

$$\tilde{x}_{\theta^{(c)}}(z) = \tilde{x}_\theta(z) \tag{46}$$

for every prefix z. This proves the first statement.

We now examine the parameter Jacobian. The attention scores depend on W_Q and W_K through

$$W_Q W_K^\top. \tag{47}$$

For a perturbation δW_Q around the reparametrized point,

$$\delta\left(W_Q^{(c)}(W_K^{(c)})^\top\right) = \delta W_Q (c^{-1}W_K)^\top, \tag{48}$$

so the sensitivity to W_Q is multiplied by c^{-1}. Similarly, the sensitivity to W_K is multiplied by c. Consequently,

$$J_Q^{(c)} = c^{-1}J_Q, \qquad J_K^{(c)} = cJ_K. \tag{49}$$

Let J_{rest} denote the Jacobian with respect to all remaining parameters. Since the forward computation is unchanged under this rescaling, their contribution is unchanged. Hence

$$J_{\theta^{(c)}} J_{\theta^{(c)}}^\top = J_{\text{rest}} J_{\text{rest}}^\top + c^{-2} J_Q J_Q^\top + c^2 J_K J_K^\top. \tag{50}$$

At $c = 1$,

$$J_\theta J_\theta^\top = J_{\text{rest}} J_{\text{rest}}^\top + J_Q J_Q^\top + J_K J_K^\top. \tag{51}$$

Suppose, for contradiction, that these two matrices were equal for every $c > 0$. Then

$$(c^{-2} - 1) J_Q J_Q^\top + (c^2 - 1) J_K J_K^\top = 0 \tag{52}$$

for every $c > 0$. Taking $c \to \infty$ implies

$$J_K J_K^\top = 0, \tag{53}$$

and therefore $J_K = 0$. Taking $c \to 0$ similarly yields $J_Q = 0$. This contradicts the nondegeneracy assumption in Theorem 4.

Therefore, whenever the attention head affects the generated solution nontrivially, there exists $c \neq 1$ such that

$$J_{\theta^{(c)}} J_{\theta^{(c)}}^\top \neq J_\theta J_\theta^\top. \tag{54}$$

Combining Eq. 54 with Theorem 2, the two functionally equivalent initializations satisfy

$$\dot{\tilde{x}}_\theta = K_\theta \nabla_{\tilde{x}} F_\lambda(\tilde{x}), \tag{55}$$

$$\dot{\tilde{x}}_\theta^{(c)} = K_{\theta^{(c)}} \nabla_{\tilde{x}} F_\lambda(\tilde{x}), \tag{56}$$

where

$$K_\theta \neq K_{\theta^{(c)}} \tag{57}$$

for an appropriate c. Thus two transformers representing the same current solution can induce different trajectory kernels solely because of how that solution is encoded in weight space.

We next show that there exists a quadratic objective

$$f(\tilde{x}) = \tilde{x}^\top Q \tilde{x} \tag{58}$$

for which the two transformers induce different instantaneous trajectories:

$$K_\theta \nabla f(\tilde{x}) \neq K_{\theta^{(c)}} \nabla f(\tilde{x}). \tag{59}$$

Let

$$D := K_\theta - K_{\theta^{(c)}}. \tag{60}$$

Since $D \neq 0$, there exists some $g \in \mathbb{R}^n$ such that

$$Dg \neq 0. \tag{61}$$

We show that g can be realized as the gradient of a quadratic objective at $\tilde{x}$. Let

$$r = \tilde{x}^\top \tilde{x} > 0 \tag{62}$$

and define the symmetric matrix

$$S = \frac{g\tilde{x}^\top + \tilde{x} g^\top}{r} - \frac{\tilde{x}^\top g}{r^2} \tilde{x}\tilde{x}^\top. \tag{63}$$

A direct calculation gives

$$S\tilde{x} = g. \tag{64}$$

Set

$$Q = \frac{1}{2} S. \tag{65}$$

Because S is symmetric,

$$\nabla_{\tilde{x}} \left(\tilde{x}^\top Q \tilde{x} \right) = (Q + Q^\top)\tilde{x} = S\tilde{x} = g. \tag{66}$$

Therefore

$$\dot{\tilde{x}}_\theta - \dot{\tilde{x}}_{\theta^{(c)}} = (K_\theta - K_{\theta^{(c)}})\, g = Dg \neq 0. \tag{67}$$

Thus the different geometries produced by the self-attention symmetry can lead to genuinely different trajectories for an objective of exactly the quadratic form considered in this proof.

□