

Accelerating Discrete Diffusion Models for Combinatorial Optimization via Momentum Inference

Bessam Azizi

TBI, Université de Toulouse
CNRS, INRAE, INSA, ANITI
31077 Toulouse, France
azizi@insa-toulouse.fr

Mimoun Mohamed

MIAT, Université de Toulouse
ANITI, INRAE, UR 875
31326 Toulouse, France
mimoun.mohamed@inrae.fr

Sophie Barbe

TBI, Université de Toulouse
CNRS, INRAE, INSA, ANITI
31077 Toulouse, France
sophie.barbe@insa-toulouse.fr

Thomas Schiex

MIAT, Université de Toulouse
ANITI, INRAE, UR 875
31326 Toulouse, France
thomas.schiex@inrae.fr

Abstract—Neural network-based Combinatorial Optimization (CO) solvers are showing increasingly convincing performances on NP-complete (NPC) problems, alleviating the need for hand-crafted heuristics. These methods are usually tested on problems with very informative features, such as coordinates in the Euclidean Traveling Salesman Problem. In this paper, we introduce D2CO-M, a discrete graph denoising diffusion model that learns how to efficiently generate high-quality solutions using an original momentum-based sampling inference strategy. The dynamics of traditional score-matching denoising diffusion can be interpreted as a stochastic gradient ascent guided by the Stein score, the gradient of the log-likelihood. D2CO-M shifts from this simple gradient ascent to a momentum-based sampling strategy fitted for discrete diffusion, inspired by fast momentum-enhanced first-order methods, as exploited in the Adam optimizer. We experiment with this momentum-based variant on two NP-hard optimization problems with complex fine-grained features: random Weighted Max-2-SAT instances, and real-world Cost Function Network instances describing Computational Protein Sequence Design problems, establishing D2CO-M as a competitive and robust tool for complex, real-world combinatorial optimization.

Index Terms—Combinatorial optimization, discrete diffusion models, adaptive momentum, cost function networks, computational protein design, (partial) weighted max-2-sat.

I. INTRODUCTION

Combinatorial Optimization (CO) encompasses some of the most fundamental and computationally intractable challenges in computer science and computational biology. Problems such as Boolean Satisfiability (SAT [1]) and Computational Protein Design (CPD [2–5]) require finding an optimal configuration of discrete variables within a vast search space. Traditionally, solving these NP-complete and NP-hard problems relies on Branch & Bound solvers or highly specialized heuristic searches. While these traditional solvers — such as Conflict-Driven Clause Learning (CDCL) for SAT [1] or ToulBar2 for Cost Function Networks (CFN [6]) — offer robust performance, they demand significant domain expertise and often suffer from exponential worst-case time complexity, making

scaling to large, densely constrained instances a persistent challenge.

Recent advances in deep learning have introduced a promising alternative: data-driven neural solvers capable of learning heuristic search strategies directly from problem instances. Among these, generative diffusion models have recently emerged as a state-of-the-art paradigm. Foundational frameworks, such as DIFUSCO [7], have successfully broadened the scope of neural solvers by casting NP-complete routing and sparse graph tasks as discrete $\{0, 1\}$ -vector optimization problems, leveraging iterative denoising to generate high-quality solutions.

Despite these algorithmic breakthroughs, the reverse inference process still offers opportunities for improvement. Standard discrete and continuous diffusion samplers are essentially memoryless Markovian processes. In their “score-matching” formulation [8], their dynamics at inference time resemble stochastic gradient ascent guided by the Stein score, the gradient of the log-likelihood of the current assignment. When navigating highly rugged, densely conflicting energy landscapes, these standard inference trajectories may stall, oscillate, or become irreversibly trapped in deep local optima. Furthermore, requiring hundreds of sequential denoising steps to reach a feasible solution severely limits the latency advantages of neural solvers.

Motivated by the need for faster and more robust generative trajectories in rugged discrete spaces, we introduce *Momentum Inference*, a novel acceleration and guidance framework for diffusion-based CO solvers. Drawing inspiration from momentum-based techniques in continuous optimization, our approach injects historical “inertia” into the discrete denoising process. By accumulating a memory of previous generative steps, the diffusion model gains the necessary momentum to smooth out highly non-convex energy landscapes, successfully powering through local optima while requiring significantly fewer inference steps.

To rigorously evaluate the versatility and effectiveness of our framework, we apply Momentum Inference to two distinct and highly complex domains: Weighted Max-2-SAT (a dense

NP-hard logical optimization problem) and Cost Function Network-based Computational Protein Design (a real-world computational biology energy minimization problem [2, 3, 9]). Unlike previous diffusion solvers, evaluated primarily on spatial routing tasks with strongly informative features such as plan coordinates in the Euclidean Traveling Salesman Problem (TSP), our targets require resolving intricate, conflicting structural weights from a fine-grained description of the instance.

In summary, our main contributions are as follows:

- We propose Momentum Inference, a novel, plug-and-play inference mechanism that incorporates historical trajectory memory to guide and accelerate the generative denoising process.
- We demonstrate that our approach consistently outperforms standard diffusion sampling, achieving higher-quality solutions and tighter optimality gaps with reduced computational latency across both Weighted Max-2-SAT and CFN-based protein design benchmarks. Our approach also outperforms Loandra, a MaxSAT local search solver that placed first in the 2021 weighted MaxSAT evaluation and is competitive with SPB-MaxSAT, a very recent local search solver that won the 2024 evaluation.

II. RELATED WORKS

A. Diffusion Models for Combinatorial Optimization

The application of generative diffusion models to Combinatorial Optimization (CO) represents a recent and promising paradigm shift. A foundational work in this domain is DIFUSCO (Diffusion-based Combinatorial Optimization) [7], which demonstrated the viability of this approach by casting NP-complete problems as discrete $\{0, 1\}$ -vector optimization tasks and leveraging graph-based discrete diffusion models driven by Bernoulli noise.

Building upon this foundation, subsequent research has expanded the flexibility and efficiency of diffusion in CO. Representative advancements include DiffUCO [10, 11], which enables unsupervised optimization by using an objective that upper-bounds the reverse KL divergence, and the development of scalable discrete diffusion samplers that introduce memory-efficient training via policy-gradient objectives.

However, the current literature, even when operating in discrete state spaces, predominantly focuses on routing problems, such as the Traveling Salesman Problem (TSP), or sparse graph tasks, such as the Maximum Independent Set (MIS) [7, 10–14]. These problems benefit from distinct structural biases, such as highly informative features (in the Euclidean TSP), spatial locality or graph sparsity, which align well with the inductive biases of current graph-based diffusion frameworks.

Applying discrete diffusion models beyond these domains to problems with dense, conflicting logical constraints or highly rugged energy landscapes and arbitrary costs has not been explored yet. In this paper, we bridge this gap by extending diffusion-based solvers to two complex, non-routing domains: Weighted Max-2-SAT and Computational Protein Design (CPD). Unlike the Euclidean TSP or MIS, Weighted

Max-2-SAT requires resolving dense networks of conflicting Boolean clauses and is mathematically equivalent to Quadratic Unconstrained Optimization (QUBO) and Ising formulations of NP-hard problems [15]. CPD involves navigating complex energy landscapes modeled via Cost Function Networks (CFNs). By targeting these domains, we demonstrate the broader applicability of discrete diffusion models for highly constrained, real-world optimization problems.

B. Traditional Solvers and Learning-based Approaches

While generative diffusion approaches to combinatorial optimization are rapidly evolving, classical algorithms and hybrid machine-learning solvers remain highly relevant in their respective domains. For Weighted Max-2-SAT, traditional resolution relies on established exact methods and advanced local search heuristics capable of efficiently guiding the solution space exploration [16–18]. Concurrently, the integration of deep learning with Boolean satisfiability has rapidly expanded since the pioneering GNN-based NeuroSAT [19]. Recent learning-based advancements include optimizing search heuristics via deep reinforcement learning [20–22], utilizing graph contrastive learning and structural formula reconstruction [23–25], and leveraging GPU-accelerated continuous relaxations for unsupervised resolution [26]. However, it is important to note that the current deep learning literature in this domain focuses predominantly on solving pure SAT or unweighted Max-2-SAT problems. Extending these neural architectures to effectively manage weighted clauses remains a significant challenge.

Similarly, in Computational Protein Design (CPD), exact conformational search and energy minimization are well established [2]. When CPD is modeled as a Cost Function Network (CFN), exact combinatorial solvers such as ToulBar2 achieve state-of-the-art performance by leveraging depth-first branch-and-bound coupled with soft local consistencies [27]. In addition to branch-and-bound techniques, advanced continuous relaxations such as LR-BCD (Efficient Low Rank Convex Bounds for Pairwise Discrete Graphical Models [28]) provide highly efficient and scalable convex bounding methods to navigate these complex discrete pairwise energy landscapes.

Despite the efficacy of these domain-specific solvers and ML-augmented heuristics, standard algorithms often suffer from exponential worst-case scaling, while current SAT-specific neural approaches struggle to generalize to weighted constraints or energy models like CFNs. Our proposed diffusion framework does not aim to outright replace these specialized solvers. Instead, we focus on fundamentally enhancing the generative process itself: by incorporating Momentum Inference, we significantly accelerate classical sampling trajectories and consistently yield high-quality solutions. This approach provides a generalized, highly parallelizable framework capable of competitively tackling both weighted logical constraints and rigorous protein design problems.

III. PROBLEM FORMULATIONS & BACKGROUND

In this paper, scalar variables are denoted by lowercase italic letters (e.g., x, y, z), and vectors or tuples are denoted by bold lowercase letters (e.g., $\mathbf{x}, \mathbf{y}, \mathbf{z}$). Sets are represented by calligraphic uppercase letters (e.g., $\mathcal{X}, \mathcal{C}, \mathcal{D}$), whereas matrices are represented by bold uppercase letters (e.g., $\mathbf{Q}, \mathbf{P}$). Assignments of variables are represented by Greek letters such as τ , and the value of a variable x in an assignment τ is denoted as $\tau[x]$.

A. Weighted Max-2-SAT Formulation

The Maximum 2-Satisfiability (Max-2-SAT) problem is a classic NP-hard combinatorial optimization challenge. An instance of Max-2-SAT is defined by a set of N Boolean variables, $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$, and a collection of M 2-clauses, $\mathcal{C} = \{c_1, c_2, \dots, c_M\}$. Each clause c_j is a disjunction (OR operation) of two literals, where a literal is either a variable x_i or its negation $\neg x_i$ (e.g., $c_j = x_1 \vee \neg x_2$). A clause is said to be satisfied by a truth assignment $\tau : \mathcal{X} \rightarrow \{0, 1\}$ when at least one of its literals is true in the assignment, and violated otherwise. In Weighted Max-2-SAT, each clause c_j is assigned a positive weight $w_j > 0$. The objective is to find a truth assignment τ that maximizes the total weight of the satisfied clauses or, equivalently, that minimizes the total weight of falsified clauses.

$$\min_{\tau} \sum_{j=1}^M w_j \cdot \mathbb{I}(\tau \not\models c_j) \quad (1)$$

where $\mathbb{I}(\tau \not\models c_j)$ is an indicator function that outputs 1 if the assignment τ violates clause c_j , and 0 otherwise.

B. Cost Function Networks

Cost Function Networks (CFNs) are used to represent and solve discrete optimization problems. Formally, a CFN is defined as a triplet $(\mathcal{X}, \mathcal{D}, \mathcal{W})$, where:

- $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ is a set of N discrete variables.
- $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_N\}$ is the set of domains, where each domain $\mathcal{D}_i \ni x_i$ contains the finite set of values that variable x_i can take.
- $\mathcal{W}$ is a collection of cost functions. Each cost function $w_S \in \mathcal{W}$ defined over a subset of variables $S \subseteq \mathcal{X}$ (its scope) and maps their assignments to a local cost: $w_S : \prod_{i \in S} \mathcal{D}_i \rightarrow \mathbb{N} \cup \{\infty\}$.

Given a CFN, the weighted constraint satisfaction problem (WCSP) is to find a global assignment τ for all variables in $\mathcal{X}$ that minimizes the total aggregated cost $W(\tau) = \sum_{w_S \in \mathcal{W}} w_S(\tau[S])$.

$$\min_{\tau} W(\tau) = \min_{\tau} \sum_{w_S \in \mathcal{W}} w_S(\tau[S]) \quad (2)$$

Thanks to their close relationship with Markov Random Fields [29], CFNs can be used to represent joint multivariate discrete probability distributions as $P(\cdot) \propto \exp(-W(\cdot))$ [30].

C. Mapping Computational Protein Design (CPD) to a CFN

Computational Protein Design (CPD) [3] aims to design new proteins algorithmically. This is motivated by the fact that proteins, the molecules that sustain most life processes, are tiny molecular machines with crucial biochemical and biological functions. They can, for instance, fight viral infections as antibodies or catalyze chemical reactions as enzymes. Natural proteins are actively exploited in both health and industrial contexts, but the catalog of functions that nature provides is too limited to satisfy all existing needs. Thus, designing new proteins computationally is of major interest.

Proteins are linear polymers composed of elementary bricks called amino acids. There are 20 natural amino acids. The chemical composition of a protein is therefore entirely determined by a sequence of characters over an alphabet of size 20. When proteins are synthesized, most of them fold into a three-dimensional structure that determines their function. The Computational Protein Design (CPD) problem, also called the inverse folding problem, is therefore to find a sequence of amino acids that folds into a desired 3D protein structure [3, 31, 32] (the inverse of what AlphaFold does [33]). This inverse folding problem has been recently addressed by the Effie neuro-symbolic architecture [9, 34] where a deep generative model, given a target 3D structure χ , predicts the probability distribution $P(\mathbf{s}|\chi)$ over amino acid sequences $\mathbf{s}$ as a Cost Function Network (CFN). This CFN is then optimized to identify the most probable sequence. The CFN can be described as follows:

- 1) **Variables:** Each variable $x_i \in \mathbf{X}$ corresponds to a specific amino acid position.
- 2) **Domains:** The domain $\mathcal{D}_i$ represents the identities of the 20 natural amino acids.
- 3) **Pairwise Cost Functions:** the binary cost $w_{ij}(u, v)$ over variables x_i and x_j captures the interactions between the amino acid identity u at position i and the amino acid identity v at position j .

Under this mapping, identifying the most probable sequence is exactly equivalent to finding the minimum of the joint cost function of the CFN:

$$W(\tau) = \sum_{w_{ij} \in \mathcal{W}} w_{ij}(\tau[x_i], \tau[x_j]) \quad (3)$$

By casting CPD as a CFN, graph-based discrete diffusion frameworks can treat amino acid residues as nodes and pairwise costs as weighted edges, translating a complex structural biology challenge into a discrete vector optimization task.

D. Continuous Diffusion Models

Continuous diffusion models are latent variable generative models: given a dataset sampled from an unknown density, they learn how to sample from this density. They are characterized by a forward and a reverse Markov process [8, 35–37]. The forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$ corrupts the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ with Gaussian noise into a sequence of increasingly noisy latent variables

$\mathbf{x}_{1:T} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$. The learned reverse Markov process $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ gradually denoises the latent variables towards the data distribution.

Under a continuous-time framework [8], the complex training objective of diffusion models simplifies into an intuitive task: learning to estimate the gradient of the log-likelihood of a simple distribution density that slowly becomes the data distribution. Indeed, using Tweedie’s formula, it has been shown that predicting the injected noise is mathematically identical to estimating this gradient, also known as the Stein score [8, 36]. During inference, the reverse sampling process uses these learned gradients to guide a noisy starting state toward areas of high data density. This generation process behaves much like a first-order optimization method (such as stochastic gradient ascent), maximizing the log-likelihood — with the unique twist that the landscape being optimized gradually sharpens and evolves as the noise level t drops to zero. This analogy has been exploited, in the continuous case, to improve inference by shifting from simple stochastic gradient ascent to more advanced momentum-based optimization algorithms [38]. In this paper, we explore the possibility of using such an accelerated optimization scheme in the context of discrete diffusion models applied to combinatorial optimization tasks.

E. Discrete Diffusion for combinatorial optimization

In a combinatorial optimization setting, every variable x_i of the problem will be represented by a random discrete variable with K categories or values, K matching the cardinality of the domain of the discrete variable (2 for Max-2-SAT and 20 for CPD as CFN). Each random variable x will be associated with a categorical probability distribution that can be represented as a vector of probabilities summing to one, one for each value.

To train the discrete diffusion model, we start from a dataset of pairs (φ_j, τ_j) where φ_j is the description of a discrete optimization instance and τ_j is a high-quality (ideally optimal) assignment of the variables of φ_j . For the data point (φ_j, τ_j) , the forward noising process over time t starts from $\mathbf{x}_0 = \tau_j$ and is defined as [39, 40]:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \text{Cat}(\mathbf{x}_t; \mathbf{p} = \tilde{\mathbf{x}}_{t-1} \mathbf{Q}_t) \quad (4)$$

Where $\text{Cat}(\cdot; \mathbf{p})$ denotes a categorical distribution over the discrete variable $\tilde{\mathbf{x}}_{t-1}$, with transition probabilities parameterized by the vector $\mathbf{p}$, and where $\tilde{\mathbf{x}}_{t-1}$ is the one-hot vector representation of $\mathbf{x}_{t-1}$. Here, $\mathbf{Q}_t \in \mathbb{R}^{K \times K}$ is a time-dependent Markov transition probability matrix, where the entry $[\mathbf{Q}_t]_{uv}$ dictates the probability of transitioning from value u to value v at time step t . The corrupted state at an arbitrary time step t can be computed directly from the initial optimal assignment data distribution $\mathbf{x}_0$ as:

$$q(\mathbf{x}_t | \mathbf{x}_0) = \text{Cat}(\mathbf{x}_t; \mathbf{p} = \tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_t) \quad (5)$$

where $\overline{\mathbf{Q}}_t = \mathbf{Q}_1 \mathbf{Q}_2 \dots \mathbf{Q}_t$ represents the cumulative product of the transition matrices up to time t . The transition matrix can be written as:

$$[\mathbf{Q}_t]_{ij} = \begin{cases} 1 - \frac{K-1}{K} \beta_t & \text{if } i = j, \\ \frac{1}{K} \beta_t & \text{if } i \neq j. \end{cases} \quad (6)$$

Here β_t denotes the corruption rate. The posterior at time $t-1$ can be obtained by Bayes’ Theorem:

$$\begin{aligned} q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} \\ &= \text{Cat} \left(\mathbf{x}_{t-1}; \mathbf{p} = \frac{\tilde{\mathbf{x}}_t \mathbf{Q}_t^T \odot \tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_{t-1}}{\tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_t \tilde{\mathbf{x}}_t^T} \right) \end{aligned} \quad (7)$$

where $\odot$ denotes element-wise multiplication.

As stated by Austin et al. [39], the denoising neural network is trained to predict the clean data $\mathbf{x}_0$, here high-quality assignments of the considered instance, as a multidimensional factorized categorical distribution $p_\theta(\tilde{\mathbf{x}}_0 | \mathbf{x}_t)$. The reverse process is obtained by substituting the predicted $\tilde{\mathbf{x}}_0$ as $\mathbf{x}_0$ in Eq. (7):

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \sum_{\tilde{\mathbf{x}}_0} q(\mathbf{x}_{t-1} | \mathbf{x}_t, \tilde{\mathbf{x}}_0) p_\theta(\tilde{\mathbf{x}}_0 | \mathbf{x}_t) \quad (8)$$

IV. PROPOSED METHODOLOGY

A. Graph-Neural Denoiser Architecture

To gradually denoise an initially random assignment of all variables into a high-quality assignment of a given instance φ , we need to train a denoising neural net that accepts a description of the instance φ and an assignment τ of the variables of φ as input. The neural net then predicts categorical distributions over the variables of the instance φ , from which higher-quality assignments should be eventually sampled.

Following the foundational DIFUSCO framework [7], we adopt the Anisotropic Graph Neural Network (AGNN) architecture with edge gating mechanisms for this denoising architecture. Standard Graph Neural Networks, such as Graph Convolutional Networks (GCNs), are fundamentally designed to update node embeddings. In contrast, the AGNN architecture computes and updates both node and edge representations simultaneously. These edge representations are essential to receive the embeddings that encode the definition of the instances. For our two target problems, Weighted Max-2-SAT and Computational Protein Design, we define distinct edge feature encodings tailored to each problem formulation. For Weighted Max-2-SAT, the encoding for each edge consists of a 4-dimensional cost vector representing the violation costs of all 4 possible assignments of the 2 variables defining the edge. Similarly, for the CFN-based CPD instances, each edge is encoded as a vector of size 400 (derived by flattening the 20×20 cost matrix for the two variables defining the edge), which explicitly represents the pairwise energy/cost terms across all possible amino acid identity pairs.

Formally, let $\mathbf{h}_i^l$ and $\mathbf{e}_{ij}^l$ denote the feature vectors of node i and edge ij at layer l , respectively. To condition the denoising network on the current diffusion stage, the scalar timestep

t is projected into sinusoidal features and injected into the edge updates. The feature propagation at each layer relies on an anisotropic message passing scheme. First, an intermediate edge representation is computed as:

$$\hat{e}_{ij}^{l+1} = \mathbf{P}^l e_{ij}^l + \mathbf{Q}^l h_i^l + \mathbf{R}^l h_j^l$$

where $\mathbf{P}^l$, $\mathbf{Q}^l$, and $\mathbf{R}^l$ are learnable weight matrices. The final edge embeddings are then updated using a Multi-Layer Perceptron (MLP) and the embedding $\mathbf{t}$ of the timestep t :

$$\mathbf{e}_{ij}^{l+1} = \mathbf{e}_{ij}^l + \text{MLP}_e(\text{BN}(\hat{\mathbf{e}}_{ij}^{l+1})) + \text{MLP}_t(\mathbf{t}) \quad (9)$$

where BN denotes Batch Normalization. Concurrently, the node representations are updated by aggregating the neighborhood features, which are modulated by a sigmoid gate (σ) applied to the intermediate edge representations:

$$h_i^{l+1} = h_i^l + \text{ReLU}(\text{BN}(\mathbf{U}^l h_i^l + \mathcal{A}_{j \in \mathcal{N}(i)}(\sigma(\hat{e}_{ij}^{l+1}) \odot \mathbf{V}^l h_j^l)))$$

where $\mathcal{A}$ is a SUM pooling aggregation function, and $\odot$ represents the element-wise product. Following the established configuration for high-performance combinatorial solvers, our backbone network utilizes a 12-layer AGNN with a hidden dimension width of 256 as in [7].

B. Adaptive Sampling for Standard Discrete Diffusion

To accelerate and stabilize inference, we adapt the training-free Adaptive Momentum (AM) framework originally proposed for continuous image spaces [38] to the categorical logit space setting of discrete diffusion models. Standard discrete diffusion samplers rely solely on localized, noisy update signals at each discrete step t , completely ignoring the global trajectory across time steps. By treating the reverse sampling sequence as an optimization path analogous to neural network training, we can accumulate historical direction vectors to smooth out erratic transitions and help the model effectively escape local optima during generation.

In the continuous case, the gradient followed during denoising is directly predicted by the denoising network [8]. In the standard discrete formulation, the network’s final output is typically a valid categorical probability distribution $p_\theta(\bar{\mathbf{x}}_0 | \mathbf{x}_t)$ residing strictly on the probability simplex (bounded in $[0, 1]$ and summing to 1). Applying momentum arithmetic—such as unconstrained addition, subtraction, or scalar multiplication—directly to these probabilities is mathematically ill-posed, as it would immediately violate the simplex constraints.

To operate safely within a continuous, unconstrained vector space, we must shift our optimization trajectory to the logit space. Specifically, we formally define $z_\theta^{(t)} \in \mathbb{R}^{N \times K}$ as the unnormalized, pre-Softmax raw logits produced by the final layer of our AGNN denoiser. By maintaining and updating our running state z_t entirely within this unconstrained space, we can cleanly compute directional changes via finite difference approximations:

$$dz_t = z_\theta^{(t)} - z_t \quad (10)$$

This difference dz_t acts analogous to a velocity vector in continuous optimization. It updates a running momentum

matrix m_{t-1} , which is controlled by the usual damping coefficient a and update velocity scale b . The valid categorical probability distributions required to sample the next discrete state $\mathbf{x}_{t-1}$ are only recovered at the end of the update step via a standard Softmax transformation over the adjusted logits. Simultaneously, we must prevent momentum from washing out fine-grained details—analogueous to over-smoothing in image generation [41]. To achieve this, we use an RMSProp-inspired second-order moment accumulator $v_{t-1}^{(n)}$ that tracks exponentially decaying squared increments. By scaling the update inversely by $\sqrt{v_{t-1}^{(n)}}$ [38], this variable-specific regulator dynamically paces the sampler to balance stable global structures with volatile localized values before a final Softmax transformation yields the probability distributions.

Algorithm 1: Adaptive Momentum Sampling for Discrete Diffusion

Input: Discrete diffusion model $\text{Model}(\cdot)$, total time steps T , hyperparameters $a, b, c > 0$, smoothing term $\zeta > 0$.

Output: Final discrete variable sequence $\mathbf{x}_0$.

- 1 Initialize random assignment from prior distribution:
 $\mathbf{x}_T \sim q(\mathbf{x}_T)$
 - 2 Initialize noise logits: $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 3 Initialize tracking buffers:
 $\mathbf{m}_T \leftarrow \mathbf{0}, \quad v_T^{(n)} \leftarrow 1 \quad \forall n \in \{1, \dots, N\}$
 - 4 **for** $t = T, T-1, \dots, 1$ **do**
 - 5 Predict clean logits: $\mathbf{z}_\theta^{(t)} \leftarrow \text{Model}(\mathbf{x}_t, t)$
 - 6 Calculate logit update direction: $d\mathbf{z}_t \leftarrow \mathbf{z}_\theta^{(t)} - \mathbf{z}_t$
 - 7 **for each variable** $n = 1$ **to** N **do**
 - 8 $v_{t-1}^{(n)} \leftarrow (1 - c) \cdot v_t^{(n)} + c \cdot \|d\mathbf{z}_t^{(n)}\|_2^2$
 - 9 $\mathbf{m}_{t-1}^{(n)} \leftarrow a \cdot \mathbf{m}_t^{(n)} + b \cdot d\mathbf{z}_t^{(n)}$
 - 10 $\mathbf{z}_{t-1}^{(n)} \leftarrow \mathbf{z}_t^{(n)} + \frac{\mathbf{m}_{t-1}^{(n)}}{\sqrt{v_{t-1}^{(n)} + \zeta}}$
 - 11 Update discrete state:
 $\mathbf{x}_{t-1} \sim \text{Cat}(\mathbf{x}_{t-1}; \mathbf{p} = \text{softmax}(\mathbf{z}_{t-1}))$
 - 12 **return** $\mathbf{x}_0$
-

V. EXPERIMENTAL SETUP & OPTIMIZATION TASKS

A. Experimental Setup for Weighted Max-2-SAT

1) *Synthetic Instance Generation:* To evaluate our framework on the Weighted Max-2-SAT problem, we construct a comprehensive dataset using the `randomKCNF` generator from the `cnfgen` package [42]. To generate a unique 2-SAT clause, a subset of $k = 2$ distinct variables is sampled uniformly at random without replacement. For each selected variable, a random boolean choice determines the sign of the variable in the clause. This sampling process is repeated to yield a collection of M unique clauses, discarding duplicate clauses or tautologies (e.g., $x_i \vee \neg x_i$) to preserve non-trivial combinatorial landscapes.

Our complete dataset comprises 100,000 unique instances, with 90,000 instances for training, 5,000 for validation, and

5,000 for testing. For each instance, the number of Boolean variables N is sampled uniformly in the range $[50, 150]$. The clause density is governed by the constraint parameter $\alpha = M/N$. In random k -SAT, this parameter is known to play a critical role in determining the satisfiability of instances. For $k = 2$, a transition from satisfiable to unsatisfiable instances is observed around $\alpha = 1$ [43]. To ensure structural diversity across the distribution, α is chosen randomly for each instance within the range $[2, 8]$, spanning the spectrum from unsatisfiable and slightly constrained to more heavily over-constrained regimes. Each clause c_j is then assigned an integer weight w_j sampled uniformly from $[1, 10]$.

2) *Optimization and Baselines Protocol*: To provide reliable target labels for supervision during training and a precise baseline for performance evaluation during inference, all generated instances are solved exactly using **Akmaxsat**, a state-of-the-art exact Max-SAT solver [44]. The solver computes the absolute ground-truth assignment that maximizes the satisfied clause weight, yielding an optimal total weight W_{exact} . Max-2-SAT being NP-hard, the running time of this exact solver grows exponentially with the number of variables N in the worst case. To make this computationally feasible, we restricted N to the range $[50, 150]$.

Finally, to ensure uniform scaling across heterogeneous problem matrices, the individual clause weights w_j within a given problem instance are normalized relative to the maximum clause weight present in that specific instance, defined as:

$$\hat{w}_j = \frac{w_j}{\max_k w_k} \quad (11)$$

This bounds all input weights within the interval $[0, 1]$, isolating the underlying graph topology from purely arbitrary scalar shifts in the weight distributions. Note that by the Max-2-Sat problem definition, the set of optimal solutions is unchanged (invariant) by this rescaling.

During inference, to benchmark our neural framework, we evaluate it against two prominent state-of-the-art incomplete heuristic solvers: **Loandra** [45] and **SPB-Maxsat** [18]. Both solvers have secured multiple victories and top-tier placements in recent international MaxSAT Evaluations: **Loandra** is widely recognized for its historical dominance in the incomplete solver track, while **SPB-Maxsat** represents the bleeding edge of modern heuristic resolution (winner in the 2024 MaxSAT Evaluation for incomplete solvers).

B. Experimental Setup for Cost Function Networks (CPD)

To evaluate the performance of our model on highly constrained Cost Function Networks (CFNs), we benchmarked against the foundational real-world task of Computational Protein Design (CPD [2, 34]). Our experimental pipeline is structured as follows:

1) *Dataset and Instance Formulation*: Following standard protocols in deep learning-based CPD evaluations, we used the CATH protein structure dataset curated by Ingraham et al. [46]. To ensure rigorous evaluation, we relied on the standard partitioning of the dataset into 18,007 structures for

training, 607 for validation, and 1,117 for testing. The protein sizes in the dataset can reach 500 amino acids, resulting in optimization problems with very vast search spaces of 20^{500} .

For each protein structure χ , a pairwise CFN was produced using the neuro-symbolic framework **Effie**, a deep learning-based scoring model [9, 34].

2) *Optimization and Baselines Protocol*: To obtain target training labels and compute comparative baselines under a tight computational budget, we rely on the approximate LR-BCD (Low-Rank Semidefinite Programming via Block Coordinate Descent) solver [28] used in the original **EffieDes** pipeline [9]. Furthermore, each CFN-based CPD instance is systematically translated into a Partial Weighted Max-2-SAT formulation. In this encoding, hard at-most-one and at-least-one clauses are used to enforce that exactly one amino acid type is selected per residue position. Concurrently, soft 2-clauses are used to encode the coarse-grained pairwise energy costs predicted by **Effie**. This direct translation allows us to benchmark our framework against the incomplete solvers **Loandra** and **SPB-Maxsat**.

To assess the biological relevance of the generated solutions, we rely on the standard evaluation criterion for Computational Protein Design algorithms that compares the chemical composition of the computationally designed protein for a given 3D structure against the natural (or native) sequence observed in nature for that same structure. The percentage of amino acids that are correctly recovered by the CPD algorithm is called the Native Sequence Recovery (NSR).

C. Implementation and Hardware Details

The **D2CO** framework was implemented in Python 3.12 and PyTorch library. All neural network training, inference, and continuous optimization experiments were executed on an Ubuntu Linux server powered by two AMD EPYC 9124 CPUs. Deep learning workloads were run on an NVIDIA H200 NVL GPU. To guarantee a rigorous and consistent evaluation environment, classical baseline solvers (**AkmaxSat**, **Loandra** and **SPB-Maxsat**) were compiled and isolated in single-threaded execution modes on the same CPU architecture.

D. Model Complexity and Hyperparameters

The architectural scale of **D2CO** for both problem domains is detailed in Table I. Due to the graph-encoded nature of our framework, the parameter count remains very compact, ensuring fast inference.

To tune the adaptive momentum sampling algorithm, we conducted a grid search over the parameters a , b , and c . Our results align with the configurations reported by [38], confirming that $a = 0.9$, $b = 0.01$, and $c = 0.01$ provides the most robust performance.

VI. EXPERIMENTAL RESULTS

A. Performance on Weighted Max-2-SAT

Table II presents the comparative evaluation on Weighted Max-2-SAT instances across increasing scales (from $N \in [150, 250]$ up to $N \in [400, 600]$) and clause densities (α).

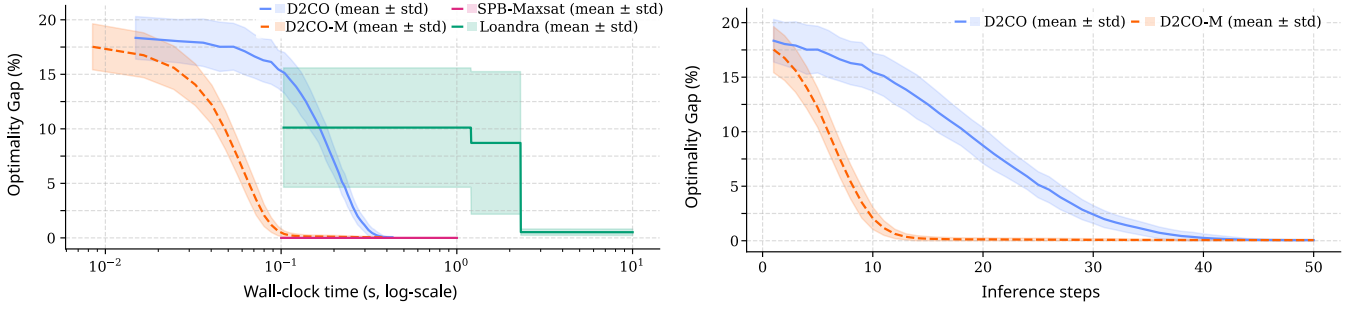


Fig. 1: Evaluation on 100 problem instances for the first OOD configuration. Each curve represents an average with a shaded standard-deviation (left) Optimality gap (%) over wall-clock time, comparing our approach against traditional heuristic solvers. (right) Optimality gap (%) across discrete sampling steps during inference.

TABLE I: Parameters and Hyperparameter Configurations

Hyperparameter / Metric	Weighted Max-2-SAT	CFN (CPD)
Graph Layers (L)	12	12
Hidden Dimension (d)	256	256
Total Trainable Parameters	5,335,810	5,451,028
Learning Scheduler	cosine-decay	cosine-decay
Learning Rate	2×10^{-4}	2×10^{-4}
Weight Decay	1×10^{-4}	1×10^{-4}
Optimizer	AdamW	AdamW
Batch Size	512	64
Diffusion Steps	1000	1000
Inference Steps	50	50
Diffusion Schedule	cosine	cosine
Total Training Time	$\approx$ 4 Hours	$\approx$ 5 Hours

Because these configurations extend well beyond the training set sizes, this assessment provides a rigorous empirical evaluation of the neural solver’s ability to generalize outside its training distribution. To contextualize these results, we evaluate performance primarily using the *optimality gap*. Let $\mathcal{W}(\tau)$ denote the objective score (i.e., the total weight of satisfied clauses) for a given assignment τ . The optimality gap (%) for a predicted assignment $\hat{\tau}$ relative to the optimal (or best-known) assignment τ^* is formally defined as:

$$\text{Gap (\%)} = \frac{\mathcal{W}(\tau^*) - \mathcal{W}(\hat{\tau})}{\mathcal{W}(\tau^*)} \times 100 \quad (12)$$

A lower gap indicates a solution closer to the absolute optimum, with a gap of 0% denoting that the predicted solution achieves the exact optimal satisfied weight.

As expected, traditional exact solvers struggle significantly with worst-case scalability. While AkmaxSat was utilized to compute the exact optimal assignments for the smallest instance scale (averaging over 2,453 seconds), its exponential time complexity rendered it computationally impractical for the medium ($N \in [250, 400]$) and large ($N \in [400, 600]$) configurations. For these two scales, computing the absolute optimum requires excessive execution times. Consequently, for the medium and large configurations, the reference assignment τ^* used to compute the optimality gap is defined as the best-known solution discovered by the state-of-the-art heuristic SPB-Maxsat.

Regarding incomplete heuristic solvers, while Loandra operates within a strict 10-second time budget, it suffers from a clear degradation in solution quality; its optimality gap widens from 0.50% to 1.38% as the problem size increases. Furthermore, it should be noted that a direct sub-second comparison with SPB-Maxsat during inference is technically constrained, as the solver’s timeout parameter strictly requires integer values. As instance complexity scales up, SPB-Maxsat requires increasingly longer search times to avoid severe degradation in solution quality. For medium instances, a computational budget of 2 seconds was necessary. On the largest configurations, this budget had to be extended to 5 seconds to maintain high-quality solutions. These results, now reported in Table II, highlight the inherent worst-case scaling limitations of traditional local search heuristics when deployed in time-sensitive, sub-second scenarios where our neural solver, D2CO-M, naturally operates.

In contrast, the generative diffusion approaches demonstrate remarkable zero-shot generalization capabilities, presenting a distinct trade-off between inference speed and solution quality. As illustrated by the temporal convergence comparison in Figure 1a, our proposed momentum variant, D2CO-M, emerges as the fastest model evaluated, drastically outperforming the heuristic baselines over wall-clock time. By injecting momentum into the reverse sampling trajectory, it systematically accelerates the sampling process compared to the base D2CO model, maintaining sub-second resolution times (from 0.39 seconds on small instances to 0.46 seconds on the largest). Remarkably, it achieves this speed while tightening the optimality gap to an exceptional 0.05% across small and medium scales, making it the optimal choice for strict real-time applications.

Importantly, the inference times reported in Table II were recorded using a fixed budget of 50 sampling steps to ensure a fair baseline comparison. However, as further detailed in our trajectory analysis (Figure 1), Momentum Inference exhibits a dramatically faster convergence rate, stabilizing at high-quality solutions after only 20 steps. Consequently, the effective inference time required by D2CO-M can be reduced in practice to less than 0.2 seconds, further emphasizing its computational efficiency.

To rigorously validate the impact of the proposed momen-

TABLE II: Comparative evaluation metrics across out-of-distribution (OOD) problem configurations.

Solver / Model	$N \in [150, 250], \alpha \in [4, 8]$			$N \in [250, 400], \alpha \in [6, 10]$			$N \in [400, 600], \alpha \in [8, 12]$		
	Gap (%)	Time (s)	GPU Mem. (MB)	Gap (%)	Time (s)	GPU Mem. (MB)	Gap (%)	Time (s)	GPU Mem. (MB)
AkmaxSat	–	2453.49 $\pm$ 7073.30	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Loandra	0.50 $\pm$ 0.32	10.0 $\pm$ 0.0	N/A	0.93 $\pm$ 0.33	10.0 $\pm$ 0.0	N/A	1.38 $\pm$ 0.37	10.0 $\pm$ 0.0	N/A
SPB-Maxsat	0.0 $\pm$ 0.0	1.0 $\pm$ 0.0	N/A	–	2 $\pm$ 0.0	N/A	–	5 $\pm$ 0.0	N/A
D2CO	0.06 $\pm$ 0.04	0.41 $\pm$ 0.02	73.25 $\pm$ 4.49	0.06 $\pm$ 0.05	0.45 $\pm$ 0.03	95.14 $\pm$ 7.64	0.06 $\pm$ 0.03	0.48 $\pm$ 0.02	133.72 $\pm$ 11.88
D2CO-M	0.05 $\pm$ 0.05	0.39 $\pm$ 0.02	73.25 $\pm$ 4.49	0.05 $\pm$ 0.04	0.43 $\pm$ 0.03	95.14 $\pm$ 7.64	0.06 $\pm$ 0.03	0.46 $\pm$ 0.02	134 $\pm$ 11.9
D2CO (Top-$K=3$)	0.02 $\pm$ 0.03	0.48 $\pm$ 0.03	114.53 $\pm$ 13.34	0.03 $\pm$ 0.02	0.59 $\pm$ 0.06	180.41 $\pm$ 22.94	0.04 $\pm$ 0.02	0.80 $\pm$ 0.07	295.9 $\pm$ 35.5
D2CO-M (Top-$K=3$)	0.04 $\pm$ 0.05	0.45 $\pm$ 0.02	114.54 $\pm$ 13.34	0.04 $\pm$ 0.03	0.56 $\pm$ 0.05	180.42 $\pm$ 22.94	0.05 $\pm$ 0.03	0.77 $\pm$ 0.07	295.91 $\pm$ 35.49
D2CO (Top-$K=5$)	0.02 $\pm$ 0.03	0.55 $\pm$ 0.04	156.1 $\pm$ 22.81	0.02 $\pm$ 0.02	0.77 $\pm$ 0.07	266.2 $\pm$ 37.85	0.03 $\pm$ 0.02	1.14 $\pm$ 0.01	458.53 $\pm$ 59.76
D2CO-M (Top-$K=5$)	0.03 $\pm$ 0.05	0.53 $\pm$ 0.03	156.2 $\pm$ 22.81	0.03 $\pm$ 0.03	0.74 $\pm$ 0.07	266.21 $\pm$ 37.86	0.04 $\pm$ 0.03	1.1 $\pm$ 0.01	458.57 $\pm$ 59.76

tum mechanism, we conducted a non-parametric Wilcoxon signed-rank test [47] comparing D2CO-M directly against the standard D2CO across a dedicated sample of 1,000 instances for each problem configuration. We defined the null hypothesis (H_0) as there being no difference in the median performance metrics either execution time or optimality gap between the two diffusion models. For execution time, the statistical analysis rejects H_0 (p -value $< 10^{-4}$), confirming that the momentum mechanism in D2CO-M provides a highly significant speedup. Conversely, for the optimality gap, we fail to reject H_0 , as the test did not reveal a statistically significant difference between the two variants. This confirms that D2CO-M successfully accelerates the standard discrete diffusion process without compromising the quality of the generated solutions.

Conversely, when absolute precision is prioritized over execution speed, the parallel sampling strategies (Top- K) provide the best overall solution quality. By performing parallel sampling across multiple candidate trajectories, the Top- $K=5$ variants achieve the lowest optimality gaps in the evaluation (reaching 0.02% for D2CO and 0.03% for D2CO-M on medium configurations). This improvement in accuracy comes with a corresponding, yet manageable, increase in inference time (up to 1.14 seconds) and GPU memory footprint due to the simultaneous processing load. Ultimately, these results highlight the flexibility of our framework: D2CO-M excels as a lightweight, time-critical solver, while the Top- K strategies dominate when maximum solution quality is paramount.

B. Performances on CPD

TABLE III: Comparative evaluation metrics across the test dataset of Ingraham et al. [46]

Solver / Model	Test dataset		
	NSR (%)	Time (s)	GPU Mem. (MB)
LR-BCD	48.62%	72.88 $\pm$ 98.92	N/A
Loandra	7.78%	60.0 $\pm$ 0.0	N/A
SPB-Maxsat	48.69%	60.0 $\pm$ 0.0	N/A
D2CO	48.54%	0.40 $\pm$ 0.04	119.95 $\pm$ 40.07
D2CO-M	48.45%	0.36 $\pm$ 0.04	119.96 $\pm$ 40.07
D2CO (Top-$K=3$)	50%	0.57 $\pm$ 0.06	264.20 $\pm$ 126.27
D2CO-M (Top-$K=3$)	49.67%	0.54 $\pm$ 0.06	264.24 $\pm$ 126.29
D2CO (Top-$K=5$)	50.40%	0.77 $\pm$ 0.06	397.69 $\pm$ 206.55
D2CO-M (Top-$K=5$)	50.41%	0.73 $\pm$ 0.06	397.76 $\pm$ 206.58

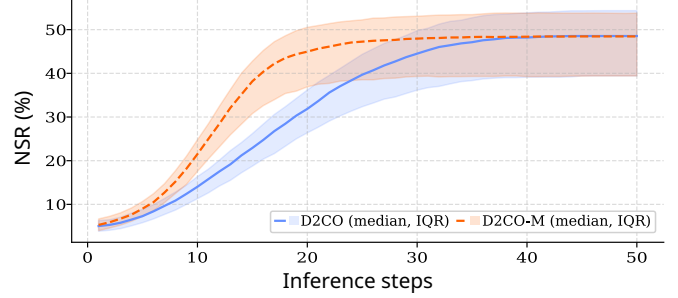


Fig. 2: Median NSR and Interquartile Range (IQR) evaluated on the test set of the Ingraham Dataset [46] during Inference, showing the evolution across discrete inference steps.

Table III summarizes the Native Sequence Recovery (NSR) rates, computational latencies, and GPU memory consumption across the CATH protein structure test dataset. The domain-specific continuous relaxation baseline, LR-BCD, which serves as the training ground truth, achieves a robust NSR of 48.62%, successfully capturing high-quality sequence assignments. However, it incurs a high and highly variable computational cost, averaging nearly 73 seconds per protein instance. When the CFN instances are translated into Partial Weighted Max-2-SAT formulations, the local search solver Loandra struggles severely to navigate the highly rugged, multi-body energy landscapes within the 60-second budget (which is consistently exhausted as evidenced by the 60.0 $\pm$ 0.0 s timing), resulting in a degraded NSR of only 7.78%. Conversely, the advanced state-of-the-art incomplete solver SPB-Maxsat successfully identifies high-quality solutions, achieving an NSR of 48.69% while also exhausting the 60-second time budget.

In stark contrast, our generative diffusion frameworks demonstrate a paradigm shift in resolution latency while maintaining biological fidelity. Both the standard D2CO and our proposed momentum-enhanced variant D2CO-M achieve highly competitive NSRs of 48.54% and 48.45%, respectively, closely matching the training ground truth and baselines. Furthermore, to evaluate the effect of exploring diverse sequence proposals, we performed parallel sampling for the Top- K configurations ($K \in \{3, 5\}$). Introducing Top- K parallel sampling successfully pushes the NSR even higher, reaching

up to 50.40% for D2CO (Top- $K=5$) and 50.41% for D2CO-M (Top- $K=5$).

Most importantly, the neural solvers achieve these near-optimal recoveries in sub-second inference times. Standard D2CO and D2CO-M report inference times of 0.40 and 0.36 seconds, respectively. While scaling up to Top- K parallel sampling slightly increases the computational overhead (reaching 0.77 seconds for $K=5$), the latency remains well under one second while yielding superior recovery performance.

Notably, we explicitly omit a comparative visualization of the NSR evolution over wall-clock time. The massive computational discrepancy between the traditional combinatorial baselines (requiring 60 seconds or more) and the diffusion solvers (converging in fractions of a second) renders a combined temporal axis practically unreadable and uninformative. Instead, we focus on the internal trajectory dynamics of our neural models. The inference times reported in Table III for D2CO and D2CO-M were recorded using a fixed conservative budget of 50 sampling steps to ensure a fair baseline comparison. However, as illustrated in our trajectory analysis (Figure 2)—which reports the median NSR alongside its interquartile range (IQR) to robustly capture performance variance across the test set—Momentum Inference exhibits a dramatically faster convergence rate on CPD tasks, stabilizing at high-quality solutions after only 30 steps. Consequently, the effective inference time required by our approach in practice is substantially lower than the conservative figures reported, further emphasizing the efficiency gains introduced by the momentum mechanism.

VII. CONCLUSION

In this work, we introduced Momentum Inference, a novel mechanism that integrates a second-order moment accumulator into discrete diffusion models for combinatorial optimization. By dynamically pacing the reverse sampling trajectory, our proposed D2CO-M framework effectively mitigates early convergence into local optima. This approach significantly tightens optimality gaps on complex, out-of-distribution instances while preserving high computational efficiency and near-constant sub-second resolution times. Extensive evaluations on Weighted Max-2-SAT and CFN-based Computational Protein Design benchmarks confirm that Momentum Inference delivers superior zero-shot generalization and markedly faster convergence when compared to standard diffusion baselines and traditional combinatorial solvers. This work also shows that when a large collection of real instances can be gathered in a precise domain, as is done here using real protein design or random Max-2-SAT instances, it becomes possible to immediately learn specialized incomplete discrete optimization algorithms that provide state-of-the-art performance, using a generic momentum-accelerated discrete diffusion approach.

REFERENCES

- [1] A. Biere et al., *Handbook of satisfiability*. 2009.
- [2] D. Allouche et al., “Computational protein design as an optimization problem,” *Artificial Intelligence*, vol. 212, 2014.
- [3] K. I. Albanese et al., “Computational protein design,” en, *Nature Reviews Methods Primers*, vol. 5, no. 1, Feb. 2025. DOI: 10.1038/s43586-025-00393-z.
- [4] D. Allouche et al., “Computational protein design as a cost function network optimization problem,” in *Principles and Practice of Constraint Programming*, 2012. DOI: 10.1007/978-3-642-33558-7_60.
- [5] S. Traoré et al., “A new framework for computational protein design through cost function network optimization,” *Bioinformatics*, vol. 29, no. 17, Jul. 2013. DOI: 10.1093/bioinformatics/btt374.
- [6] M. C. Cooper et al., “Soft arc consistency revisited,” *Artif. Intell.*, vol. 174, no. 7-8, 2010.
- [7] Z. Sun et al., “DIFUSCO: graph-based diffusion solvers for combinatorial optimization,” in *Advances in Neural Information Processing Systems 36, NeurIPS 2023*, Dec. 2023.
- [8] Y. Song et al., “Generative Modeling by Estimating Gradients of the Data Distribution,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [9] M. Defresne et al., “A Generative Neuro-Symbolic AI for Protein Sequence Design,” en, *Advanced science*, Aug. 2026. DOI: 10.64898/2026.03.31.715526.
- [10] S. Sanokowski et al., “A diffusion model framework for unsupervised neural combinatorial optimization,” in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, vol. 235, 2024.
- [11] S. Sanokowski et al., “Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics,” in *The 30th International Conference on Learning Representations, ICLR, Apr. 2025*.
- [12] H. Zhao et al., “DISCO: efficient diffusion solver for large-scale combinatorial optimization problems,” *Graph. Model.*, vol. 141, 2025. DOI: 10.1016/J.GMOD.2025.101284.
- [13] Y. Li et al., “Fast T2T: optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization,” in *Advances in Neural Information Processing Systems 37 (NeurIPS)*, Dec. 2024.
- [14] Y. Li et al., “Efficient Few-Step Solution Generation via Discrete Flow Matching for Combinatorial Optimization,” en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 43, Mar. 2026. DOI: 10.1609/aaai.v40i43.41035.
- [15] A. Lucas, “Ising formulations of many np problems,” *Frontiers in physics*, vol. 2, 2014.
- [16] J. Berg et al., “Solver and Benchmark Descriptions,” en,
- [17] Y. Chu et al., “NuWLS: Improving Local Search for (Weighted) Partial MaxSAT by New Weighting Techniques,” en, *Proceedings of the AAAI Conference on*

- Artificial Intelligence*, vol. 37, no. 4, Jun. 2023. DOI: 10.1609/aaai.v37i4.25505.
- [18] J. Zheng et al., “Rethinking the soft conflict pseudo boolean constraint on maxsat local search solvers,” in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, Aug. 2024. DOI: 10.24963/ijcai.2024/220.
 - [19] D. Selsam et al., “Learning a SAT solver from single-bit supervision,” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.
 - [20] E. Yolcu et al., “Learning Local Search Heuristics for Boolean Satisfiability,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
 - [21] V. Kurin et al., “Can q-learning with graph networks learn a generalizable branching heuristic for a SAT solver?” In *Advances in Neural Information Processing Systems 33: NeurIPS*, Dec. 2020.
 - [22] J. Tönshoff et al., “One model, any CSP: graph neural networks as fast global search heuristics for constraint satisfaction,” in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI*, Aug. 2023.
 - [23] H. Duan et al., “Augment with care: Contrastive learning for combinatorial problems,” in *International Conference on Machine Learning ICML*, vol. 162, Jul. 2022.
 - [24] Y. Li et al., “Hardsatgen: Understanding the difficulty of hard SAT formula generation and A strong structure-hardness-aware baseline,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD*, Aug. 2023. DOI: 10.1145/3580305.3599837.
 - [25] Z. Zhang et al., “Grass: Combining graph neural networks with expert knowledge for SAT solver selection,” in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD*, Aug. 2024.
 - [26] A. Hosny et al., “Automatic MILP solver configuration by learning problem similarities,” *Ann. Oper. Res.*, vol. 339, no. 1-2, 2024. DOI: 10.1007/S10479-023-05508-X.
 - [27] D. Simoncini et al., “Guaranteed discrete energy optimization on large protein design problems,” *Journal of chemical theory and computation*, vol. 11, no. 12, 2015.
 - [28] V. Durante et al., “Efficient Low Rank Convex Bounds for Pairwise Discrete Graphical Models,” en, in *Proceedings of the 39th International Conference on Machine Learning*, Jun. 2022.
 - [29] D. Koller et al., *Probabilistic Graphical Models - Principles and Techniques*. 2009.
 - [30] M. C. Cooper et al., “Graphical models: Queries, complexity, algorithms (tutorial),” in *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.
 - [31] J. Dauparas et al., “Robust deep learning-based protein sequence design using ProteinMPNN,” *Science*, vol. 378, no. 6615, Oct. 2022. DOI: 10.1126/science.add2187.
 - [32] Z. Gao et al., *Pifold: Toward effective and efficient protein inverse folding*, 2023.
 - [33] J. Jumper et al., “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, 2021.
 - [34] M. Defresne et al., “Scalable coupling of deep learning with logical reasoning,” in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI*, Aug. 2023.
 - [35] J. Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics,” en, in *Proceedings of the 32nd International Conference on Machine Learning*, Jun. 2015.
 - [36] J. Ho et al., “Denoising Diffusion Probabilistic Models,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020.
 - [37] D. Kingma et al., “Variational Diffusion Models,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021.
 - [38] X. Wang et al., “Boosting diffusion models with an adaptive momentum sampler,” in *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024*, Aug. 2024.
 - [39] J. Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021.
 - [40] E. Hoogetboom et al., “Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021.
 - [41] M. C. Mukkamala et al., “Variants of rmsprop and adagrad with logarithmic regret bounds,” in *Proceedings of the 34th International Conference on Machine Learning, ICML*, vol. 70, Aug. 2017.
 - [42] M. Lauria et al., “CNFgen: A Generator of Crafted Benchmarks,” en, in *Theory and Applications of Satisfiability Testing – SAT 2017*, 2017. DOI: 10.1007/978-3-319-66263-3_30.
 - [43] A. Goerdt, “A Threshold for Unsatisfiability,” *Journal of Computer and System Sciences*, vol. 53, no. 3, Dec. 1996. DOI: 10.1006/jcss.1996.0081.
 - [44] A. Kuegel, “Improved Exact Solver for the Weighted MAX-SAT Problem,” en. DOI: 10.29007/38lm.
 - [45] J. Berg et al., “Loandra in the 2020 MaxSAT evaluation,” *MaxSAT Evaluation*, vol. 2020, 2020.
 - [46] J. Ingraham et al., “Generative Models for Graph-Based Protein Design,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
 - [47] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, 1945.