CONSTRAINED GRAPH DIFFUSION FOR MIXED-INTEGER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes a novel learning-based approach to approximately solve instances of mixed-integer optimization problems. These problems are computationally challenging, as they require jointly determining discrete and continuous decisions while satisfying complex combinatorial constraints. We introduce Constrained Graph Diffusion (CGD), a learning-based framework that approximately solves recurring instances of such problems by learning a conditional distribution over their discrete decisions. CGD uses a graph-based diffusion model and incorporates constraint information directly into the reverse diffusion process, steering intermediate predictions toward the feasible region throughout generation. By operating on continuous relaxations of the discrete variables, CGD defines a differentiable constrained generation pathway up to terminal discrete recovery. Once the discrete decision is recovered and fixed, a numerical optimizer solves the remaining continuous problem, avoiding online combinatorial search over the binary variables while retaining numerical optimization for continuous completion. We evaluate CGD on AC-OPF with branch switching and discrete portfolio optimization, demonstrating substantial improvements in feasibility and solution quality over learning-based baselines while achieving speedups of up to $543\times$ over state-of-the-art MIP solvers on large instances.

1 INTRODUCTION

Mixed-integer optimization provides a powerful framework for modeling problems involving both discrete and continuous variables. Applications arise in a wide range of domains, including power systems (Hedman et al., 2010), logistics (Amiri, 2006) and finance (Bienstock, 1996).

This work considers problems of the form

$$\begin{aligned} \underset{x,z}{\text{Minimize}} \quad & f(x,z) \quad \text{s.t.} \quad g(x,z) \leq 0, \\ & x \in \mathcal{X} \subseteq \mathbb{R}^n, \quad z \in \{0,1\}^m, \end{aligned} \tag{1}$$

where $x \in \mathcal{X}$ are the *continuous* decision variables, z the *binary* decision variables, f is the objective, and g collects the problem constraints, with equality constraints represented as pairs of inequalities. In the most general case, Problem (1) is a nonconvex mixed-integer nonlinear program. In such problems, two distinct sources of difficulty compound: *(1) Combinatorial structure:* the binary variables z induce a search space of size 2^m, forcing exact methods to branch over an exponential number of assignments; *(2) Nonconvexity:* the objective f and the constraints g are nonconvex, so even a single fixed relaxation is a nonconvex program with no guarantee of global optimality.

When instances of Problem (1) share the same structure, Learning to Optimize can amortize solution effort across instances (Bengio et al., 2021; Kotary et al., 2021). Existing approaches either improve decisions within conventional solvers (Khalil et al., 2016; Gasse et al., 2019; Nair et al., 2020) or directly predict solutions through learned surrogates and differentiable optimization layers (Wilder et al., 2019a; Ferber et al., 2020). The former retain online combinatorial search, whereas the latter may overlook multiple near-optimal assignments and produce fractional or infeasible decisions.

The proposed approach learns instead a conditional distribution over the discrete variables. Once an assignment $\hat{z}$ is fixed, a numerical optimizer recovers the continuous variables x, avoiding online combinatorial search without eliminating the potential nonconvexity of the continuous subproblem.

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.

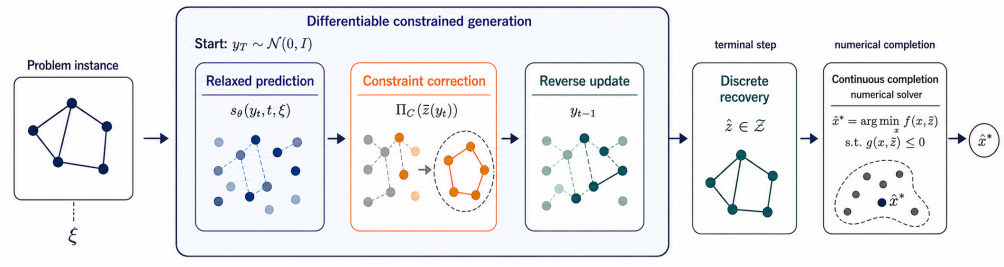


Figure 1: Overview of the generation phase of CGD. At each reverse step t, the denoiser produces a relaxed binary decision, whose Monte Carlo estimate $\bar{z}_t$ is corrected with respect to the relaxed feasible set $\mathcal{C}_\xi$. The resulting correction is incorporated into the reverse update. Terminal recovery produces the binary decision $\hat{z}$, while a numerical solver recovers the continuous variables $\hat{x}$.

Diffusion models naturally represent such multimodal distributions and have been applied to graph-structured problems (Sun & Yang, 2023; Sanokowski et al., 2024). However, existing diffusion solvers do not natively enforce combinatorial constraints during generation, although an invalid discrete assignment may render the downstream continuous problem infeasible, as shown in Fig. 2.

Motivated by this gap, we introduce *Constrained Graph Diffusion* (CGD), a diffusion framework that incorporates feasibility corrections directly into the reverse denoising process. CGD operates in a differentiable fashion over relaxed binary decisions, allowing each correction to influence all subsequent denoising steps and steer generation toward the feasible set. Hard discretization is then deferred to the terminal step, after which the recovered binary decision is fixed and the remaining continuous variables are obtained using a standard numerical solver, ensuring validity (Fig. 1).

Contributions. The paper makes the following contributions: **(1)** It introduces Constrained Graph Diffusion (CGD), a novel end-to-end differentiable generation framework over relaxed discrete decisions. It uses application-specific feasibility corrections to steer the reverse trajectory, with hard recovery deferred to the terminal step. **(2)** It proposes a learning-assisted decomposition that generates the discrete variables and recovers the continuous variables through a downstream numerical optimization problem, thereby avoiding online combinatorial search, while retaining solution validity. **(3)** It evaluates CGD on AC optimal power flow with branch switching and mixed-integer portfolio optimization. Across these highly nonlinear problems, CGD improves feasibility and solution quality over learning-based baselines while achieving substantial speedups over MIP solvers.

2 Related Work

Our work lies at the intersection of learning for mixed-integer optimization and diffusion models for combinatorial optimization. We summarize the closest connections here and provide a comprehensive methodological review in Appendix B.

Learning for Mixed-Integer Optimization. Learning-based methods can accelerate repeated optimization by predicting solutions from instance data (Bengio et al., 2021; Kotary et al., 2021). One line of work learns components of conventional solvers, including branching, primal heuristics, and cut selection (Khalil et al., 2016; Gasse et al., 2019; Nair et al., 2020; Tang et al., 2020; Kannan et al., 2025). These methods preserve the solver's feasibility and certification mechanisms but retain online combinatorial search. More direct approaches predict solutions or optimization strategies (Bertsimas & Stellato, 2022; Wilder et al., 2019a; Ferber et al., 2020; Tang et al., 2025); however, they commonly return a single assignment or rely on relaxations, rounding, repair, or downstream optimization. CGD instead learns a distribution over discrete variables, avoiding online branch-and-bound while retaining numerical optimization for continuous completion.

Diffusion Models for Combinatorial Optimization. Diffusion models have produced high-quality solutions for graph-structured combinatorial problems (Sun & Yang, 2023; Sanokowski et al., 2024; Hong et al., 2024; Feng et al., 2026; Lei et al., 2025). Existing approaches typically address constraints through objective penalties, guidance, or post-processing. CGD instead incorporates feasibility corrections throughout reverse diffusion, allowing constraint information to shape the generative trajectory before terminal discrete recovery and continuous completion.

3 Problem Setting and Learning-Assisted Decomposition

The proposed approach exploits the fact that the combinatorial difficulty of Problem (1) lies in determining its binary decision variables z. Once these variables are fixed (we write $z = \hat{z}$), Problem (1) reduces to the continuous optimization problem

$$\min_{x} \quad f(x, \hat{z}) \quad \text{s.t.} \quad g(x, \hat{z}) \leq 0, \quad x \in \mathcal{X}. \tag{2}$$

Although this problem retains the original nonlinearity and nonconvexity, it contains no combinatorial search and can be solved with a standard numerical solver at significantly lower cost.
We collect the constraints involving only the binary variables in

$$\mathcal{Z} = \{z \in \{0,1\}^m \mid h(z;\xi) \leq 0\}.$$

For example, in branch switching, $\mathcal{Z}$ contains the line configurations satisfying the required connectivity conditions. These constraints can be enforced while generating z, independently of the downstream continuous solve. Given training data $\mathcal{D} = (\xi_i, z_i^\star)_{i=1}^N$, where ξ_i denotes an instance and $z_i^\star \in \mathcal{Z}$ is a high-quality reference decision, we learn a conditional generative model that produces a feasible $\hat{z} \in \mathcal{Z}$ for an unseen instance ξ. The continuous variables are then recovered by solving (2) with $\hat{z}$ fixed. This decomposition eliminates the combinatorial search while retaining numerical optimization for continuous completion.

The next section reviews the diffusion framework that forms the basis of our approach and exposes the feasibility gap that motivates the proposed constrained diffusion model.

4 Diffusion Background and the Feasibility Gap

Diffusion models generate samples by learning to reverse a noising process that maps data to a simple prior (Song & Ermon, 2019; Song et al., 2021). Let $y_0 \sim p_{\text{data}}$ denote a sample in $\mathbb{R}^d$. The forward diffusion process progressively perturbs y_0 through a sequence of Markov transitions parameterized by a noise schedule $\{\beta_t\}_{t=1}^T$. A key property of this process is that the conditional distribution $q(y_t \mid y_0)$ admits the closed-form representation $y_t = \sqrt{\bar{\alpha}_t}\, y_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$, with $\bar{\alpha}_t = \prod_{i=1}^t (1-\beta_i)$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, with y_t interpolating between data and noise.

The reverse process is parameterized by a denoising network $s_\theta(y_t, t, \xi)$. The denoiser may parameterize the reverse process through a noise estimate or directly through a relaxed estimate of the clean binary state. In the former case, the denoiser parameters are optimized by minimizing the error between the true noise ϵ and the predicted noise:

$$\min_{\theta} \; \mathbb{E}_{t,\, p_{\text{data}}(y_0),\, \mathcal{N}(\epsilon;\mathbf{0},\mathbf{I})} \left[\left\| \epsilon - s_\theta\big(y_t, t, \xi\big) \right\|_2^2 \right] \tag{3}$$

where y_t is the noisy sample obtained from the forward diffusion process at timestep t. Sampling starts from $y_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iterates as $y_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \left(y_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} s_\theta(y_t, \xi, t) \right) + \sigma_t \eta$, with $\eta \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, down to $t = 0$, mapping the diffusion prior toward the data distribution. For the constrained generation mechanism developed next, the essential quantity is the relaxed estimate $\hat{z}_t \in [0,1]^m$ of the binary decisions produced at each reverse step. This representation provides the interface between the diffusion model and the feasibility-correction operator.

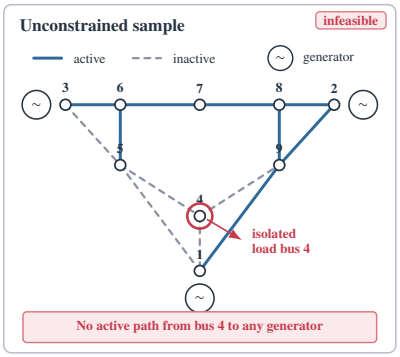


Figure 2: Infeasible 9-bus topology. Opening (1,4), (4,5), and (4,9) isolates load bus 4.

4.1 Limitations of Diffusion Models for Combinatorial Decisions

Diffusion models can learn distributions over discrete decisions but do not natively enforce application-specific constraints (Christopher et al., 2024). This is key because in mixed-integer optimization, even a small discrete error may render the downstream continuous problem infeasible. The AC OPF with branch switching provides a representative example (see Section 6 for details). The binary vector $z \in \{0,1\}^m$ specifies active lines,

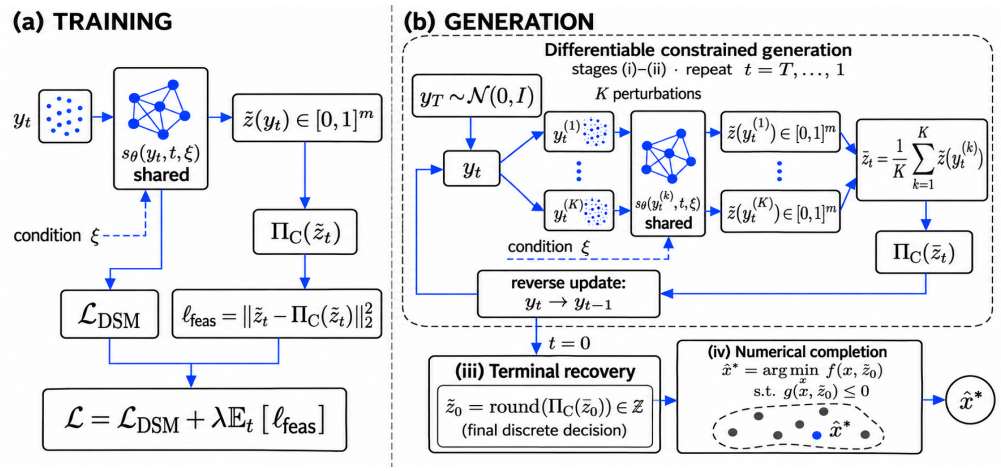


Figure 3: CGD training and generation. **(a)** During training, the projected relaxed decision defines the feasibility target used to regularize the diffusion objective. **(b)** During generation, K perturbed predictions are aggregated into a Monte Carlo estimate used for constraint guidance throughout stages (i)–(ii), followed by discrete recovery and continuous completion in stages (iii)–(iv).

which must connect each load bus to generation. In Fig. 2, opening the three lines incident to bus 4 isolates its load, making the downstream AC-OPF infeasible (Babaeinejadsarookolaee et al., 2019). This motivates enforcing combinatorial feasibility during reverse denoising.

5 Constrained Graph Diffusion for Combinatorial Decisions

This section develops Constrained Graph Diffusion (CGD) to generate a high-quality binary decision satisfying the combinatorial requirements of an unseen instance ξ. CGD couples a conditional graph diffusion model with a differentiable optimization layer implementing a projection operator. The projection plays two key roles: during training, it defines a feasibility regularizer for the denoiser expected relaxed prediction, as illustrated in Figure 3(a); during generation, it provides a plug-and-play mechanism that corrects the relaxed decision throughout the diffusion trajectory. We defer the complete training procedure to Appendix A.4 and focus next on the generation procedure, where constraint information is incorporated throughout the reverse diffusion trajectory.

Framework overview. At generation time, CGD consists of the following four stages. *(1)* Given an unseen problem instance ξ, the graph-based denoiser s_θ starts from Gaussian noise and estimates a relaxed representation of the binary decision variables. *(2)* At every reverse step, a Monte Carlo estimate of this prediction is projected onto the (relaxed) constrained set and reinserted into the reverse update, steering subsequent predictions toward the application-specific combinatorial constraints. *(3)* After the terminal reverse step, the projected estimate is discretized to obtain the binary decision $\hat{z}$. *(4)* Finally, $\hat{z}$ is fixed in the original mixed-integer problem (1), reducing it to Problem (2), which is solved with a numerical optimizer to recover the continuous decision $\hat{x}$.

5.1 Constraint-Aware Generation

Given an unseen problem instance ξ, the goal of the reverse diffusion process is to generate a feasible binary decision vector $\hat{z} \in \mathcal{Z}_\xi$. CGD realizes this goal through a new plug-and-play mechanism for constrained generation, illustrated schematically in Figure 3(b). This subsection develops stages (1) and (2) described above, while Section 5.2 presents stages (3) and (4).

Stage(1): graph-conditioned relaxed decoding. The central constraint-enforcement operator in CGD is a differentiable optimization step (a projection). Notice that the constraint-correction operator acts on a relaxed decision in $[0,1]^m$, whereas the reverse diffusion process evolves in the noisy continuous state y_t. Stage (1) therefore maps the denoiser output to a continuous relaxed representation of the binary decisions on which the projection can act. A binary decision $z \in \{0,1\}^m$ vector is represented in the diffusion space through $e(z) = 2z - 1 \in \{-1, 1\}$ with $e^{-1}(y) = \frac{y+1}{2}$, which preserves Hamming geometry since $\|e(z) - e(z')\|_2^2 = 4d_H(z, z')$. At reverse timestep t, CGD

considers K independently perturbed diffusion states $\{y_t^{(k)}\}$ for the same problem instance ξ and noise level. The denoiser $s_\theta(y_t^{(k)}, t, \xi)$ maps each state to a relaxed estimate of the binary decision,

$$\hat{z}_t^{(k)} = \hat{z}_\theta(y_t^{(k)}, t, \xi) = \frac{1}{2}\left[\tanh\left(\frac{y_t^{(k)} - \sqrt{1-\bar{\alpha}_t}\, s_\theta(y_t^{(k)}, t, \xi)}{\sqrt{\bar{\alpha}_t}}\right) + 1\right] \in [0,1]^m. \tag{4}$$

These K predictions are aggregated through a Monte Carlo estimate $\bar{z}_t = \frac{1}{K}\sum_{k=1}^{K} \hat{z}_t^{(k)}$, which reduces variance and provides a more stable estimate of the current relaxed decision on which constraint guidance can act (Owen, 2013, §2).

Stage(2): Constraint correction. Stage (1) produces a relaxed prediction $\bar{z}_t \in [0,1]^m$, but this prediction need not satisfy the combinatorial constraints of instance ξ. Stage (2) therefore uses constraint information to correct $\bar{z}_t$ before reinserting it into the reverse diffusion process. More formally, given an instance ξ, its associated discrete feasible set is $\mathcal{Z}_\xi = \{z \in \{0,1\}^m : h(z;\xi) \le 0\}$; direct projection onto $\mathcal{Z}_\xi$ would require solving a combinatorial problem at every reverse step. CGD instead uses the continuous relaxation:

$$\mathcal{C}_\xi = \{u \in [0,1]^m : \widetilde{h}(u;\xi) \le 0\} \qquad \mathcal{C}_\xi \cap \{0,1\}^m = \mathcal{Z}_\xi. \tag{5}$$

To project this Monte Carlo prediction $\bar{z}_t$ onto C_ξ, CGD uses the Euclidean projection

$$\Pi_{\mathcal{C}_\xi}(v) \in \arg\min_{u \in \mathcal{C}_\xi} \|u - v\|_2^2. \tag{6}$$

At timestep t, the Monte Carlo prediction $\bar{z}_t$ is corrected as $u_t = \Pi_{\mathcal{C}_\xi}(\bar{z}_t)$.The projection leaves a feasible prediction unchanged or makes the smallest Euclidean correction. Next, we use the discrepancy between the Monte Carlo prediction and its corrected counterpart to define a guidance

$$\ell_{\text{guide}}(t, \xi) = \|\bar{z}_t - \text{sg}[u_t]\|_2^2, \tag{7}$$

whose gradient $\nabla_{\hat{z}_t^{(k)}} \ell_{\text{guide}}$ is used to steer the reverse trajectory toward feasibility. In the above, sg treats the corrected point as a fixed target; by Proposition 1, this leaves the exact-projection gradient unchanged, while for approximate corrections it defines a fixed guidance target. Backpropagating these gradients through the shared denoiser and the K perturbations gives a single guidance direction for the current diffusion state,

$$\nabla_{y_t} \ell_{\text{guide}} = \frac{2}{K}\sum_{k=1}^{K} \left(\nabla_{y_t} \hat{z}_\theta(y_t^{(k)}, t, \xi)\right)^\top (\bar{z}_t - u_t), \tag{8}$$

where the derivative includes the dependence of each perturbation $y_t^{(k)}$ on the current state y_t. The constrained-guidance direction is then combined with the denoising direction for the reverse update:

$$y_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(y_t - \gamma_t \nabla_{y_t} \ell_{\text{guide}} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t^C\right) + \sigma_t \eta_t, \tag{9}$$

where $\gamma_t > 0$ controls the strength of the constraint guidance at timestep t. Therein, $\epsilon_t^C = \frac{y_t - \sqrt{\bar{\alpha}_t}\,(2u_t - 1)}{\sqrt{1-\bar{\alpha}_t}}$. is the noise estimate corresponding to the corrected relaxed decision u_t.

The resulting state y_{t-1} is fed back to the denoiser, allowing the constraint information introduced at timestep t to influence all subsequent reverse steps. The following result characterizes the exact projection and motivates its use for both reverse generation and constraint-aware training.

Proposition 1 (Projection geometry)**.** *Let $\mathcal{C}_\xi$ be nonempty, closed, and convex. Then the projection is unique, the squared distance $d^2_{\mathcal{C}_\xi}(v) = \|v - \Pi_{\mathcal{C}_\xi}(v)\|_2^2$ is continuously differentiable, and*

$$\nabla_v d^2_{\mathcal{C}_\xi}(v) = 2\big(v - \Pi_{\mathcal{C}_\xi}(v)\big).$$

Moreover, for every $z^\star \in \mathcal{Z}_\xi$,

$$\|\Pi_{\mathcal{C}_\xi}(v) - z^\star\|_2^2 \le \|v - z^\star\|_2^2 - \|v - \Pi_{\mathcal{C}_\xi}(v)\|_2^2.$$

The proof is provided in Appendix A.5.1. Thus, the exact projection is a minimum-distance correction whose application cannot increase the distance to any feasible reference, and yields the least disruptive feasible modification of the reverse transition. Let

$$R_t^v(y_t, \eta_t) = a_t y_t + b_t e(v) + \sigma_t \eta_t, \qquad b_t = \frac{\beta_t \sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t}, \tag{10}$$

denote the reverse transition anchored at a relaxed endpoint v.

Proposition 2 (Least-disruptive feasible transition)**.** *Let $\mathcal{C}_\xi$ be nonempty, closed, and convex, and let $u_t = \Pi_{\mathcal{C}_\xi}(v_t)$. For every $w \in \mathcal{C}_\xi$ and common noise realization η_t,*

$$\|R_t^{u_t} - R_t^w\|_2^2 \le \|R_t^{v_t} - R_t^w\|_2^2 - 4b_t^2 \|v_t - u_t\|_2^2.$$

The proof is provided in Appendix A.5.2. Proposition 2 gives a local guarantee for the least-disruptive feasible reverse transition; because corrected states are fed back to the denoiser, these corrections also influence subsequent predictions. This trajectory-level effect is reflected empirically in Section 6, where stepwise correction improves downstream objective quality over unconstrained generation and final-step correction.

5.2 Discrete Recovery and Continuous Completion

Stage (3): discrete recovery. Note that, after projection, the terminal estimate u_0 remains continuous, requiring a discretization operation. This step is deferred until this point because hard decisions made at an earlier timestep would discard the uncertainty needed by the remaining reverse process. A recovery map then converts u_0 into a binary decision:

$$\hat{z} = R_\xi(u_0), \qquad R_\xi : [0,1]^m \to \{0,1\}^m. \tag{11}$$

The default recovery uses componentwise thresholding, $T(u)_i = \mathbf{1}\{u_i \ge 1/2\}$. The default recovery uses $R_\xi = T$, but a problem-specific repair can augment this map when available, as in the case of our experiments. Membership of u_0 in C_ξ does not alone imply that $T(u_0)$ belongs to $\mathcal{Z}_\xi$. The following result quantifies what exact convex projection preserves through thresholding.

Corollary 3 (Threshold-recovery error)**.** *For every $u \in [0,1]^m$ and $z^\star \in \{0,1\}^m$,*

$$d_H(T(u), z^\star) \le 4\|u - z^\star\|_2^2. \tag{12}$$

If C_ξ is nonempty, closed and convex, $u = \Pi_{\mathcal{C}_\xi}(v)$ and $z^\star \in \mathcal{Z}_\xi$, then

$$d_H(T(u), z^\star) \le 4\left(\|v - z^\star\|_2^2 - \|v - u\|_2^2\right). \tag{13}$$

The proof is provided in Appendix A.5.3. Corollary 3 connects the projection geometry to the Hamming and exact-reconstruction metrics used in Section 6. It essentially shows that exact projection improves an upper bound on the error to every feasible reference decision. However, the bound certifies discrete feasibility only when it implies exact recovery of such a reference. More generally, a recovery map satisfying $R_\xi(C_\xi) \subseteq \mathcal{Z}_\xi$ certifies the combinatorial condition; Appendix A.6 gives a sufficient slack condition for thresholding to have this property at a particular terminal point.

Stage (4): continuous completion. For a recovered decision z, define its continuous completion set

$$\mathcal{X}_\xi(z) = \{x \in \mathcal{X} : g(x, z; \xi) \le 0\}. \tag{14}$$

which is used to recover the continuous decision $\hat{x}$ by solving Problem (2) with $z = \hat{z}$. The next proposition states when these steps produce a solution that satisfies both the discrete and continuous constraints of the original problem.

Proposition 4 (Feasibility by composition)**.** *Suppose that the terminal correction returns $u_0 \in \mathcal{C}_\xi$, $R_\xi(\mathcal{C}_\xi) \subseteq \mathcal{Z}_\xi$, and $\mathcal{X}_\xi(z)$ is nonempty for every $z \in \mathcal{Z}_\xi$. If the completion procedure returns $\hat{x} \in \mathcal{X}_\xi(\hat{z})$ whenever this set is nonempty, then $\hat{z} = R_\xi(u_0)$ and $\hat{x}$ form a feasible solution of Problem 1. A globally solved completion is optimal only conditional on $\hat{z}$.*

The proof is provided in Appendix A.5.4. Proposition 4 shows that a feasible final solution is guaranteed when the corrected prediction satisfies the relaxed constraints, recovery produces a feasible binary decision, and the continuous solution satisfies the remaining constraints. For the portfolio problem considered next, a nonempty support satisfying the quadratic combinatorial constraint admits a continuous completion. However, for AC transmission switching, the load-relevant connectivity condition used by CGD is necessary but not sufficient for the nonlinear AC equations; thus the evaluation fixes $z = \hat{z}$, solves Problem (2) with an off-the-shelf numerical optimizer, and reports discrete violations, downstream feasibility, objective quality, and end-to-end runtime separately.

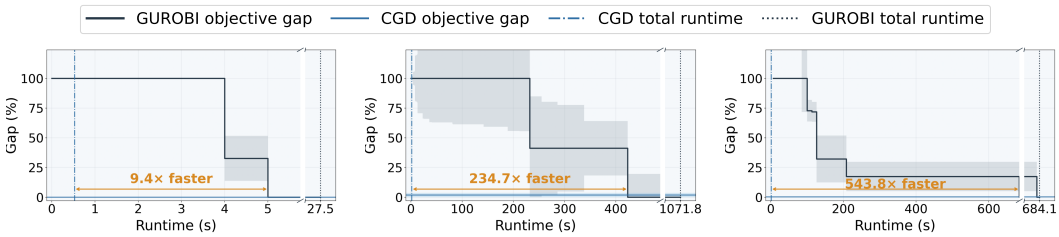


Figure 4: Runtime comparison for AC-OPF with branch switching on the IEEE 9-(left), 197-(middle), and 500-(right) bus systems. CGD attains objective gaps of 0.01%, 1.76%, and 0.20% in 0.53, 1.80, and 1.26 s, respectively. The orange arrows report matched-gap speedups of $9.4\times$, $234.7\times$, and $543.8\times$, computed relative to the time at which GUROBI first reaches the objective gap achieved by CGD. Shaded regions indicate one standard deviation across test instances.

6 Experiments

In this section we evaluate CGD on two hard mixed-integer optimization problems with fundamentally different combinatorial structures: AC optimal power flow with branch switching (Fisher et al., 2008) and portfolio optimization (Rubinstein, 2002) with discrete asset selection. The experiments address three questions: *(1) Discrete quality:* does CGD recover high-quality combinatorial decisions? *(2) Feasibility and downstream quality:* do these decisions induce feasible continuous problems with near-optimal objective values? *(3) Computational advantage:* does learning the discrete decisions reduce end-to-end solving time relative to numerically solving the original MIP?

Baselines and evaluation. We compare CGD against the following baseline methods: MLP and GNN, which directly predict the discrete decisions using feed-forward and graph-based architectures, DIFUSCO (Sun & Yang, 2023), a SOTA graph-based diffusion framework for combinatorial optimization, and LTO for MIP (Tang et al., 2025), which provides a learning-to-optimize baseline for mixed-integer problems by combining learned integer predictions with feasibility correction. Additionally, we perform an ablation study on CGD using two variants: **CGD without projection**, which removes the proposed feasibility guidance during denoising, and **CGD + final-step projection**, which applies a projection only after the final denoising step as a post-processing repair. Separately, to evaluate the computational cost, we compare CGD against Gurobi 13.0, the state-of-the-art commercial numerical MIP solver. Tables 1 and 2 report Hamming distance and exact reconstruction to quantify recovery of the reference discrete decision, while constraint violations, downstream infeasibility, and objective gap assess downstream performance. We refer to Appendix E for exact metric definitions. For the nonconvex AC-OPF benchmarks, reference solutions are the best feasible solutions returned by the solver, whereas for the portfolio MIQPs we use optimal solutions. For space limitations, additional implementation details and solver settings are reported in Appendix H.

6.1 AC Optimal Power Flow with Branch Switching

AC-OPF with branch switching jointly selects energized lines and generator dispatch while satisfying nonlinear power-flow constraints. We consider the 9-, 197-, and 500-bus systems (Babaeinejadsarookolaee et al., 2019), containing, respectively, $12, 286$ and 597 binary variables and $172, 3916$ and 8651 continuous variables. CGD guides relaxed line decisions using a load-relevant connectivity condition together with constraint information from the downstream AC-OPF problem. Both these corrections promote load-relevant connected network topologies and steer generation toward satisfying the nonlinear AC constraints, whose downstream feasibility is then evaluated separately. The problem formulation, projection operator, and datasets are given in Appendices F.1-F.2 and C-D.

Solution quality and feasibility. Table 1 addresses our first two questions. For *(1) discrete quality*, CGD achieves the lowest Hamming distance and highest exact reconstruction across all three networks, reaching 69.43%, 41.20%, and 33.24% exact reconstruction on the 9-, 197-,and 500-bus systems, respectively, consistently outperforming the baseline methods. For *(2) feasibility and downstream quality*, CGD achieves zero observed downstream infeasibility on all three networks and the lowest objective gaps of 0.01%, 1.76%, and 0.20%, respectively. The ablation shows that the correction mechanism is essential: removing it from generation increases both prediction er-

Case	Method	Discrete decision quality		Feasibility	Downstream quality
		Ham. (%) ↓	Exact (%) ↑	Infeasible (%) ↓	Gap (%) ↓
9-Bus	**CGD (ours)**	**1.71** (0.02)	**69.43** (4.36)	**0.00** (0.00)	**0.01** (0.02)
	CGD without projection	3.86 (0.14)	55.59 (6.86)	7.55 (1.21)	0.55 (0.05)
	CGD + final-step projection	2.71 (0.09)	57.51 (9.12)	**0.00** (0.00)	0.61 (0.10)
	MLP	11.72 (1.84)	29.66 (11.92)	**0.00** (0.00)	0.87 (0.17)
	GNN	13.71 (2.46)	9.76 (1.03)	5.67 (0.84)	0.28 (0.03)
	DIFUSCO	5.36 (2.19)	41.03 (7.59)	15.63 (9.02)	0.58 (0.32)
	LTO-for-MIP	4.52 (1.84)	47.36 (17.12)	**0.00** (0.00)	0.67 (0.25)
197-Bus	**CGD (ours)**	**0.17** (0.03)	**41.20** (4.96)	**0.00** (0.00)	**1.76** (0.23)
	CGD without projection	1.14 (0.27)	27.17 (10.81)	4.53 (1.03)	2.99 (0.51)
	CGD + final-step projection	2.42 (0.63)	34.79 (13.28)	**0.00** (0.00)	2.32 (0.31)
	MLP	24.55 (4.72)	0.00 (0.00)	100.00 (0.00)	n/a
	GNN	3.47 (0.52)	25.80 (8.28)	25.18 (10.42)	2.34 (0.51)
	DIFUSCO	6.36 (2.19)	21.03 (6.59)	7.41 (2.02)	3.64 (1.29)
	LTO-for-MIP	8.72 (3.61)	27.12 (6.19)	**0.00** (0.00)	2.67 (1.10)
500-Bus	**CGD (ours)**	**2.05** (0.31)	**33.24** (5.75)	**0.00** (0.00)	**0.20** (0.04)
	CGD without projection	4.79 (0.51)	24.70 (6.73)	3.26 (0.00)	0.25 (0.11)
	CGD + final-step projection	5.31 (0.87)	22.68 (4.53)	**0.00** (0.00)	0.41 (0.10)
	MLP	7.49 (0.89)	0.00 (0.00)	100.00 (0.00)	n/a
	GNN	5.24 (0.59)	15.61 (3.41)	**0.00** (0.00)	0.55 (0.17)
	DIFUSCO	10.36 (15.19)	11.03 (2.59)	21.18 (4.73)	0.78 (0.23)
	LTO-for-MIP	6.15 (2.64)	21.53 (4.70)	**0.00** (0.00)	0.74 (0.12)

Table 1: Discrete decision quality and downstream AC-OPF performance on 1000 test instances. Objective gaps are averaged over instances with a feasible downstream solve; n/a indicates that no such instance exists. Values are reported as mean, with 1 standard deviation in parentheses.

ror and downstream infeasibility, while applying it only at the final step restores feasibility, but does not recover the solution quality of CGD. These results show that incorporating constraint information throughout reverse diffusion improves the generated network topology and downstream performance. In addition to these experiments, Appendix F.3 reports results for the DC counterpart (a linear approximation of the AC model, widely used in industry). Interestingly, here, the linear downstream problem enables end-to-end differentiation. The results show the same qualitative pattern than the AC-OPF experiment, with zero observed infeasibility and lower objective gaps than other CGD variants.

Runtime. Next we investigate question *(3) computational cost*, comparing CGD with Gurobi. For a fair evaluation, Table 3 reports a matched-gap comparison, that is, in each benchmark, we measure the time required by Gurobi to first reach the same objective gap attained by CGD. We also stress that here, the majority of the time spent by Gurobi is on searching for a good primal solution, as opposed to simply trying to prove the dual bound. The proposed approach solves the 9-, 197-, and 500-bus instances in 0.53, 1.80, and 1.26 seconds, yielding increasing speedups with the problem size and complexity, respectively of $9.4\times$, $234.7\times$, and $543.8\times$. Figure 4 visualizes this comparison, illustrating that Gurobi spends practically the majority of the solving time searching for better primal solutions. For the linear DC-OPF cases in Appendix F.3, CGD is slower on the 9-bus system ($0.25\times$), comparable on the 197-bus system ($1.11\times$), and achieves a $16.71\times$ speedup on the 500-bus system. This suggests that the benefit of the proposed decomposition becomes more pronounced as the underlying optimization problem becomes harder.

6.2 Mixed-Integer Portfolio Optimization

Next, we also consider a mixed-integer extension of Markowitz portfolio optimization (Rubinstein, 2002), which formulation is reported in Appendix G.1. Binary variables z select the assets, while continuous variables x determine their portfolio weights. The selected support must satisfy the quadratic constraint $z^\top Pz \leq \rho$, after which the weights are recovered through a convex quadratic program. We evaluate CGD and the baseline methods on instances with $z = 50$ and $z = 150$ assets; the correction operator and data-generation algorithm are described in Appendices G.2–G.3.

Solution quality and feasibility. Table 2 shows that CGD performs strongly in both *(1)* discrete recovery and *(2)* downstream optimization. For $n = 50$, CGD achieves the lowest Hamming distance (8.21%), highest exact reconstruction (54.31%), and lowest objective gap (4.92%). For $n = 150$, the GNN baseline achieves slightly better reconstruction, yet incurs high constraint violations and a 13.53% objective gap. In contrast, CGD maintains zero observed violation and achieves the lowest

Number of assets	Method	Discrete quality		Constraint violation		Downstream quality
		Ham. (%) ↓	Exact (%) ↑	Mean ↓	Max. ↓	Gap (%) ↓
50	**CGD (ours)**	**8.21** (3.91)	**54.31** (5.46)	**0.00** (0.00)	**0.00** (0.00)	**4.92** (1.71)
	CGD without projection	15.45 (4.91)	20.32 (4.38)	0.51 (0.34)	16.72 (3.63)	8.43 (2.95)
	CGD + final-step projection	11.39 (2.48)	34.76 (10.32)	**0.00** (0.00)	**0.00** (0.00)	8.50 (3.12)
	MLP	27.15 (5.61)	17.10 (10.53)	0.03 (0.01)	15.81 (3.85)	19.32 (5.90)
	GNN	15.44 (2.47)	18.20 (11.51)	0.29 (0.24)	7.25 (1.74)	8.82 (3.15)
	DIFUSCO	20.15 (5.89)	5.90 (4.17)	0.04 (0.01)	6.55 (0.84)	5.79 (3.12)
	LTO-for-MIP	16.91 (4.63)	0.40 (0.12)	0.03 (0.01)	15.80 (0.78)	19.28 (6.76)
150	**CGD (ours)**	15.84 (4.12)	44.16 (13.65)	**0.00** (0.00)	**0.00** (0.00)	**6.56** (2.32)
	CGD without projection	20.29 (5.45)	29.71 (9.12)	0.85 (0.22)	8.14 (4.76)	11.25 (5.81)
	CGD + final-step projection	20.26 (3.12)	28.32 (3.98)	**0.00** (0.00)	**0.00** (0.00)	10.39 (3.15)
	MLP	26.91 (4.98)	13.09 (5.22)	0.26 (0.01)	17.12 (4.73)	26.30 (5.62)
	GNN	**12.18** (4.21)	**47.82** (12.84)	0.48 (0.12)	5.95 (1.73)	13.53 (3.86)
	DIFUSCO	18.48 (5.12)	35.31 (4.12)	0.36 (0.18)	5.82 (2.96)	18.48 (6.73)
	LTO-for-MIP	16.33 (7.15)	33.43 (3.70)	0.12 (0.10)	4.37 (2.22)	19.23 (5.13)

Table 2: Portfolio decision quality, quadratic-constraint violation, and downstream objective gap on 1200 test instances. Values are reported as mean, with 1 standard deviation in parentheses.

Stage	AC-OPF with branch switching			Portfolio optimization	
	9-Bus	197-Bus	500-Bus	50 assets	150 assets
GUROBI joint solve	27.54	1,071.81	684.12	0.093	1.213
GUROBI time to CGD gap	4.98	422.46	684.12	0.027	0.441
CGD sampling	0.13	0.16	0.11	0.014	0.015
Continuous completion	0.40	1.64	1.15	0.006	0.010
CGD end-to-end	**0.53**	**1.80**	**1.26**	**0.020**	**0.025**
Speedup	$\mathbf{9.4}\times$	$\mathbf{234.7}\times$	$\mathbf{543.8}\times$	$\mathbf{1.35}\times$	$\mathbf{17.64}\times$

Table 3: Average runtime per test instance, in seconds. The speedups are computed relative to the time required by GUROBI to reach the objective gap achieved by CGD.

objective gap, 6.56%. Thus, the accurate reconstruction of the reference support alone is not sufficient for strong downstream performance; constraint-aware generation instead can identify different feasible supports with much better objectives. As in AC-OPF, applying the correction throughout generation also outperforms terminal-only correction and unconstrained variants, highlighting the importance of integrating constraints during the reverse diffusion process.

Runtime. Table 3 shows that CGD solves the 50- and 150-asset instances in 0.020 and 0.025 seconds, corresponding to "matched-gap" speedups of $1.35\times$ and $17.64\times$ over Gurobi. Similarly to the AC-OPF experiment, the larger speedup on the 150-asset problem shows that CGD becomes more advantageous as the problem size increases. *These results are significant: CGD has the potential to enable more frequent and reliable optimization in many large-scale engineering systems.*

7 CONCLUSION

This paper presented Constrained Graph Diffusion (CGD), a constraint-aware diffusion framework that formulates discrete generation as a differentiable constrained reverse process over relaxed binary variables. CGD incorporates constraint information directly into the reverse trajectory, steering generation toward the feasible region before terminal recovery. This recovered binary decision is then fixed, and a numerical optimizer recovers the remaining continuous variables. Experiments on AC-OPF with branch switching and mixed-integer portfolio optimization show substantial gains in feasibility and downstream performance with respect to learning-based baselines, while achieving significant speedups over state-of-the-art MIP solvers while sacrificing only a fraction of the optimality gap. Across both applications, these gains increase with problem size, highlighting the importance of separating discrete generation from continuous completion. We believe that CGD can be applied to a wide range of combinatorial optimization problems beyond those studied in this paper, including scheduling, routing, and resource allocation problems. A possible avenue for future work is the development of foundation diffusion models to allow the resolution of different families of mixed-integer problems and multi-stage problems, where the discrete decisions are made sequentially and the continuous variables are recovered at each stage.

AI USE STATEMENT

We used generative AI tools to improve the clarity and presentation of the manuscript and refine the visual presentation of figures. All AI-assisted text and figures were reviewed and verified by the authors. Generative AI was not used to generate experimental data or determine the scientific conclusions of this work. The authors take full responsibility for the paper's content, including all claims and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

The proposed methodology is evaluated on standard optimization benchmarks and is not intended for harmful or discriminatory applications. We are not aware of any specific concerns related to privacy, security, fairness, or conflicts of interest arising from this research.

REPRODUCIBILITY STATEMENT

The paper and appendix provide the problem formulations, model architecture, training procedure, evaluation metrics, theoretical assumptions, and complete proofs. Additional implementation details, dataset-generation procedures, hyperparameters, and solver configurations are included in the appendix. An anonymized implementation will be provided with the supplementary materials, and the complete source code will be released publicly upon publication.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Ali Amiri. Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. *European Journal of Operational Research*, 171(2):567–576, 2006. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2004.09.018. URL https://www.sciencedirect.com/science/article/pii/S0377221704006435.

S. Babaeinejadsarookolaee et al. The power grid library for benchmarking AC optimal power flow algorithms, Aug. 2019. URL https://arxiv.org/abs/1908.02788.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021.

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

Daniel Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74(2):121–140, 1996. doi: 10.1007/BF02592208.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022. URL http://jmlr.org/papers/v23/21-0308.html.

Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017.

Shengyu Feng, Tarun Suresh, and Yiming Yang. Unsupervised diffusion solver for combinatorial optimization via combinatorial adjoint matching, 2026. URL https://arxiv.org/abs/2605.30920.

Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1504–1511, 2020. doi: 10.1609/aaai.v34i02.5509.

Emily B. Fisher, Richard P. O'Neill, and Michael C. Ferris. Optimal transmission switching. *IEEE Transactions on Power Systems*, 23(3):1346–1355, 2008. doi: 10.1109/TPWRS.2008.922256.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d14c2267d848abeb81fd590f371d39bd-Abstract.html.

Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d1e946f4e67db4b362ad23818a6fb78a-Abstract.html.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL https://www.gurobi.com.

Kory W. Hedman, Michael C. Ferris, Richard P. O'Neill, Emily Bartholomew Fisher, and Shmuel S. Oren. Co-optimization of generation unit commitment and transmission switching with n-1 reliability. *IEEE Transactions on Power Systems*, 25(2):1052–1063, 2010. doi: 10.1109/TPWRS.2009.2037232.

Seong-Hyun Hong, Hyun-Sung Kim, Zian Jang, Deunsol Yoon, Hyungseok Song, and Byung-Jun Lee. Unsupervised training of diffusion models for feasible solution generation in neural combinatorial optimization, 2024. URL https://arxiv.org/abs/2411.00003.

Rohit Kannan, Harsha Nagarajan, and Deepjyoti Deka. Strong partitioning and a machine learning approximation for accelerating the global optimization of nonconvex quadratically constrained quadratic programs. *INFORMS Journal on Computing*, 2025.

Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, pp. 6348–6358, 2017.

James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4475–4482, 2021. doi: 10.24963/ijcai.2021/610. URL https://doi.org/10.24963/ijcai.2021/610.

Haoyu Lei, Kaiwen Zhou, Yinchuan Li, Zhitang Chen, and Farzan Farnia. Boosting cross-problem generalization in diffusion-based neural combinatorial solver via inference time adaptation, 2025. URL https://arxiv.org/abs/2502.12188.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*. https://artowen.su.domains/mc/, 2013.

Seonho Park and Pascal Van Hentenryck. Self-supervised primal-dual learning for constrained optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4052–4060, 2023.

Max B. Paulus, Giulia Zarpellon, Andreas Krause, Laurent Charlin, and Chris Maddison. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17584–17600. PMLR, 2022. URL https://proceedings.mlr.press/v162/paulus22a.html.

Mark Rubinstein. Markowitz's" portfolio selection": A fifty-year retrospective. *The Journal of finance*, 57(3):1041–1045, 2002.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. End-to-end learning to warm-start for real-time quadratic optimization. In *Learning for Dynamics and Control Conference*, pp. 220–234. PMLR, 2023.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 43346–43367, 2024.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Bo Tang, Elias B. Khalil, and Ján Drgoňa. Learning to optimize for mixed-integer non-linear programming with feasibility guarantees, 2025. URL https://arxiv.org/abs/2410.11061.

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9367–9376. PMLR, 2020. URL https://proceedings.mlr.press/v119/tang20a.html.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pp. 1658–1665, 2019a.

Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. End to end learning and optimization on graphs. *Advances in Neural Information Processing Systems*, 32, 2019b.

A Additional CGD Details

This appendix complements Section 5 with parameterization details A.1– A.4, implementation considerations, training derivations, pseudocode (see Algorithm 1), and proofs A.5.

A.1 Relaxed Prediction Details

Under noise prediction, the denoiser first reconstructs the clean embedded state:

$$\widetilde{y}_{0,t} = \frac{y_t - \sqrt{1-\bar{\alpha}_t}\, s_\theta(y_t, t, \xi)}{\sqrt{\bar{\alpha}_t}}, \qquad \hat{y}_{0,t} = \tanh(\widetilde{y}_{0,t}). \tag{15}$$

The $\tanh$ map bounds the estimate in $[-1,1]^m$, after which the inverse embedding $e^{-1}(y) = (y+1)/2$ produces Equation 4. Each component has the interpretation

$$p_\theta(z_i = 1 \mid y_t, \xi) = \frac{\hat{y}_{0,t,i} + 1}{2}, \qquad \hat{y}_{0,t,i} = 2p_\theta(z_i = 1 \mid y_t, \xi) - 1. \tag{16}$$

The relaxed prediction can also be viewed as a finite-temperature smoothing:

$$S_\tau(a) = \frac{1}{2}\left[\tanh\left(\frac{a}{\tau}\right) + 1\right] = \frac{1}{1+\exp(-2a/\tau)}, \qquad \tau > 0. \tag{17}$$

Equation 4 uses $\tau = 1$. As $\tau \downarrow 0$, $S_\tau(a)$ approaches the hard decision $\mathbf{1}\{a \geq 0\}$ away from the threshold. For finite τ, it provides a differentiable path from the feasibility regularizer to the denoiser during training, while hard recovery remains an inference-time terminal operation.

A.2 Exact and Approximate Corrections

The exact projection in Equation 6 provides the geometric reference for CGD. For complex constraint sets, however, solving it at every reverse step may be computationally prohibitive. The approximate operator $\widetilde{\Pi}_{\mathcal{C}_\xi}$ may instead be realized through a fixed number of differentiable optimization iterations or through a surrogate gradient estimator.

In the AC-OPF with branch experiments, we use differentiable primal–dual iterations together with a straight-through estimator for evaluating binary branch status. The full application-specific construction is given in Appendix D.

The exact-projection guarantees in Propositions 1 and 2 are not automatically inherited by an approximate operator. In particular, an approximate correction need not return a point in $\mathcal{C}_\xi$ unless this is established separately.

The noisy state y_{t-1} also need not belong to $e(\mathcal{C}_\xi)$. The one-step result in Proposition 2 does not telescope through the nonlinear denoiser without additional contractivity assumptions. Repeated correction therefore produces a sequence of locally improved reverse transitions; it does not imply that every noisy state is feasible or that CGD samples exactly from the constrained data distribution.

A.3 Recovery and Completion Details

Exact projection improves the threshold-recovery bound in Corollary 3, but relaxed feasibility does not generally survive thresholding. Discrete feasibility can be certified when the recovery map satisfies

$$R_\xi(\mathcal{C}_\xi) \subseteq \mathcal{Z}_\xi.$$

Depending on the application, R_ξ may therefore use simple thresholding, structured rounding, or a problem-specific recovery rule.

For an approximate correction, the projected threshold bound in Equation 13 is not assumed to hold. The quality and feasibility of the recovered binary decision must instead be assessed explicitly.

Proposition 4 additionally requires complete recourse: every recovered discrete decision in $\mathcal{Z}_\xi$ must admit at least one continuous completion. This assumption is application-dependent. Satisfying the constraints exposed to the correction operator may be insufficient for feasibility of the complete continuous model, particularly for nonlinear physical systems. This motivates reporting discrete violations and downstream feasibility as separate metrics.

A.4 Constraint-Aware Training Details

The training algorithm of CGD uses the same relaxed representation introduced in Section 5, but does not use the Monte Carlo construction employed during generation. For each training pair $(\xi, z^\star)$, a timestep t and noise realization ϵ are sampled, yielding

$$y_t = \sqrt{\bar{\alpha}_t}\, e(z^\star) + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{18}$$

The shared denoiser maps y_t to the relaxed prediction

$$\hat{z}_t = \hat{z}_\theta(y_t, t, \xi) \in [0,1]^m, \tag{19}$$

using the smooth parameterization described in Appendix A.1.

The relaxed prediction is then corrected to define a feasibility target,

$$p_t = \mathrm{sg}\left[\widetilde{\Pi}_{\mathcal{C}_\xi}(\hat{z}_t)\right], \qquad \ell_{\mathrm{feas}}(t, \xi, z^\star) = \|\hat{z}_t - p_t\|_2^2, \tag{20}$$

where sg denotes stop-gradient. Thus, gradients propagate through the relaxed prediction and the denoising network, while the corrected point is treated as a fixed target. The resulting gradient is

$$\nabla_\theta \ell_{\mathrm{feas}} = 2\left(\nabla_\theta \hat{z}_t\right)^\top \left(\hat{z}_t - \widetilde{\Pi}_{\mathcal{C}_\xi}(\hat{z}_t)\right). \tag{21}$$

When $\widetilde{\Pi}_{\mathcal{C}_\xi} = \Pi_{\mathcal{C}_\xi}$ is the exact Euclidean projection onto a nonempty, closed, and convex set, Proposition 1 gives

$$\nabla_{\hat{z}_t} d^2_{\mathcal{C}_\xi}(\hat{z}_t) = 2\left(\hat{z}_t - \Pi_{\mathcal{C}_\xi}(\hat{z}_t)\right), \tag{22}$$

so the stop-gradient expression in Equation (21) coincides with the gradient of the squared distance to $\mathcal{C}_\xi$. The projection Jacobian is not required, since the optimizing projection point is already accounted for by the envelope theorem. For an approximate correction, the same expression defines a feasibility-target surrogate rather than the exact gradient of a distance function.

The denoiser is trained with the combined objective

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{DSM}}(\theta) + \lambda_{\mathrm{feas}} \mathbb{E}_{(\xi, z^\star), t}\left[\ell_{\mathrm{feas}}(t, \xi, z^\star)\right], \tag{23}$$

where $\lambda_{\mathrm{feas}} > 0$ balances denoising fidelity and relaxed feasibility. The denoising term preserves fidelity to the reference decision distribution, while the feasibility term encourages relaxed predictions toward the supplied combinatorial feasible set. This training-time correction is distinct from the Monte Carlo guidance used during generation in Section 5.1. During training, the projection defines a fixed feasibility target for learning the denoiser; during generation, K perturbed predictions are aggregated and their projection residual is backpropagated with respect to the current diffusion state to steer the reverse trajectory.

A.5 Proofs

A.5.1 Proof of Proposition 1

Let $u = \Pi_{\mathcal{C}_\xi}(v)$. Because $\mathcal{C}_\xi$ is nonempty, closed, and convex, u exists and is unique. The squared distance to a closed convex set is continuously differentiable, with

$$\nabla_v d^2_{\mathcal{C}_\xi}(v) = 2(v - u).$$

The variational characterization of projection gives

$$\langle v - u, z - u \rangle \le 0 \qquad \text{for every } z \in \mathcal{C}_\xi.$$

For $z^\star \in \mathcal{Z}_\xi \subseteq \mathcal{C}_\xi$,

$$\|v - z^\star\|_2^2 = \|v - u\|_2^2 + \|u - z^\star\|_2^2 + 2\langle v - u, u - z^\star \rangle.$$

Since the final inner product is nonnegative,

$$\|u - z^\star\|_2^2 \le \|v - z^\star\|_2^2 - \|v - u\|_2^2,$$

which proves the second inequality in Proposition 1. □

Algorithm 1 Four-stage generation with CGD

Require: Instance ξ, denoiser $\hat{z}_\theta$, recovery map R_ξ

Sample $y_T \sim \mathcal{N}(0, I)$
for $t = T, \dots, 1$ **do**
  Construct K perturbed states $\{y_t^{(k)}\}_{k=1}^K$ from the current state y_t
  $\hat{z}_t^{(k)} \leftarrow \hat{z}_\theta(y_t^{(k)}, t, \xi), \quad k = 1, \dots, K$
  $\bar{z}_t \leftarrow \frac{1}{K} \sum_{k=1}^K \hat{z}_t^{(k)}$
  $u_t \leftarrow \widetilde{\Pi}_{\mathcal{C}_\xi}(\bar{z}_t)$
  $\ell_{\text{guide}} \leftarrow \|\bar{z}_t - \text{sg}[u_t]\|_2^2$
  Compute $\nabla_{y_t} \ell_{\text{guide}}$ using Equation 8
  $\epsilon_t^C \leftarrow \dfrac{y_t - \sqrt{\bar{\alpha}_t}(2u_t - 1)}{\sqrt{1 - \bar{\alpha}_t}}$
  Sample $\eta_t \sim \mathcal{N}(0, I)$
  Update y_{t-1} using Equation 9
end for
$u_0 \leftarrow u_1$ ▷ Terminal corrected prediction
$\hat{z} \leftarrow R_\xi(u_0)$ ▷ Discrete recovery
Fix $z = \hat{z}$ and use a numerical optimizer to solve

$$\min_{x \in \mathcal{X}} f(x, \hat{z}) \quad \text{subject to} \quad g(x, \hat{z}; \xi) \le 0.$$

return $(\hat{x}, \hat{z})$

A.5.2 Proof of Proposition 2

From Equation 10 and $e(v) = 2v - 1$,

$$R_t^v(y_t, \eta_t) - R_t^w(y_t, \eta_t) = 2b_t(v - w).$$

Applying the projection inequality from Proposition 1 with $v = v_t$, $u = u_t$, and $w \in \mathcal{C}_\xi$ yields

$$4b_t^2 \|u_t - w\|_2^2 \le 4b_t^2 \|v_t - w\|_2^2 - 4b_t^2 \|v_t - u_t\|_2^2,$$

which proves the inequality stated in Proposition 2.

For fixed y_t, $K_t^{v_t}$ and K_t^w are Gaussian distributions with equal covariance and means differing by $2b_t(v_t - w)$. Therefore,

$$W_2^2(K_t^{v_t}, K_t^w) = 4b_t^2 \|v_t - w\|_2^2. \tag{24}$$

Since $u_t = \Pi_{\mathcal{C}_\xi}(v_t)$, minimizing the right-hand side over $w \in \mathcal{C}_\xi$ gives

$$u_t = \arg\min_{w \in \mathcal{C}_\xi} W_2^2(K_t^{v_t}, K_t^w)\,. \tag{25}$$

Thus, among reverse transitions with the same covariance and a relaxed feasible endpoint, the projected transition is the closest to the original transition in 2-Wasserstein distance. □

A.5.3 Proof of Corollary 3

Whenever $T(u)_i \neq z_i^\star$, $|u_i - z_i^\star| \ge 1/2$. Hence,

$$\mathbf{1}\{T(u)_i \neq z_i^\star\} \le 4|u_i - z_i^\star|^2.$$

Summing over i proves Equation 12. If $u = \Pi_{\mathcal{C}_\xi}(v)$, combining this inequality with with the projection inequality from Proposition 1 proves Equation 13. □

A.5.4 Proof of Proposition 4

Because $u_0 \in \mathcal{C}_\xi$ and $R_\xi(\mathcal{C}_\xi) \subseteq \mathcal{Z}_\xi$, the recovered decision $\hat{z} = R_\xi(u_0)$ belongs to $\mathcal{Z}_\xi$. Complete recourse gives $\mathcal{X}_\xi(\hat{z}) \neq \emptyset$, and the completion procedure returns $\hat{x} \in \mathcal{X}_\xi(\hat{z})$. Thus, $(\hat{x}, \hat{z})$ satisfies the discrete and continuous constraints of Problem 1.

A global solution of the completion problem minimizes the objective only over continuous decisions paired with $\hat{z}$; it does not compare $\hat{z}$ with other discrete assignments. □

A.6 A Slack Certificate for Threshold Recovery

Corollary 3 controls discrete error relative to a feasible reference. The next result instead gives a directly checkable sufficient condition for thresholding a relaxed feasible point without violating its discrete constraints.

Proposition 5 (Feasibility after threshold recovery). *Suppose*

$$C_\xi = \left\{ u \in [0,1]^m : \widetilde{h}_j(u;\xi) \le 0, j = 1, \dots, q \right\},$$

where $\widetilde{h}_j$ agrees with h_j on binary points and is L_j-Lipschitz under a fixed norm. For $u \in C_\xi$, let $z = T(u)$ and define

$$s_j(u) := -\widetilde{h}_j(u;\xi), \qquad \delta(u) := \|T(u) - u\|.$$

Then, with $[r]_+ := \max\{r, 0\}$,

$$\left[h_j(z;\xi)\right]_+ \le \left[L_j \delta(u) - s_j(u)\right]_+. \tag{26}$$

Consequently, $T(u) \in \mathcal{Z}_\xi$ whenever $L_j \delta(u) \le s_j(u)$ for every j.

Proof. Since z is binary and $\widetilde{h}_j$ agrees with h_j on binary points, Lipschitz continuity gives

$$h_j(z;\xi) = \widetilde{h}_j(z;\xi) \le \widetilde{h}_j(u;\xi) + L_j \|z - u\| = -s_j(u) + L_j \delta(u).$$

Taking positive parts proves Equation (26).

Proposition 5 identifies the two quantities governing recovery: the relaxed constraint slack $s_j(u)$ and the integrality defect $\delta(u)$. Relaxed feasibility alone requires only $s_j(u) \ge 0$; thresholding is certified only when this slack is large enough to absorb the rounding displacement. The condition is sufficient, not necessary, and it applies only when the application admits the stated Lipschitz extension.

A.7 Portfolio Recovery and Continuous Completion

Proposition 6 (Quadratic feasibility after portfolio recovery). *Let $P = P^\top \succeq 0$, let $u \in [0,1]^m$ satisfy $u^\top P u \le \rho$, and let $z = T(u)$. Define*

$$d := z - u, \qquad r := \|d\|_2, \qquad s := \rho - u^\top P u, \qquad \lambda := \lambda_{\max}(P).$$

Then

$$\left[z^\top P z - \rho\right]_+ \le \left[-s + 2\|Pu\|_2 r + \lambda r^2\right]_+ \le \left[-s + 2\sqrt{\lambda\, u^\top P u}\, r + \lambda r^2\right]_+. \tag{27}$$

In particular, thresholding preserves the quadratic constraint whenever $2\|Pu\|_2 r + \lambda r^2 \le s$.

Proposition (6) characterizes preservation of the quadratic constraint under componentwise threshold recovery. In the experiments, the recovery map additionally enforces the cardinality and nonempty-support conditions, $1 \le \mathbf{1}^\top z \le k$. These conditions are therefore enforced by construction, while quadratic feasibility is evaluated separately after recovery.

Proof. Expanding the recovered decision $z = u + d$ gives

$$z^\top P z - \rho = -s + 2u^\top P d + d^\top P d.$$

Cauchy-Schwarz and positive semidefiniteness give

$$2u^\top P d \le 2\|Pu\|_2 \|d\|_2, \qquad d^\top P d \le \lambda \|d\|_2^2.$$

Finally, diagonalizing P gives $\|Pu\|_2^2 \le \lambda u^\top P u$, proving Equation (27).

Corollary 7 (Portfolio completion). *For the fixed-support constraints in Problem (49), a continuous completion exists if and only if $z \ne \mathbf{0}$. Consequently, a recovered support induces a feasible portfolio problem whenever $z^\top P z \le \rho$ and $z \ne \mathbf{0}$.*

Proof. If $z_i = 1$ for some asset i, then $x = \mathbf{e}_i$ satisfies $\mathbf{1}^\top x = 1$ and $0 \le x_j \le z_j$ for every j. If $z = \mathbf{0}$, the linking constraints force $x = \mathbf{0}$, contradicting $\mathbf{1}^\top x = 1$.

Proposition 6 explains why exact projection onto the relaxed quadratic set does not alone certify the thresholded support: a projection near the boundary can have insufficient slack s to absorb its integrality defect r. Corollary 7 adds the distinct nonempty-support condition required by the downstream budget equality. These certificates are a posteriori conditions, not additional operations performed by CGD.

B Related Works

Our work lies at the intersection of learning-for-optimization and diffusion models for combinatorial optimization. Here we provide an overview of the most closely related literature in these two areas.

Learning for Mixed-Integer Optimization. When many optimization instances share a formulation and differ only in their parameters, learned proxies can amortize computation by mapping instance data to solutions or useful algorithmic decisions (Donti et al., 2017; Kotary et al., 2021; Park & Van Hentenryck, 2023; Chen et al., 2022; Bengio et al., 2021). Although this paradigm is particularly natural for continuous problems with locally regular solution maps, integrality makes the corresponding map discontinuous and potentially set-valued: small parameter changes can switch the optimal discrete assignment, and multiple structurally different assignments can be near-optimal.

One major line of work therefore learns selected components of a conventional MIP solver. Early learning-to-branch methods imitate strong branching using features of candidate variables (Khalil et al., 2016); later work represents a MIP as a bipartite variable–constraint graph and uses graph neural networks to transfer branching policies across instances (Gasse et al., 2019). Hybrid architectures reduce the inference overhead of graph-based branching (Gupta et al., 2020), while Neural Diving and Neural Branching learn primal partial assignments and branching decisions within a solver pipeline (Nair et al., 2020). Related methods learn cut selection (Tang et al., 2020; Paulus et al., 2022) or graph-based construction heuristics (Khalil et al., 2017). For solver-embedded methods, the learned model proposes valid algorithmic actions while the surrounding solver remains responsible for feasibility, bounds, and certification when run to completion. Their scope is nevertheless deliberately local: training often requires expensive expert decisions such as strong-branching or lookahead labels, policy inference adds overhead at many search nodes, and the resulting method still performs online branch-and-bound or solves residual MIPs. Consequently, learned solver components accelerate combinatorial search but do not amortize it away.

A second line learns more direct mappings from parameters to MIP solutions or optimization strategies. Examples include predicting a reusable optimal strategy and reconstructing the associated solution (Bertsimas & Stellato, 2022), training construction or decision-focused models through continuous surrogates (Wilder et al., 2019a;b), differentiating through a cutting-plane representation of a MIP (Ferber et al., 2020), and designing feasibility-aware proxies for mixed-integer nonlinear programs (Tang et al., 2025). These approaches can amortize a larger fraction of the solve, but typically restrict the strategy class, return a single decision, or rely on relaxations, rounding, repair, or a downstream solver. In particular, a deterministic predictor does not represent alternative near-optimal discrete modes, while interpolation through a continuous relaxation provides no inherent guarantee of integrality or combinatorial feasibility.

CGD occupies a different point in this landscape. It neither learns a branching or cutting policy nor differentiates through the complete mixed-integer solve. Instead, it learns a conditional distribution over the discrete block, formulates the generation of the discrete block as a differentiable constrained reverse process, fixes the recovered binary assignment, and subsequently solves the remaining continuous problem with a numerical optimizer. This removes online branch-and-bound search over the discrete variables and can represent multiple plausible assignments, while retaining a numerical optimizer for the continuous variables. It does not remove continuous nonconvexity or, by itself, provide a global-optimality certificate.

Diffusion Models for Combinatorial Optimization. Recent work has demonstrated that diffusion models provide an effective generative framework for combinatorial optimization by learning distributions over discrete solution spaces. A seminal contribution in this direction is DIFUSCO (Sun & Yang, 2023), which introduced graph-based diffusion models for combinatorial optimization prob-

lems and demonstrated that iterative denoising can recover high-quality solutions for a variety of graph-structured tasks. Subsequent work has extended this paradigm to unsupervised settings, data-free training, and inference-time adaptation, further demonstrating the flexibility of diffusion-based optimization (Sanokowski et al., 2024; Hong et al., 2024; Feng et al., 2026; Lei et al., 2025).

Despite their success, existing diffusion-based optimization methods typically handle constraints through soft objective formulations or post-processing procedures, rather than explicitly enforcing feasibility throughout the generative process. In mixed-integer optimization, however, the discrete decisions directly affect the feasibility of the continuous variables. This motivates our approach, which integrates the problem combinatorial constraints throughout the generation process, enabling a decomposition in which the discrete decisions are learned by a diffusion model while the continuous variables are subsequently recovered by solving the resulting continuous optimization problem.

C AC-OPF WITH BRANCH SWITCHING - PROJECTION ONTO THE LOAD-RELEVANT CONNECTIVITY SET AT TRAINING TIME

For this problem, the training-time correction $\Pi_{\mathcal{C}_\xi}$ projects the relaxed network topology prediction $\hat{z} \in [0,1]^{|\mathcal{E}|}$ onto a continuous relaxation of the load-relevant connectivity set. The projection is constructed so that its binary points recover the original connectivity requirement: every connected component containing nonzero demand must contain at least one generator, whereas load-free islands are permitted.

Let

$$d_i = \begin{cases} 1, & \Re(S_i^d) > \epsilon, \\ 0, & \text{otherwise}, \end{cases}$$

where $\epsilon > 0$ is a small threshold used to identify load buses, and let

$$D = \sum_{i \in \mathcal{N}} d_i.$$

The projection introduces relaxed branch-status variables $z_e \in [0,1]$, directed flow variables $f_{ij,e} \geq 0$, and generator supply variables $s_i \geq 0$. Given a relaxed prediction $\hat{z}$, the projection is

$$\Pi_{\mathcal{C}_\xi}(\hat{z}) \in \arg\min_{z,f,s} \quad \|z - \hat{z}\|_2^2 \tag{28}$$

$$\text{s.t.} \quad f_{ij,e} \leq D z_e, \qquad f_{ji,e} \leq D z_e, \qquad \forall e = (i,j) \in \mathcal{E}, \tag{29}$$

$$s_i = 0, \qquad \forall i \notin \mathcal{G}, \tag{30}$$

$$\sum_{e \in \delta^+(i)} f_e - \sum_{e \in \delta^-(i)} f_e = s_i - d_i, \qquad \forall i \in \mathcal{N}, \tag{31}$$

$$\sum_{i \in \mathcal{N}} s_i = D, \tag{32}$$

$$0 \leq z_e \leq 1, \qquad f_e \geq 0. \tag{33}$$

The objective performs the minimum Euclidean correction of the relaxed prediction $\hat{z}$, consistently with the projection operator introduced in Equation (6). Constraints (29) couple the continuous branch-status variables to the auxiliary flow, while Constraint (30) restricts flow sources to generator buses. Constraints (31)–(32) require one unit of flow to be delivered to every load bus. Consequently, when z is binary, every connected component containing load must contain at least one generator, whereas components containing no load may remain disconnected.

The relaxation allows the same connectivity construction to act directly on the continuous predictions produced by the denoiser. During training, this projection is applied to the relaxed prediction $\hat{z}_t$ in Equation (19), providing the corrected target used by the feasibility regularizer in Equation (20). Since the feasible set defined above is convex, the projection is a convex quadratic program and corresponds directly to the ideal Euclidean projection analyzed in Proposition 1.

Binary recovery for evaluation. For evaluation purposes only, we use the same connectivity formulation with binary branch-status variables, i.e., $z_e \in \{0, 1\}$. Given a binary input $\hat{z}$, the Euclidean objective $\|z - \hat{z}\|_2^2$ is equivalent to the Hamming distance. The resulting MILP therefore computes the closest binary topology satisfying the load-relevant connectivity condition.

D AC-OPF WITH BRANCH SWITCHING - PROJECTION OPERATOR AT GENERATION TIME

For this experiment, CGD uses a differentiable correction based on an augmented-Lagrangian formulation of the AC power-flow constraints in Model 1.

The sampling-time correction acts jointly on the relaxed topology $\hat{z}_t$ and on the continuous AC variables

$$x = \{S_i^r, V_i, S_{ij}\}, \tag{34}$$

consistently with Model 1, where

$$S_i^r = P_i^r + \mathrm{j}Q_i^r, \qquad V_i = V_i^m e^{\mathrm{j}V_i^a}. \tag{35}$$

During the correction, the generator injections S_i^r and voltages V_i are updated as continuous variables, while the branch flows S_{ij} are computed from the AC branch-flow equations using the current topology and voltages. These continuous variables are used only to guide the topology correction. After sampling, the final binary topology $\hat{z}$ is fixed and the downstream AC-OPF is solved independently with the numerical optimizer.

Although $\hat{z}_t \in [0, 1]^{|\mathcal{L}|}$ is continuous, the AC equations are evaluated using binary branch statuses. We therefore use the hard decision

$$T(\hat{z}_t)_{ij} = \mathbf{1}\{\hat{z}_{t,ij} \geq 1/2\}, \tag{36}$$

together with a straight-through estimator: the forward evaluation uses $T(\hat{z}_t)$, while gradients with respect to the topology are propagated through the relaxed prediction $\hat{z}_t$.

The differentiable AC formulation evaluates the constraints of Model 1. We denote by $g_{\mathrm{eq}}(x, \hat{z}_t; \xi)$ the residuals associated with active- and reactive-power balance and the reference-bus angle condition, and by $g_{\mathrm{ineq}}(x, \hat{z}_t; \xi) \geq 0$ the positive violations associated with connectivity, branch thermal and angle limits, generator active- and reactive-power limits, and bus-voltage bounds. Hence, satisfaction of the constraints included in the correction corresponds to

$$g_{\mathrm{eq}}(x, \hat{z}_t; \xi) = 0, \qquad g_{\mathrm{ineq}}(x, \hat{z}_t; \xi) = 0. \tag{37}$$

To prevent the correction from moving arbitrarily far from the prediction produced by the denoiser, the augmented Lagrangian additionally contains a proximity term between the corrected topology u_t and the original relaxed prediction $\hat{z}_t$. The sampling-time correction approximately minimizes

$$\begin{aligned}\mathcal{L}_\rho = \lambda_{\mathrm{cost}} f(x, u_t) + (\lambda^{\mathrm{eq}})^\top g_{\mathrm{eq}}(x, u_t; \xi) + (\lambda^{\mathrm{ineq}})^\top g_{\mathrm{ineq}}(x, u_t; \xi) \\ + \frac{\rho}{2}\left(\|g_{\mathrm{eq}}(x, u_t; \xi)\|_2^2 + \|g_{\mathrm{ineq}}(x, u_t; \xi)\|_2^2\right) + \lambda_{\mathrm{anchor}} D_{\mathrm{KL}}\left(u_t \,\|\, \hat{z}_t\right),\end{aligned} \tag{38}$$

where λ^{eq} and $\lambda^{\mathrm{ineq}} \geq 0$ are the equality and inequality multipliers, respectively, $\rho > 0$ is the augmented-Lagrangian penalty parameter, $\lambda_{\mathrm{anchor}} > 0$ controls proximity to the original diffusion prediction, and $\lambda_{\mathrm{cost}} \geq 0$ controls the contribution of generation cost. Setting $\lambda_{\mathrm{cost}} = 0$ yields a feasibility-only correction.

At each reverse-diffusion steps, a small number of primal-dual iterations is performed. Starting from $u_t = \hat{z}_t$, the corrected topology and the continuous variables are updated by gradient descent on Equation (38),

$$u_t \leftarrow u_t - \eta_z \nabla_{u_t} \mathcal{L}_\rho, \tag{39}$$

$$x \leftarrow x - \eta_x \nabla_x \mathcal{L}_\rho, \tag{40}$$

while the dual variables are updated by residual ascent,

$$\lambda^{\mathrm{eq}} \leftarrow \lambda^{\mathrm{eq}} + \eta_\lambda g_{\mathrm{eq}}(x, u_t; \xi), \tag{41}$$

$$\lambda^{\mathrm{ineq}} \leftarrow \left[\lambda^{\mathrm{ineq}} + \eta_\lambda g_{\mathrm{ineq}}(x, u_t; \xi)\right]_+. \tag{42}$$

After the prescribed number of inner iterations, the refined topology defines the sampling-time correction

$$u_t = \Pi_{\mathcal{C}_\xi}(\bar{z}_t), \tag{43}$$

which is reinserted into the reverse diffusion process as described in Section 5.1.

This procedure is an approximate realization of the projection operator introduced in Section 5.1. Unlike the ideal Euclidean projection analyzed in Propositions 1–2, the AC constraint set is nonlinear and the inner optimization is performed for only a finite number of iterations. The augmented-Lagrangian terms steer the prediction toward AC-compatible topologies, while the proximity term retains the minimum-change principle underlying the ideal projection.

E Experimental Details

E.1 Baselines

We compare CGD against four learning-based baselines and two ablation variants. **CGD without projection** uses the same diffusion architecture but removes constraint correction during training and generation. **CGD + final-step projection** applies the correction only after the last denoising step and represents terminal post-processing. **MLP** predicts a relaxed decision in $[0,1]^m$ using a feed-forward network trained with mean-squared error and thresholds it at 0.5. **GNN** directly predicts the decisions using message passing over the problem graph. **DIFUSCO** Sun & Yang (2023) is a graph diffusion model without application-specific constraint correction during sampling. **LTO-for-MIP** Tang et al. (2025) predicts integer decisions using differentiable correction layers followed by feasibility projection.

The MLP and GNN produce one deterministic prediction per instance. For diffusion-based methods, we use the same Monte Carlo sample budget K during generation. In CGD, these samples correspond to the auxiliary perturbations used at each reverse step to construct the Monte Carlo estimate described in Section 5.1.

E.2 Evaluation Metrics

Let $(x^\star, z^\star)$ denote the reference solution and $\hat{z}$ the decision returned by a learned method. For N test instances with m binary variables, normalized Hamming distance is

$$\text{Ham } (\%) = \frac{100}{N}\sum_{j=1}^{N}\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}\left[\hat{z}_i^{(j)} \neq z_i^{\star(j)}\right], \tag{44}$$

and exact reconstruction is

$$\text{Recon } (\%) = \frac{100}{N}\sum_{j=1}^{N}\mathbf{1}\left[\hat{z}^{(j)} = z^{\star(j)}\right]. \tag{45}$$

We separately report discrete constraint violations and the percentage of predictions whose induced continuous problem is infeasible. For feasible instances, the objective gap is

$$\text{Objective Gap } (\%) = \frac{100}{N_{\text{feas}}}\sum_{j\in\mathcal{I}_{\text{feas}}}\frac{\left|\hat{f}^{(j)} - f^{\star(j)}\right|}{\left|f^{\star(j)}\right|}, \tag{46}$$

where $\hat{f}^{(j)} = f(\hat{x}^{(j)}, \hat{z}^{(j)})$, $f^{\star(j)} = f(x^{\star(j)}, z^{\star(j)})$, and $\mathcal{I}_{\text{feas}}$ contains the instances with a feasible continuous completion. The objective gap is therefore conditional on downstream feasibility.

Throughout the paper, *reference solution* denotes the discrete and continuous decisions returned by the corresponding optimization solver. For the nonconvex AC-OPF instances, these solutions are the best feasible solutions returned under the prescribed time limit and are not assumed to be globally certified optima.

Table 4: Scale of the experimental benchmarks.

Benchmark	Binary	Continuous	Instances	Test
IEEE 9-bus	12	172	10,000	1,000
IEEE 197-bus	286	3,916	10,000	1,000
IEEE 500-bus	597	8,651	10,000	1,000
Portfolio, $n = 50$	50	50	12,000	1,200
Portfolio, $n = 150$	150	150	12,000	1,200

F AC-OPF WITH BRANCH SWITCHING

F.1 PROBLEM FORMULATION

Let $\mathcal{N}$, $\mathcal{L}$, and $\mathcal{G}$ denote the buses, transmission lines (or branches), and generators. The binary variable z_{ij} indicates whether line (i, j) is energized. The continuous variables comprise generator injections S_i^r, complex bus voltages V_i, and branch flows S_{ij}. The problem is formulated as follows

Model 1 AC-OPF with branch switching.

$$\textbf{variables:} \quad x = \{S_i^r, V_i, S_{ij}\}, \qquad z_{ij} \in \{0, 1\},$$

$$\textbf{minimize:} \quad \sum_{i \in \mathcal{G}} c_{2i} \big(\Re(S_i^r)\big)^2 + c_{1i} \Re(S_i^r) + c_{0i} \tag{47a}$$

$$\textbf{subject to:} \quad v_i^l \le |V_i| \le v_i^u, \qquad i \in \mathcal{N}, \tag{47b}$$

$$-\theta_{ij}^{\Delta} z_{ij} \le \angle(V_i V_j^*) \le \theta_{ij}^{\Delta} z_{ij}, \qquad (i,j) \in \mathcal{L}, \tag{47c}$$

$$S_i^{rl} \le S_i^r \le S_i^{ru}, \qquad i \in \mathcal{N}, \tag{47d}$$

$$|S_{ij}| \le s_{ij}^u z_{ij}, \qquad (i,j) \in \mathcal{L}, \tag{47e}$$

$$S_i^r - S_i^d = \sum_{(i,j) \in \mathcal{L}} S_{ij}, \qquad i \in \mathcal{N}, \tag{47f}$$

$$S_{ij} = z_{ij} \left(Y_{ij}^* |V_i|^2 - Y_{ij}^* V_i V_j^*\right), \qquad (i,j) \in \mathcal{L}, \tag{47g}$$

$$\theta_{\text{ref}} = 0. \tag{47h}$$

F.2 DATA GENERATION

For each network, we perturb the parameters $\xi = \{S^d, Y^*, S^{ru}\}$ and solve the resulting nonconvex mixed-integer AC-OPF with GUROBI Gurobi Optimization, LLC (2026). For the IEEE 9-bus benchmark, the active and reactive demands at each load bus share a uniformly sampled scaling factor within $\pm 50\%$ of their nominal values, preserving their correlation.

For the IEEE 197- and 500-bus systems, demands are sampled between 80% and 120% of nominal values, transmission line parameters between 80% and 125%, and generator upper limits between 90% and 115%. Line resistance parameters are perturbed between 75% and 125%, and generator costs between 80% and 120% of their nominal values.

Each network contains $10{,}000$ instances, split into 80% training, 10% validation, and 10% test data. A separate model is trained for each network. Because the underlying formulation is a nonconvex MINLP, reference solutions are the best feasible solutions returned under the solver settings and time limit reported in Appendix H. When generating the data, we obtain the following fraction of open lines: 13.59%, 2.49%, and 4.70% for AC-OTS on the 9-, 197-, and 500-bus systems, and 11.20%, 3.94%, and 7.65% for the corresponding DC-OTS benchmarks.

F.3 DC-OPF WITH BRANCH SWITCHING

For completeness, we additionally evaluate CGD on DC-OPF with branch switching, the linearized counterpart of the AC-OPF problem considered in Section 6. The DC formulation retains the same combinatorial line-switching decisions, with binary variables indicating whether transmission lines

are active, but replaces the nonlinear AC power-flow equations with the DC power-flow approximation. Consequently, once the topology is fixed, the remaining continuous completion problem is convex and substantially easier to solve than the corresponding AC-OPF. This provides a complementary setting in which the combinatorial structure is preserved while the nonconvexity of the continuous power-flow model is removed. We use this experiment to assess how the benefit of learning the discrete topology changes when the downstream numerical optimization problem is comparatively simpler. The resulting prediction quality, downstream DC-OPF performance, and runtimes are reported in Tables 5 and 6.

During constrained generation, the DC power-flow constraints are incorporated through a differentiable convex optimization layer (Agrawal et al., 2019), allowing gradients from the DC correction to propagate to the relaxed topology decisions. As in the AC experiments, the recovered topology is fixed and the remaining continuous DC-OPF is solved numerically.

Model 2 DC-OPF with branch switching.

$$
\begin{aligned}
&\textbf{variables:} && x = \{P_i^r, \theta_i, P_{ij}\}, \qquad z_{ij} \in \{0,1\}, && && \\
&\textbf{minimize:} && \sum_{i \in \mathcal{G}} c_{2i}(P_i^r)^2 + c_{1i}P_i^r + c_{0i} && && \text{(48a)} \\
&\textbf{subject to:} && P_i^{rl} \le P_i^r \le P_i^{ru}, && i \in \mathcal{N}, && \text{(48b)} \\
& && -\bar{P}_{ij} z_{ij} \le P_{ij} \le \bar{P}_{ij} z_{ij}, && (i,j) \in \mathcal{L}, && \text{(48c)} \\
& && P_i^r - P_i^d = \sum_{(i,j) \in \mathcal{L}} P_{ij}, && i \in \mathcal{N}, && \text{(48d)} \\
& && -M_{ij}(1 - z_{ij}) \le P_{ij} - B_{ij}(\theta_i - \theta_j) \le M_{ij}(1 - z_{ij}), && (i,j) \in \mathcal{L}, && \text{(48e)} \\
& && \theta_i^l \le \theta_i \le \theta_i^u, && i \in \mathcal{N}, && \text{(48f)} \\
& && \theta_{\text{ref}} = 0. && && \text{(48g)}
\end{aligned}
$$

Here, P_i^r denotes active-power generation, θ_i is the voltage phase angle at bus i, and P_{ij} is the active-power flow on line (i,j). The parameter B_{ij} denotes the line susceptance and $\bar{P}_{ij}$ its thermal capacity. Constraint (48e) enforces the DC power-flow relation when the line is active and relaxes it through a sufficiently large constant M_{ij} when $z_{ij} = 0$. Constraint (48c) forces the corresponding power flow to zero when the line is switched off. Unlike AC-OTS, the DC formulation removes reactive power, voltage-magnitude variables, and the nonlinear AC branch-flow equations. With the binary topology fixed, the resulting continuous completion problem is therefore a convex quadratic program.

Solution quality and feasibility. CGD achieves the lowest Hamming distance and highest exact reconstruction on all three DC-OPF with branch switching benchmarks while maintaining zero observed downstream infeasibility. Its objective gaps are 0.25%, 1.13%, and 0.17% on the 9-, 197-, and 500-bus systems, respectively, consistently improving over both unconstrained generation and terminal-only correction. In particular, correction throughout generation reduces the infeasibility observed without correction from 6.18%, 5.52%, and 4.85% to zero on the three systems. These results show that constraint-aware generation remains beneficial even when the downstream continuous problem is convex.

Runtime. We compare CGD with GUROBI at matched objective quality, using the time required by GUROBI to first reach the objective gap attained by CGD. On DC-OPF, CGD is slower on the 9-bus system ($0.25\times$), roughly comparable on the 197-bus system ($1.11\times$), and reaches a $16.71\times$ matched-gap speedup on the 500-bus system. In contrast, the AC benchmarks exhibit substantially larger speedups, indicating that the proposed decomposition is most beneficial when combinatorial search is coupled with a difficult nonlinear continuous problem.

Table 5: Network topology prediction and downstream DC-OPF with OTS performance on $N = 1{,}000$ test instances per network. Objective gaps are averaged over instances with a feasible downstream solve; n/a indicates that no such instance exists. Values are reported as mean (standard deviation).

		Discrete quality		Feasibility	Downstream quality
Case	**Method**	**Ham. (%) ↓**	**Exact (%) ↑**	**Infeasible (%) ↓**	**Gap (%) ↓**
9-Bus	**CGD (ours)**	**0.43 (0.04)**	**58.83 (5.61)**	**0.00 (0.00)**	**0.25 (0.02)**
	CGD without projection	3.92 (0.24)	49.12 (5.92)	6.18 (1.21)	1.47 (0.17)
	CGD + final-step projection	1.53 (0.12)	53.14 (8.65)	0.00 (0.00)	1.68 (0.10)
	MLP	8.14 (1.13)	27.01 (5.92)	15.74 (2.01)	2.04 (0.17)
	GNN	10.03 (0.46)	19.74 (2.77)	45.23 (4.19)	2.24 (0.03)
	DIFUSCO	3.61 (0.53)	35.39 (6.85)	9.84 (2.24)	1.51 (0.32)
	LTO-for-MIP	3.81 (0.75)	32.86 (5.41)	0.00 (0.00)	1.43 (0.25)
197-Bus	**CGD (ours)**	**0.65 (0.05)**	**42.59 (5.42)**	**0.00 (0.00)**	**1.13 (0.17)**
	CGD without projection	1.21 (0.38)	32.87 (6.12)	5.52 (0.75)	1.32 (0.29)
	CGD + final-step projection	1.25 (0.31)	34.23 (5.87)	0.00 (0.00)	1.98 (0.21)
	MLP	15.32 (3.54)	0.00 (0.00)	100.00 (0.00)	n/a
	GNN	4.53 (0.77)	28.03 (4.65)	13.74 (3.46)	3.34 (0.23)
	DIFUSCO	5.43 (1.42)	30.67 (3.90)	10.42 (1.05)	3.41 (0.58)
	LTO-for-MIP	6.34 (1.42)	23.91 (4.48)	0.00 (0.00)	2.03 (0.32)
500-Bus	**CGD (ours)**	**1.31** (0.23)	**41.54** (4.32)	**0.00** (0.00)	**0.17** (0.04)
	CGD without projection	3.51 (0.63)	31.64 (4.82)	4.85 (0.60)	1.12 (0.27)
	CGD + final-step projection	4.16 (0.59)	29.31 (3.86)	0.00 (0.00)	1.67 (0.25)
	MLP	8.43 (1.74)	0.00 (0.00)	100.00 (0.00)	n/a
	GNN	6.83 (0.98)	20.43 (4.92)	0.00 (0.00)	1.89 (0.21)
	DIFUSCO	11.23 (8.25)	8.23 (1.29)	19.85 (4.12)	1.12 (0.23)
	LTO-for-MIP	3.87 (0.44)	38.98 (5.36)	0.00 (0.00)	0.98 (0.25)

Table 6: Average runtime per DC-OPF test instance, in seconds. Speedup is computed relative to the time required by GUROBI to reach the objective gap achieved by CGD.

	DC-OPF with branch switching		
Stage	**9-Bus**	**197-Bus**	**500-Bus**
GUROBI joint solve	0.01	0.12	18.10
GUROBI time to CGD gap	0.01	0.10	12.87
CGD sampling	0.03	0.08	0.76
Continuous completion	0.01	0.01	0.01
CGD end-to-end	**0.04**	**0.09**	**0.77**
Speedup	**0.25×**	**1.11×**	**16.71×**

G PORTFOLIO EXPERIMENTAL DETAILS

G.1 MIXED-INTEGER PORTFOLIO FORMULATION

We consider the following mixed-integer extension of Markowitz portfolio optimization Rubinstein (2002):

$$\begin{aligned} \min_{x,z} \quad & x^\top Q x - \mu^\top x \\ \text{s.t.} \quad & \mathbf{1}^\top x = 1, \\ & 0 \le x_i \le z_i, \qquad i = 1, \dots, n, \\ & z^\top P z \le \rho, \\ & \mathbf{1}^\top z \le k, \\ & z_i \in \{0, 1\}, \qquad i = 1, \dots, n. \end{aligned} \tag{49}$$

Here, μ contains expected returns, Q is the return covariance matrix, $P \succeq 0$ defines a quadratic constraint over the selected assets, and ρ is its budget. CGD predicts the support z and recovers the weights x by solving the induced convex quadratic program.

G.2 Portfolio Correction and Recovery

During reverse diffusion, CGD corrects relaxed selection probabilities toward

$$\mathcal{C}_\xi = \left\{ z \in [0,1]^n | \mathbf{1}^T z \leq k, z^\top P z \leq \rho \right\}.$$

Terminal recovery must satisfy both the quadratic selection constraint and the nonempty-support requirement implied by $\mathbf{1}^\top x = 1$. Relaxed projection alone does not generally guarantee feasibility after thresholding. Appendix A.7 provides a sufficient condition under which the relaxed quadratic slack absorbs the displacement introduced by rounding. We therefore report the observed mean and maximum discrete violation after recovery in addition to downstream objective quality.

G.3 Portfolio Data Generation

Given the problem and data description of Sambharya et al. (2023), we construct a mixed-integer extension of a Markowitz portfolio model by introducing binary selection variables, a cardinality upper bound, and a quadratic constraint on the selection vector. Each entry of the expected-return vector μ is sampled independently from $\mathcal{U}(0,1)$. The quadratic constraint matrix P is generated to be positive semidefinite, and ρ varies across instances to produce different feasible asset-selection problems. The reference MIQP solution supplies the support $z^\star$.

For both $n = 50$ and $n = 150$, the dataset contains 12,000 instances. Both datasets use an $80\%/10\%/10\%$ training-validation-test split, and a separate model is trained for each problem size. The average fraction of selected assets is 30.51% for $n = 50$ and 12.95% for $n = 150$.

H Implementation Details.

Both the AC-OPF and DC-OPF branch-switching benchmarks, as well as the portfolio benchmark, are solved using Gurobi (v13.0.1).

For the AC and DC branch-switching benchmarks, the denoiser is an edge-aware GNN with hidden dimension 256, 6 message-passing layers, 2 conditioning layers, and a 64-dimensional timestep embedding. The network is trained with AdamW using learning rate 2×10^{-4}, weight decay 10^{-6}, and gradient clipping at 1. For the AC-OPF benchmark, we use $T = 100$ denoising steps, Monte Carlo sample size $K = 20$, and batch size 128 during training. Downstream AC-OPF evaluation uses time limits of 60, 3600, and 3600 s for the 9-, 197-, and 500-bus systems, respectively, with a MIP-gap tolerance of 10^{-4}.

For the DC-OPF with branch switching experiments, separate models are trained because the DC and AC formulations produce different reference solutions and network topologies. The DC models are trained for 100 epochs using AdamW; batch sizes are 128, 64, and 32 for the 9-, 197-, and 500-bus systems, respectively.

For the portfolio benchmark, we use $T = 30$ denoising steps and Monte Carlo sample size $K = 16$ during generation. The $n = 50$ model uses hidden dimension 160, 4 transformer layers, 5 attention heads, and batch size 256, while the $n = 150$ model uses hidden dimension 192, 4 transformer layers, 6 attention heads, and batch size 192. Both are trained for up to 300 epochs with AdamW, learning rate 3×10^{-4}, weight decay 10^{-4}, dropout 0.1, and early-stopping patience 50.

Hyperparameters for all learning-based methods are selected using the validation set. Training is performed on NVIDIA GPUs, while optimization is carried out on AMD EPYC/Intel Xeon CPU nodes.