POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A Generative Algorithm for Topological Design in Facility Layout Planning via Guided Discrete Diffusion

MSc in Mechanical Engineering

Advisor

Prof. Andrea Matta

Author

Damiano Marcon

249191

Academic Year 2024–2025

Abstract

In the landscape of Industry 5.0, the necessity for rapid reconfiguration has turned Facility Layout Planning (FLP) into a strategic driver for operational responsiveness. However, a major bottleneck remains: traditional FLP treats the network topology as a fixed input rather than a dynamic design variable. Because conventional tools rely on local optimizations and rigid templates, they lack the capability to effectively navigate the vast range of network configurations required by today's dynamic production environments.

To bridge this gap, this work introduces a Discrete Denoising Diffusion Probabilistic Model (D3PM) specifically tailored for the synthesis of functional material flow patterns. The proposed approach integrates a Graph Transformer backbone with Classifier-Free Guidance (CFG) to bias the sampling trajectory toward optimal operational regions, aiming to maximize throughput and minimize energy consumption. Furthermore, to ensure structural integrity, the architecture employs a two-stage validation process: a Strict Projector keeps the generation within the feasible design space during inference, while a Logit-Guided Repair mechanism acts as a post-processing step to correct any remaining local or global topological violations. This dual approach ensures functional validity and engineering compliance while preserving generative diversity.

The developed framework is applied across four distinct configuration typologies: Open, Closed, Input-Only, and Output-Only systems. Results demonstrate the model's ability to effectively decouple structural validity from performance, generating novel layouts that outperform unconditional synthesis. Specifically, a detailed analysis of Open Systems reveals a significant performance improvement compared to the baseline. Finally, a targeted study defines the model's sensitivity and operational limits in response to the parameter variations.

Keywords: Facility Layout Planning (FLP), Topological Design, Discrete Diffusion Models, graph generation.

Abstract in Lingua Italiana

Nel paradigma dell'Industria 5.0, la necessità di una rapida riconfigurazione ha reso la Pianificazione del Layout degli Impianti (FLP) un fattore strategico determinante per la reattività operativa. Tuttavia, un limite fondamentale deriva dal considerare la topologia delle connessioni un input statico, anziché una variabile di progetto dinamica. L'affidamento degli strumenti convenzionali a ottimizzazioni locali e modelli rigidi impedisce infatti un'esplorazione efficace dell'ampio spettro di configurazioni di rete richieste dai moderni sistemi di produzione flessibile.

In risposta a tali ciriticità, il presente lavoro introduce un Discrete Denoising Diffusion Probabilistic Model (D3PM), specificamente formulato per la generazione di schemi logici per il trasporto dei materiali. L'approccio proposto integra un'architettura Graph Transformer con la Classifier-Free Guidance (CFG) al fine di orientare la traiettoria di campionamento verso regioni operative ottimali, massimizzando la produttività e minimizzando il consumo energetico. Per garantire l'integrità strutturale, il sistema adotta un processo di validazione a due fasi: uno Strict Projector vincola la generazione allo spazio di progettazione ammissibile durante l'inferenza, mentre un meccanismo di Logit-Guided Repair corregge in post-elaborazione. Tale duplice approccio assicura la validità funzionale e la conformità ingegneristica, preservando al contempo la diversità generativa.

L'architettura sviluppata è stata applicata a quattro distinte tipologie di sistema: Open, Closed, Input-Only e Output-Only. I risultati dimostrano la capacità del modello di scindere efficacemente la validità strutturale dalle prestazioni, generando layout innovativi che superano i risultati della sintesi non condizionata. Nello specifico, un'analisi dettagliata dei sistemi Open rivela un significativo incremento prestazionale rispetto alla generazione randomica. Infine, uno studio di sensibilità ha permesso di caratterizzare i limiti operativi del modello in risposta alle variazioni dei parametri di progetto.

Parole Chiave: Pianificazione del Layout di Impianto, Progettazione Topologica, Modelli di Diffusione Discreti, generazione di grafi.

Contents

Introduction

The transition toward Industry 5.0 is redefining manufacturing paradigms, exposing the limits of rigid, traditional production systems. To compete in a market characterized by high product variety and low volumes, factories must evolve into adaptive environments capable of rapid reconfiguration. Within this landscape, Facility Layout Planning (FLP) assumes a paramount role, establishing the structural foundation required for a truly responsive production environment.

In literature, the FLP is defined as a hierarchical process consisting of two distinct stages: Topological Design, which establishes the logical material flows and functional interactions between resources, and Geometric Design, which handles the physical placement of equipment within Euclidean space. Within this hierarchy, the topological phase serves as the critical "backbone" of the system, determining the fundamental resource paths that dictate overall operational efficiency.

Historically, the interaction between these two stages has been characterized by a significant limitation in scope. When the topological stage was explicitly addressed, it was restricted to combinatorial optimization and fixed heuristics — essentially searching for the best arrangement among a limited set of predefined templates. Conversely, in many industrial practices and academic studies, this stage is often bypassed entirely and treated as a rigid, unoptimized input. In such cases, the focus is confined to the geometric arrangement of machines, under the assumption that the underlying material flow logic is fixed or already optimized.

To overcome this stagnation, this research conducts a systematic investigation into the Topological Design process, aimed at developing a formal methodology that moves beyond traditional static constraints. This approach addresses a recurring bottleneck in layout planning: the tendency to relegate the system's logical backbone to a secondary concern. Such an oversight renders even the most sophisticated geometric optimization incapable of compensating for a fundamentally flawed or rigid structure. Consequently, current methodologies often struggle to navigate the vast landscape of possible network configurations, frequently falling back to established templates or incremental adjustments. However, as the extreme flexibility of modern production environments makes these static approaches insufficient, a paradigm shift toward Generative Design becomes essential. This approach en-

ables the synthesis of entirely new, diverse, and high-performing architectures that traditional heuristics and fixed-input models typically fail to explore.

Focusing on these limitations, this research introduces a Constraint-Guided Discrete Diffusion Model, implemented in Python, specifically designed for the synthesis of industrial graphs. This approach moves beyond traditional heuristics by integrating a PyTorch-based Graph Transformer backbone with a Classifier-Free Guidance (CFG) system, which biases the generative process toward optimal operational regions to maximize throughput and minimize energy consumption. Crucial to this architecture is the decoupling of structural validity from system performance, achieved through a two-stage validation strategy: a Strict Projector constrains the sampling trajectory within the feasible design space during inference, while a Logit-Guided Repair mechanism acts as a post-processing step to resolve any remaining topological violations. By combining this dual-layer protection with generative guidance, the work ensures that the resulting architectures are fundamentally sound and engineering-compliant without sacrificing operational efficiency.

This approach provides a formal framework for bridging the gap between deep generative learning and industrial engineering, providing a robust, automated tool for the design of complex, logical production networks.

The development of this research is organized into several key chapters, each addressing a specific stage of the proposed methodology. The first chapter provides a systematic review of the State of the Art, analyzing the topological design phase and identifying why prior generative models struggle with the rigorous constraints of industrial layouts. Chapter 2 establishes the theoretical foundation through Problem Formulation and System Modeling, defining the graph-theoretic representation of manufacturing systems and the Key Performance Indicators used for evaluation. The core methodology is detailed in Chapter 3, which explains the Guided Discrete Diffusion Framework, covering the training dynamics and the multi-stage generative pipeline. Chapter 4 describes the Experimental Setup, focusing on the computational environment, dataset preparation, and the sensitivity analysis used to calibrate hyperparameters. The Results Analysis in Chapter 5 offers a quantitative assessment of the model's performance across the different system types, supported by ablation studies that isolate the impact of individual architectural components. Finally, Chapter 6 summarizes the main research achievements, discusses current limitations, and outlines future developments for the field.

1

State of the Art

1.1 The Topological Design Phase in Facility Planning

The Facility Layout Planning (FLP) is described in the literature as a hierarchical process composed of two subsequent phases: the logical definition of material flows and functional interactions, and the physical placement of resources within Euclidean space [10]. While the classical optimization literature has focused extensively on the second stage — often modeling it as a Quadratic Assignment Problem (QAP) aimed at minimizing transportation distances — the first one is frequently assumed to be a given, pre-existing input [27, 2].

However, in the context of Greenfield design or Flexible Manufacturing Systems (FMS), the logical structure of the production line is not fixed a priori, but needs to be generated from scratch. This thesis specifically addresses this pre-geometric phase, formally referred to as Material Flow Pattern Design or Topological Layout Synthesis.

1.1.1 Layout as a Graph Synthesis Problem

Before establishing physical coordinates or aisle widths, a layout exists as a purely relational structure [25]. At this stage, the goal is not to minimize travel distances but to establish a valid interconnection logic that satisfies production requirements and topological constraints. Within this framework, the plant is modeled as a directed graph $G(V, E)$, consisting of:

- **Nodes (V)**: represent the active components of the production line, such as machines, storage buffers, and assembly or disassembly stations. Each element is defined by a specific functional role and must comply with engineering rules that govern its valid connections within the system.
- **Edges (E)**: define the logical material flow components. A directed link $u \to v$ depicts a potential transfer from resource u to resource v, focusing on the relational connection rather than the physical distance between them.

This topological abstraction is essential because it establishes the logical backbone of the production system. A layout that is geometrically optimized but topologically invalid — for instance, due to dead ends, incompatible machine interfaces, or broken cycles in closed systems — remains operationally infeasible regardless of its spatial efficiency [31].

1.1.2 The Complexity of Topological Generation

Although removing the geometric dimension may appear to simplify the problem, generating valid flow topologies requires solving a complex Constraint Satisfaction Problem (CSP) over discrete structures. Unlike the physical placement phase, where errors typically translate into suboptimal costs, mistakes at the topological stage result in system infeasibility.

Specifically, a valid flow graph must strictly adhere to engineering constraints at three distinct levels:

1. **Local Process Logic**: each resource has intrinsic I/O limitations based on its function. Processing units, for instance, often operate with a single-input/single-output configuration, whereas assembly stations require converging flows from multiple buffers.
2. **Operational Consistency**: material flow must be continuous and conserved. The system forbids redundant movements, such as transfers between identical storage units, and ensures every path leads to a valid destination to prevent material accumulation.
3. **System Topology (Global Constraints)**: the overall graph structure must reflect the intended production strategy. For instance, closed-loop systems

require the graph to be composed of Strongly Connected Components (SCCs) to ensure recirculation — a property that is difficult to enforce through local rules alone [12].

1.2 Deep Generative Models on Graphs

Following the success of deep learning, in computer vision and Natural Language Processing (NLP), significant research has focused on extending generative models to the graph domain [38]. Unlike images, which are defined on regular Euclidean grids, or texts, which follow a sequential structure, graphs are irregular, non-Euclidean, and inherently combinatorial objects [5]. This complexity introduces unique challenges: rather than simply generating features, graph models must learn to synthesize discrete connectivity patterns — typically represented by an adjacency matrix — while accounting for the vast space of possible permutations and the lack of a fixed node ordering.

Prior to the advent of diffusion models, the field was dominated by two main architectural families: Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN) [13]. Despite their pioneering role, both approaches exhibit limitations that are particularly problematic when applied to the strict engineering constraints of industrial layout synthesis [4].

1.2.1 Variational Autoencoders (VAE) on Graphs

Variational Graph Autoencoders (VGAE) adapt the principles of generative autoencoders to graph-structured data by latent compression and subsequent reconstruction [23]. As illustrated in Figure 1.1, the model encodes the discrete graph structure into a continuous latent space — a simplified numerical representation that distills the identity of categorical nodes and their connections into probabilistic regions. Rather than assigning static coordinates, the encoder defines stochastic distributions that capture the underlying logic of the layout, allowing for coherent variations during the sampling process. Finally, the decoder interprets these latent representations to reconstruct a probabilistic adjacency matrix. In this dense tensor, each entry indicates the likelihood of a connection between nodes, enabling the synthesis of new, structurally consistent topologies without relying on fixed templates.

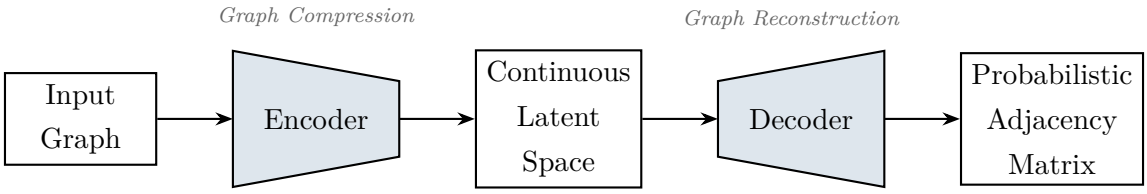


Figure 1.1: Synthetic representation of the VGAE architecture and its functional process.

Training these models relies on an objective function composed of two fundamental terms. The first is the reconstruction loss, which drives the model to maintain high fidelity to the original graph. The second is the Kullback-Leibler (KL) divergence term, which acts as a stabilizer by organizing the latent space into a regular, continuous map. This regularization prevents the model from simply memorizing training data and ensures that nearby points in the latent space correspond to similar physical layouts, which is essential for effective generation.

The versatility of VGAEs has led to their widespread use in bioinformatics for molecular graph generation and drug discovery, as well as in recommendation systems for link prediction and anomaly detection in communication networks. However, despite this success in other domains, VGAEs suffer from two major limitations that hinder their direct application to the synthesis of functional industrial layouts:

- **The Alignment Problem**: computing the reconstruction loss requires establishing a correspondence between the generated graph and the ground-truth. By compressing the entire topological structure into a single global latent vector, the model destroys the direct node-to-node mapping, resulting in a loss of "positional memory". Consequently, the probabilistic framework often forgets the alignment between input and generated nodes. This issue is closely related to graph isomorphism, which is computationally expensive to solve [35].
- **Blurry Generation**: similar to their behavior with images, VAEs tend to produce "averaged" or "blurred" structures. In the context of industrial layouts, this often manifests as overly dense or fully connected graphs that fail to respect the sparsity and specific flow logic of production lines [13].

1.2.2 Generative Adversarial Networks (GAN) on Graphs

Generative Adversarial Networks, exemplified by models such as Molecular Generative Adversarial Network (MolGAN) and Network Generative Adversarial Network (NetGAN), present an alternative paradigm grounded in Game Theory, diverging from the reconstruction-based approach of VAEs [3]. As illustrated in Figure 1.2, GANs frame the generation process as a zero-sum minimax game between two adversarial neural networks. The Generator (G) creates synthetic graphs starting from a latent noise vector, while the Discriminator (D) attempts to distinguish real graphs from the dataset from "fake" ones produced by its counterpart. The training objective is to reach a Nash equilibrium where the Generator produces structures realistic enough to deceive the Discriminator.

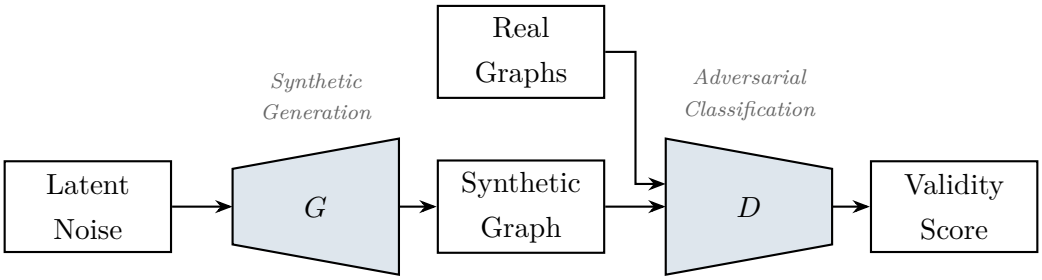


Figure 1.2: Synthetic representation of the GAN architecture and its adversarial process.

The versatility of GANs has made them a cornerstone in fields where high-fidelity generation is paramount. In chemistry, they are employed to design novel molecular structures with specific properties, while in network science, they generate realistic social or biological networks that maintain the statistical properties of original data. In cybersecurity, they are further utilized to simulate synthetic attack patterns to train robust intrusion detection systems. However, despite their ability to generate sharper samples compared to VAEs, applying GANs to discrete graphs — such as industrial layouts — reveals three primary structural limitations:

- **Non-Differentiability**: the continuous domain produced by the Generator are translated in discrete topological structures to be evaluated by the Discriminator. This discretization — often performed from a Categorical or Bernoulli distribution — is mathematically equivalent to a step function, which has a derivative that is zero almost everywhere or undefined. Consequently, during backpropagation, the error gradient calculated by the Discriminator cannot flow backward to the generator's weights, thereby interrupting the learning signal (vanishing gradient) [40].
- **Mode Collapse**: GANs are inherently prone to "collapsing" onto a limited number of configurations (or "modes") that successfully deceive the Discriminator, while disregarding the diversity of the real distribution. In facility planning, where the variety of layout options is a fundamental asset, this inability to generate varied solutions constitutes a critical flaw [13].
- **Validity Issues**: similar to VAEs, GANs typically generate the adjacency matrix in a single step (one-shot generation). It is inherently difficult to enforce strict engineering constraints on a one-shot output without compromising the differentiability of the model.

To overcome the limitations of one-shot generation and the intractability of node alignment, this work adopts Diffusion Models. Unlike previous architectures that attempt to generate a complete layout in a single step, this approach utilizes iterative dynamics to gradually refine the graph structure. By combining native categorical data handling with this step-by-step evolution, the model provides a robust framework for enforcing the syntactic and semantic validity required by industrial systems.

1.3 Diffusion Models

Diffusion Models are a class of generative architectures designed to learn the reversal of a gradual data corruption process [18]. As illustrated in Figure 1.3, in the forward direction, a structured sample is progressively degraded through a Markov chain $q(x_t|x_{t-1})$ until it converges to a simple reference distribution, such as Gaussian noise. Training focuses on approximating the reverse transitions $p_\theta(x_{t-1}|x_t)$, which allows the model to recover the original data distribution through sequential denoising. During inference, the generation begins by sampling from random noise and iteratively applying the learned reverse kernels — i.e., the step-by-step rule according to which noise is injected — to reconstruct a realistic sample.

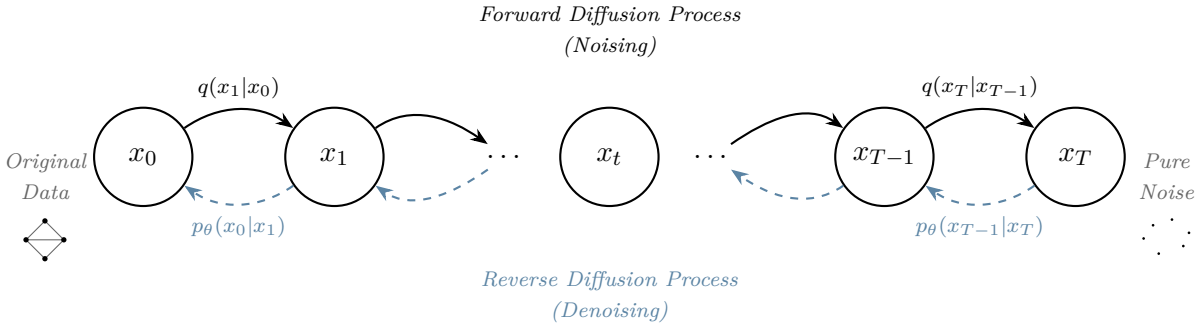


Figure 1.3: Schematic representation of the forward and reverse processes in Diffusion Models.

Although Diffusion Models were originally developed for continuous signals like images and audio, their extension to the graph domain has emerged as a significant area of research. However, the transition from pixels to graphs fundamentally alters the nature of the problem, as graphs are inherently discrete and combinatorial. Unlike the continuous domain, small local perturbations in a graph — such as adding or removing a single edge — can drastically impact global properties, including connectivity, cyclicity, and routing feasibility. Furthermore, industrial layouts require the satisfaction of strict engineering constraints that local consistency alone cannot enforce. Consequently, diffusion-based graph generation requires a careful balance between structural validity, controllability, and computational efficiency.

From a modeling standpoint, a critical design decision concerns the domain over which the diffusion Markov chain is defined. Approaches based on continuous score matching embed graphs into a Euclidean space to apply Gaussian perturbations,

leveraging established methodologies for continuous data. In contrast, Discrete (Categorical) Diffusion defines the forward kernel directly over discrete random variables. This strategy is particularly advantageous for facility planning, as it preserves the syntactic nature of the categorical data at every intermediate step of the generation process, providing a more natural framework for enforcing the strict rules of a production system.

1.3.1 Continuous Score-Based Diffusion.

By adapting techniques originally developed for image or audio synthesis, the forward noising process is generalized to Euclidean space [20]. This approach, commonly referred to as Continuous Score-Based Diffusion, requires graphs to be encoded as vectors and tensors within $\mathbb{R}^d$ — typically through adjacency matrices and node/edge attribute tensors. In this framework, a stochastic forward process progressively injects Gaussian noise into the data, while a neural score network is trained to approximate the score function: the gradient of the log-density $\nabla_x \log q(x_t)$.

This gradient acts as a geometric guide: rather than explicitly modeling the complex probability distribution of all possible topologies, the network learns to estimate the direction in which a noisy graph must be perturbed to increase its likelihood. In practice, the score function defines a vector field that points toward regions of high data density, effectively providing a "pathway" from random noise back to structured layouts. During the generation phase, the model follows this gradient to iteratively denoise an initial Gaussian sample, reconstructing a coherent graph representation through learned reverse-time dynamics.

Despite its success in other domains, this continuous approach faces significant limitations when applied to discrete industrial layouts. Because the diffusion trajectory evolves within a continuous space, the model often departs from the underlying discrete domain during the denoising process [37]. This transition frequently produces artifacts, such as fractional adjacency entries or invalid one-hot label encodings, which do not represent real physical components. To address this, researchers often rely on projection or discretization heuristics to map intermediate continuous states back onto valid graph structures [20]. However, these corrections can introduce errors and may not guarantee that the final layout is operationally feasible [15].

Furthermore, accurate sampling through continuous methods typically requires

hundreds of reverse steps. This iterative demand significantly compromises scalability, especially when dealing with industrial systems that may contain thousands of vertices. These obstacles — namely the loss of discrete integrity and the high computational cost — motivate the shift toward discrete diffusion strategies that can more naturally handle the categorical nature of production line components.

1.3.2 Discrete (Categorical) Diffusion.

To prevent the model from deviating from the discrete graph domain during denoising, researchers often adopt Discrete Diffusion [1]. Instead of using continuous perturbations, this approach defines the forward process directly over categorical variables, such as node labels in $\{0, \dots, K-1\}^N$ and edges in $\{0,1\}^{N \times N}$. In this framework, the noise kernel is formulated as a Markov corruption process where the data is gradually degraded through discrete operations, such as random node reassignment or Bernoulli edge flipping.

A primary advantage of this formulation is that the generative model predicts logits — raw, unnormalized scores representing the model's confidence in each possible category — within the original discrete state space of the variables. Consequently, every intermediate step of the reverse denoising process yields a categorical representation rather than a continuous approximation. This effectively avoids the fractional values and encoding errors that characterize continuous formulations, ensuring that the data remains within its original domain throughout the generation.

Two pivotal methodological advancements have defined the State of the Art in this domain: GraphDDPM and DiGress. While both share the same categorical diffusion kernel, they represent distinct stages in the evolution of denoising architectures:

GraphDDPM pioneered the application of categorical diffusion to graphs, ensuring 100% formal validity by strictly enforcing the discrete nature of the data. Using a Multi-Layer Perceptron (MLP) as the denoiser, GraphDDPM achieved significant efficiency gains over earlier Langevin dynamics approaches, particularly in small-molecule benchmarks [15].

DiGress represents the subsequent evolution of this paradigm. While retaining the validity of the GraphDDPM kernel, DiGress replaced the MLP with a Graph Transformer. This architectural shift allowed for shared attention mechanisms

across nodes and edges, enabling the model to capture complex structural dependencies. Consequently, DiGress reduced the number of required sampling steps by an order of magnitude compared to its predecessors [37].

However, despite these advances in syntactic validity, both methods suffer from a shared limitation derived from the strictly local nature of the discrete noise. Neither the MLP nor the Transformer contains intrinsic mechanisms to enforce global semantic consistency. Without explicit guidance, these models struggle to satisfy long-range constraints — such as material flow balancing or the prevention of invalid cycles — that are essential for a functional plant topology. This gap between local validity and global feasibility provides the primary motivation for the Constraint-Guided approach developed in this work.

1.3.3 Constrained and Guided Generation

While Discrete Diffusion successfully ensures syntactic validity — guaranteeing that every generated graph is technically well-formed — there remains a critical gap between validity and feasibility. In complex domains such as industrial layout generation, a graph that possesses structural integrity is of limited value if it fails to satisfy rigorous global system requirements. To bridge this gap, the State of the Art has evolved from monolithic Diffusion Models toward modular, composite architectures. These frameworks decouple the task of generating structure from the tasks of semantic conditioning and topological compliance by introducing specialized components that operate across three distinct but complementary directions:

- **Intent-Driven Conditioning: Classifier-Free Guidance (CFG)**
  A fundamental requirement for industrial layout synthesis is the ability to steer the generative process toward specific design objectives. Standard unconditional models are designed to estimate the general data distribution $p(x)$, essentially producing "random" valid layouts. However, engineering applications require sampling from a conditional distribution $p(x|c)$, where the generation is governed by a specific context c. This context is not restricted to simple category labels; it can encode complex design intents, such as specific production volumes or continuous performance benchmarks

Early methods, such as Classifier Guidance, achieved this level of control by utilizing an external auxiliary model (a classifier) $p(c|x_t)$ to act as a "guide" during the generation process [9]. This separate model serves as an external critic that evaluates noisy, intermediate samples at each step of the denoising process. By processing the incomplete graph x_t, the classifier calculates a gradient that dictates the specific topological adjustments needed to increase the probability of matching the target condition c. As a result, this external classifier effectively "pushes" the diffusion process toward the desired design intent at each step of the denoising trajectory.

Despite its effectiveness in continuous domains like image generation, this approach introduces significant challenges for industrial graph synthesis. First, it complicates the training pipeline by requiring a separate, high-performance classifier specifically capable of recognizing patterns in noisy data. Second, it incurs heavy computational overhead, as the external model must be queried at every iteration of the reverse process. Most importantly, the reliance on gradients is mathematically problematic for discrete graph structures. Since changes in a production line's topology (such as adding an edge or changing a node type) are discrete rather than continuous, there is no "smooth slope" for the classifier's gradient to follow. This makes it difficult to reliably "steer" the generation toward valid engineering outcomes.

To resolve these limitations, CFG offers a more robust alternative [19]. This paradigm eliminates the need for an external critic by training a single diffusion model to act as both a conditional and an unconditional generator simultaneously. During the training phase, the conditioning information c is randomly discarded and replaced with a "null token" $\emptyset$ with a fixed probability p_{uncond}. This stochastic masking forces the model to learn both the conditional prediction error $\epsilon_\theta(x_t, c)$ and the unconditional prediction error $\epsilon_\theta(x_t, \emptyset)$ simultaneously within the same set of weights, effectively internalizing the guidance mechanism.

At inference time, the combination of these two estimates allows for the definition of the update direction $\tilde{\epsilon}_\theta(z_t, c)$ as a weighted balance between the conditional and unconditional models:

$$\tilde{\epsilon}_\theta(x_t, c) = \epsilon_\theta(x_t, \emptyset) + \gamma \cdot [\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t, \emptyset)]$$

The term in brackets represents the "semantic direction" — the features that distinguish the conditioned layout from a generic one. Scaling this difference by the Guidance Scale γ allows for a sharper emphasis on the design intent. A higher γ ensures stricter adherence to performance targets while managing the variety of the generated solutions.

- **Topology-Aware Control: Guidance and Projection**
  To address the challenge of global semantics, the denoising process must ensure adherence to specific topological objectives, such as acyclicity, target degree distributions, or manufacturing flow-balance constraints. Topology-aware methods intervene directly at each step of the reverse generation process, introducing either an auxiliary guiding signal or a corrective operator to shape the trajectory [37]. Crucially, these approaches operate on the sampling path, uncoupling the control mechanism from the forward diffusion kernel. Within this framework Endpoint Guidance and Constraint Projection arose as dominant in the State of the Art.

  Endpoint Guidance leverages an auxiliary network to predict a coarse target, such as a desired degree sequence or the number of connected components. Throughout the generation process, the reverse model's logits are progressively steered toward this objective, promoting convergence to the prescribed topology in a substantially smaller number of iterations compared to an unguided Markov chain [7].
  Constraint Projection follows the logic of Network Diffusion (NetDiff) by applying a dedicated operator after each denoising update to map the current intermediate sample back onto the set of valid configurations. During the generation process, the model's raw mathematical updates may produce "illegal" states — configurations that violate fundamental rules, such as overlapping machines or broken material flows. The projection operator acts as an immediate corrector: it identifies these invalid intermediate samples and "re-projects" them to the nearest valid configuration within the allowed design space [29].
  In multi-level or hierarchical architectures, this mechanism is essential for maintaining consistency across different scales of the layout. It ensures that high-level topological decisions always correspond to valid low-level structures.

By correcting violations immediately at each time step, this method prevents the accumulation of small errors — which would otherwise result in a completely non-functional final layout — ensuring the feasibility of the trajectory without relying on extensive post-generation refinement.

While both strategies integrate easily with discrete categorical kernels, they operate through fundamentally different mechanisms. Guidance functions by injecting a learnable, probabilistic bias that steers the generation process, whereas projection enforces constraints via a deterministic "hard" mapping. This distinction primarily manifests in a significant computational trade-off between the training and inference phases. On one hand, guidance necessitates an auxiliary neural module and careful hyperparameter tuning, effectively shifting the complexity and the resource burden to the training phase. On the other hand, projection is typically parameter-free and requires no additional training, but it introduces a substantial cost during inference. This is because projection often requires the resolution of a complex Constraint Satisfaction Problem — which can be potentially NP-hard — at every single iteration of the denoising process. Consequently, modern latent and hierarchical architectures frequently coordinate these two mechanisms to optimize the balance between strict controllability and overall computational efficiency [7] [29].

- **Scalability: Latent Diffusion**
  When processing graphs with high-dimensional attributes or extensive topologies, executing the diffusion process in the explicit discrete space can become prohibitive regarding memory consumption and sampling runtime. Latent diffusion addresses this issue by decoupling the data representation from the generative process. An encoder maps the input graph into a compressed lower-dimensional latent code, upon which the diffusion and denoising operations are performed; subsequently, a decoder reconstructs the explicit graph structure from the denoised latent representation.
  A concrete realization of this paradigm is Latent Graph Diffusion (LGD), which compresses topology and nodal attributes into a continuous bottleneck achieving speedups of one to two orders of magnitude compared to explicit-

space formulations. Furthermore, the learned latent representation provides a compact, semantic-rich embedding that can be directly leveraged by downstream predictive heads, facilitating efficient multi-task learning without the need to re-process the raw graph structure. The primary trade-off concerns fidelity and information loss: if the encoder discards rare patterns — such as infrequent node types or subtle connectivity details — the decoder may fail to recover them, potentially compromising the validity of the final output. To mitigate this risk, State of the Art hierarchical models often incorporate "discrete bridges" or auxiliary reconstruction losses, ensuring that critical semantic details are preserved alongside the compressed latent features [6].

In industrial layout generation, semantic control and topological integrity are inseparable requirements. While general graph synthesis focuses on structural validity, production plant design demands strict alignment with performance targets and the continuous satisfaction of domain-specific constraints. Although Latent Diffusion is frequently proposed for scalability, its inherent compression risks discarding subtle connectivity details — such as the precise mapping between machines and buffers — that are essential for operational functionality.

Consequently, this thesis adopts an Explicit Diffusion framework to prioritize structural fidelity over latent compression. The developed approach integrates internalized CFG for performance-based steering with a dual-layer control system (Strict Projection and Logit-Guided Repair). By operating directly on the graph domain and enforcing hard industrial constraints at every denoising step, this methodology ensures that each generated component maintains its physical significance and operational role.

2

Problem Formulation and System Modeling

The development of a robust graph generation framework, capable of adhering to stringent topological and semantic constraints, requires a rigorous modeling approach. The foundation of the adopted algorithm lies in the graph-theoretic representation of manufacturing layouts and their functional classification. This taxonomy is crucial for defining metrics that serve both to analyze observed configuration and to formulate the objective function required for model guidance. Together, these structural and evaluative components define the operational space within which the diffusion-based generative model and the associated multi-stage methodology function.

2.1 Graph-Based Layout Representation

The industrial layout generation problem is formulated as a directed graph generation task, wherein the graph topology encodes the material flow structure among the various processing stations. Unlike standard graph problems where vertices are often treated as homogeneous entities, an industrial environment requires a rigorous distinction between components to satisfy domain-specific constraints. A processing unit, for instance, implies different connectivity rules compared to a storage one. To capture this intrinsic functional heterogeneity and enable the model to learn context-aware placement rules, these resources are categorized into four distinct functional types, defined as:

$$\mathcal{C} = \{\text{Machine}, \text{Buffer}, \text{Assembly}, \text{Disassembly}\}$$

To render this semantic information processable by the generative model, a bijec-

tive mapping is applied to translate the categorical labels into the discrete numerical space $f : \mathcal{C} \to \{0, 1, 2, 3\}$, associating each functional type with specific topological requirements:

- **Machine** (0): represents standard processing unit. It requires exactly one incoming flow and one outgoing flow, both connected to buffers.
- **Buffer** (1): is an intermediate storage station. It is essential for process decoupling and for managing variability in material flows.
- **Assembly** (2): merges multiple flows. It requires two incoming flows and produces a single outgoing flow.
- **Disassembly** (3): splits a flow into multiple branches. It requires one incoming flow and produces two outgoing flows.

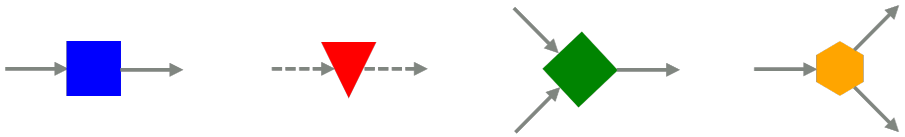


Figure 2.1: Visual classification of the industrial processing nodes. From left to right: Machine (0), Buffer (1), Assembly (2), and Disassembly (3).

Consequently, for a system with N total nodes, the layout is formally represented as a directed graph $G(V, E)$, where $V \in \{0, 1, 2, 3\}^N$ denotes the node feature vector containing the encoded functional types, and $E \in \{0, 1\}^{N \times N}$ represents the binary adjacency matrix encoding the logistical connections among them.

Furthermore, the definition of the adjacency matrix E is subject to strict industrial logic constraints that distinguish valid layouts from infeasible ones. Specifically, the following topological rules are enforced to ensure the physical consistency of the material flow:

- **No Self-loops**: for any node i, the diagonal elements of the adjacency matrix must be zero ($E_{ii} = 0$), as a resource cannot discharge material into itself.
- **Buffer-Mediated Flow**: direct connections between two storage units are prohibited. Every material transfer must be mediated by a processing resource, meaning that if V_i = Buffer and V_j = Buffer, then $E_{ij} = 0$.
- **Mandatory Buffer Interface**: to ensure process decoupling, all processing units (Machine, Assembly, and Disassembly) must strictly receive input from, and send output to, nodes of type Buffer.

These constraints define the "legal" search space for the generative model and serve as the basis for the Strict Projector and Repairer mechanisms discussed in the following chapters.

2.2 System Topology Classification

The development of the industrial graph generation framework is predicated on a rigorous characterization of production systems, ensuring that the generative process is grounded in functional manufacturing logic.

While such systems can be classified according to numerous criteria, this work focuses specifically on the interaction established between the plant and the external environment. This interaction is formally defined through the identification and topological analysis of input sources and output sinks.

Let $G(V, E)$ be the representative graph of the layout and $V_B \subset V$ the subset of Buffer nodes. The in-degree $d_{in}(i)$ and the out-degree $d_{out}(i)$ of a node i are defined as the number of incoming and outgoing edges internal to the system, respectively. Consequently, the sets of the input source nodes S_{in} and output sink nodes S_{out} have been identified as follows:

- **Input Sources** (S_{in}): are buffers acting as entry points for raw materials in the system. Topologically, they are defined as Buffer-type nodes without any entering edge from other nodes of the graph:

$$S_{in} = \{i \in V_B | d_{in}(i) = 0\}$$

- **Output Sink** (S_{out}): are the buffers that collect the finished products destinated to leave the system. Topologically, they are defined as Buffer-type

nodes without any exiting edge to other nodes of the graph:

$$S_{out} = \{i \in V_B | d_{out}(i) = 0\}$$

Based on the cardinality of these boundary sets ($\mathcal{S}_{in}$ and $\mathcal{S}_{out}$), and specifically by evaluating the presence or absence of external interfaces, industrial layouts are classified into four distinct system topologies:

- **Open System**
  An Open System represents the standard configuration of a linear production line where raw materials are continuously converted into finished products. The system operates as a production unit, receiving inputs from an upstream source and discharging processed outputs to the downstream environment.

  $$|\mathcal{S}_{in}| \geq 1 \wedge |\mathcal{S}_{out}| \geq 1$$

  From the semantic standpoint, the material flow is directional and acyclic, crossing the graph from a set of sources nodes to a set of sink nodes.
- **Closed System**
  A Closed System is typical of continuous recirculation processes or operations on circular pallet systems. It is an insulated configuration, characterized by the complete absence of exchanges with the external environment.

  $$|\mathcal{S}_{in}| = 0 \wedge |\mathcal{S}_{out}| = 0$$

  No buffer serves as a boundary interface (input or output). From a topological perspective, this implies that the graph must be strongly connected (or composed of strongly connected components), ensuring that every entity within the system can circulate indefinitely without accumulating in dead-end states.
- **Input-Only System**
  An Input-Only System models storage or filling processes (e.g., a warehousing or loading operations), where material flows into the system but is not processed for output within the considered time horizon.

  $$|\mathcal{S}_{in}| \geq 1 \wedge |\mathcal{S}_{out}| = 0$$

  The system behaves as a global accumulator (sink), continuously absorbing

material without releasing it externally.

- **Output-Only System**
  An Output-Only System models emptying or order fulfillment processes, in which an initial inventory distributed across buffers is processed and discharged to the surrounding environment without any new incoming material.

$$|\mathcal{S}_{in}| \geq 1 \wedge |\mathcal{S}_{out}| = 0$$

  The system operates then in a depletion mode, evolving from a saturated state toward and empty one.

This taxonomy is crucial to the generative framework, as each topological class imposes unique structural constraints on graph connectivity (e.g., cyclicity versus linearity) and dictates the appropriate methodology for performance assessments. Consequently, this classification serves as the necessary theoretical basis for the Key Performance Indicators and the Optimization Objective defined in the following sections.

To bridge the gap between these theoretical definitions and the practical generation of industrial layouts, the framework translates each topological class into a set of formal operational requirements. These are enforced through the integration of the Strict Projector during the denoising phase and a final Logit-Guided Repair step.

Beyond boundary cardinalities, the framework ensures the global integrity of the four system types by asserting specific connectivity properties:

- **Open Systems Enforcement**: in Open systems a verification algorithm ensures that every node in the graph is reachable from the set $\mathcal{S}_{in}$ via a directed path, preventing the formation of isolated functional "islands" that cannot contribute to the system's throughput.
- **Closed Systems Enforcement**: the insulation of these systems requires a strictly enforced strong connectivity. The framework verifies that the graph consists of a single Strongly Connected Component (SCC), ensuring that every entity can circulate indefinitely.
- **Input/Output-Only Enforcement**: these semi-bounded configurations are validated as directional flow accumulators or depleters. The generative logic

ensures that for Input-Only systems, all material trajectories terminate within internal buffers, while for Output-Only systems, every internal resource has a valid path to the discharge sinks in $\mathcal{S}_{out}$.

Furthermore, every generated configuration must comply with the topological constraints defined in the layout representation phase (Section 2.1), which govern the admissible connections and functional relations between resources. These include the categorical prohibition of direct buffer-to-buffer edges and the strict adherence to node-specific I/O cardinalities. By embedding these requirements directly into the generative pipeline, the framework ensures that the resulting layouts are not only diverse but also strictly compliant with the industrial logic of the selected system type.

2.3 Key Performance Indicators (KPIs)

To quantitatively asses the quality of the observed industrial layouts, two competing Key Performance Indicators have been defined: System Throughput (X) and Total Energy Consumption (E).

These metrics are computed via a Discrete Event Simulation (DES) module over a fixed time period T_{sim}. While the energy metric is universally defined regardless of the graph structure, the throughput formulation is strictly dependent on the topological class of the system presented in Section 2.2.

2.3.1 System Throughput

The definition of System Throughput varies strictly according to the system topology, as the operational objective shifts from production flow to storage capacity or internal processing intensity. The specific formulations for each system type are defined as follows:

- **Open System (Production Rate)**
  In Open Systems, characterized by the linear flow of material from input sources to output sinks, Throughput is measured as the Production Rate. It quantifies the system's ability to process raw materials releasing finished products. It is computed as the total count of items that successfully exit the system through an output sink ($\mathcal{S}_{out}$), divided by the simulation horizon

(T_{sim}):

$$X_{open} = \frac{\sum_{v \in \mathcal{S}_{out}} n_v(T_{sim})}{T_{sim}}$$

where $n_v(T_{sim})$ is the cumulative count of parts collected by sink v up to the simulation time T_{sim}.

- **Closed System (Average Machine Activity)**
  Closed Systems behaves as recirculating loops with a fixed number of circulating elements and no interaction with the external environment. Consequently, Throughput reflects the internal processing intensity. It is computed as the aggregate number of operations completed by all processing stations, normalized by the number of active processing nodes (N_{act}) and the simulation duration:

$$X_{closed} = \frac{\sum_{v \in V \setminus V_B} ops_v(T_{sim})}{N_{act} \cdot T_{sim}}$$

  where ops_v is the total number of operations completed by node v at time T_{sim}.
- **Input-Only System (Filling Rate)**
  Input-Only Systems operate as absorption networks, capturing material from sources without discharging it. The Throughput is defined as the Filling Rate, measuring the system capability to store incoming flow. It is computed as the total count of items accumulated inside all internal buffers divided by the effective time (T_{eff}) required to reach the saturation state (or T_{sim} if saturation is not reached):

$$X_{in} = \frac{\sum_{v \in V_B} n_v(T_{eff})}{T_{eff}}$$

- **Output-Only System (Emptying Rate)**
  Output-Only Systems model depletion processes where an initial inventory is discharged without replenishment. Throughput is formulated as the Emptying Rate, representing the speed of inventory discharge. It is calculated as the net reduction in inventory divided by the effective depletion time:

$$X_{out} = \frac{C_{tot} - \sum_{v \in V_B} n_v(T_eff)}{T_{eff}}$$

  where C_{tot} denotes the total initial inventory of the system at time $t = 0$.

2.3.2 Total Energy Consumption

Unlike Throughput, the Total Energy Consumption metric is invariant to the system topology classification. It is defined as the aggregate energy expenditure of the entire manufacturing plant over the simulated time horizon. The total consumption is modeled as the summation of the energy absorbed by the subset of active nodes $v \in V \setminus V_B$ (Machines, Assembly, Disassembly) based on their specific operational states:

$$E = \sum_{t=0}^{T_{sim}} \sum_{v \in V \setminus V_B} P_v(t) \cdot \Delta t$$

where $P_v(\cdot)$ represents the state-dependent power consumption rate for node v. This function assigns a specific energy cost based on the resource's functional type and its instantaneous status, differentiating between the high energy demand of active processing and the lower baseline consumption of idle states. Minimizing this metric encourages the generation of efficient layouts that reduce idle times and blocking/starving phenomena.

2.4 Optimization Objective

The objective of the proposed framework extends beyond the mere generation of topologically valid graphs, aiming to steer the exploration toward layouts that are both high-performing and energy efficient. To this end, a scalar score function S is introduced, to balance two competing objectives: the maximization of the productivity and the minimization of the energy consumption.

The score function is defined as linear combination of normalized metrics:

$$S = w_X \cdot X_{norm} - w_E \cdot E_{norm}$$

where the weights w_X and w_E can be tuned to reflect the design preferences. Since the considered metrics are heterogeneous physical quantities characterized by different unit of measure and order of magnitude, a rigorous normalization procedure is required to make them comparable within a single objective function. This standardization process is articulated into two phases:

- **Per-Node Scaling**: to avoid the distortion due to the graph dimension, the absolute metrics are firstly normalized according to the number of active processing resources. The per-node metrics are computed as:

$$x_{node} = \frac{X}{N_{act}}, \qquad e_{node} = \frac{E}{N_{act}}$$

- **Population Standardization**: subsequently, a statistical standardization (Z-score) is applied to center the metrics around the distribution of the current generated population. This step converts the per-node values into dimensionless quantities representing the deviation from the population mean:

$$X_{norm} = \frac{x_{node} - \mu_X}{\sigma_X}, \qquad E_{norm} = \frac{e_{node} - \mu_E}{\sigma_E}$$

  Where μ and σ represent respectively the mean and the standard deviation of the per-node metrics computed within the reference dataset.

3

Methodology: Guided Discrete Diffusion Framework

Diffusion Models are generative models designed to learn how to reverse a gradual data corruption process [18]. In the forward direction, a structured sample is progressively degraded through a Markov process $p(x_t|x_{t-1})$ until it becomes simple Gaussian noise. Training focuses on approximating the reverse transitions $p_\theta(x_{t-1}|x_t)$, allowing the model to recover the original data distribution through sequential denoising. During the inference phase, the model generates a realistic sample by starting from random noise and iteratively applying a learned reverse kernel, which is the step-by-step rule used to reconstruct the data.

Building on these principles, the methodology in this work is structured into two primary pillars: the Diffusion Training Framework and the Generative Pipeline.

The first phase, the Training Framework, focuses on optimizing the model's internal weights. This is achieved by coupling the forward noising process with a continuous evaluation of the model's predictions. The discrepancy between these outputs and the ground-truth data is used to update the architecture's parameters via backpropagation.

Complementing this learning phase, the Generative Pipeline translates the acquired logic into concrete industrial layouts during inference. Through the reverse denoising process, these elements operate in synergy to form a cohesive architecture, capable of interpreting the complex structural rules of manufacturing plants and synthesizing optimized configurations tailored to specific user requirements.

3.1 The Diffusion Training Framework

The Diffusion Training Framework defines the methodological core of the generative agent, formalized as a Discrete Denoising Diffusion Probabilistic Model (D3PM). This approach shifts the design of industrial layouts from a traditional deterministic optimization task, to a probabilistic inverse problem: the model learns to reconstruct valid, high-performing plant configurations by iteratively removing noise from a state of pure entropy.

To ensure the agent accurately internalizes the distinct structural rules of different manufacturing paradigms, the framework is implemented through independent training sessions for each system type. This specialization is fundamental to prevent "architectural confusion" within the network's weights; since the topological constraints of different configurations are often mutually exclusive, a single global model would struggle to satisfy divergent requirements simultaneously. For instance, the gradients required to enforce the recirculating flow characteristic of a Closed System directly contradict the mathematical signals needed to establish global entry and exit points in an Open System. By developing dedicated architectural weights for Open, Closed, Input-Only, and Output-Only configurations, the framework eliminates interference between these divergent constraints, ensuring that each model instance is perfectly aligned with the specific operational logic and flow dynamics of its target domain.

To manage these specialized learning processes effectively, the framework adopts a hierarchical structure organized into Epochs, Batches, and Items, as illustrated in Figure 3.1. This division is fundamental to the optimization logic, as it balances computational efficiency with the stability required to learn complex structural patterns.

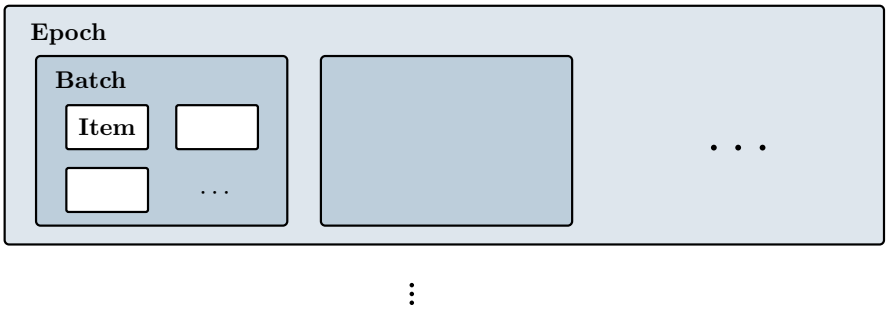


Figure 3.1: Diffusion Training Framework: Item, Batch, and Epoch hierarchy.

The analysis unfolds across these three levels to ensure that every update is grounded in solid data. At the Item level, the model processes individual graphs to calculate specific loss signals. However, because single samples may contain noise or outliers, the Batch level acts as a statistical mediator. By aggregating and averaging the results of multiple items, the Batch level smooths out individual irregularities, providing a more stable and reliable gradient computation for the optimization phase. Finally, the Epoch level represents a complete pass through the entire dataset, serving as a fundamental multiplier for the overall learning process. At the beginning of each new cycle, the dataset is reshuffled to prevent the model from memorizing fixed sequences, a stochastic approach that promotes better generalization and ensures the network internalizes the underlying structural logic rather than specific data orderings. Under this repetitive regime, the system progressively refines its internal weights, with the total number of parameter updates ultimately determined by the number of epochs multiplied by the number of batches within each.

To maintain consistency, the Performance Scores S undergo a preliminary preprocessing phase where Min-Max normalization maps them into a controlled $[0, 1]$ range. This results in a uniform scale defined as the Conditioning Score c, ensuring a standardized input space for the model. Once scaled, the data proceeds through the six fundamental stages of the training pipeline, as illustrated in the flowchart of Figure 3.2.

Guided by the aforementioned hierarchy, the process at the individual sample level follows a precise scheme: the Forward Corruption Process first transforms the original layouts into noisy states while, in parallel, Conditioning Dropout is applied to the performance targets. The framework then integrates these signals through an Early Fusion Strategy, creating a unified representation that feeds into the Reverse Learning Process. This stage represents the core of the training, where the model is taught to reconstruct the original structure from the corrupted input. Finally, these interactions are synthesized into a stable learning signal via a Composite Loss Function and a dedicated Optimization Strategy, ensuring the model produces layouts that are both structurally valid and functionally efficient.

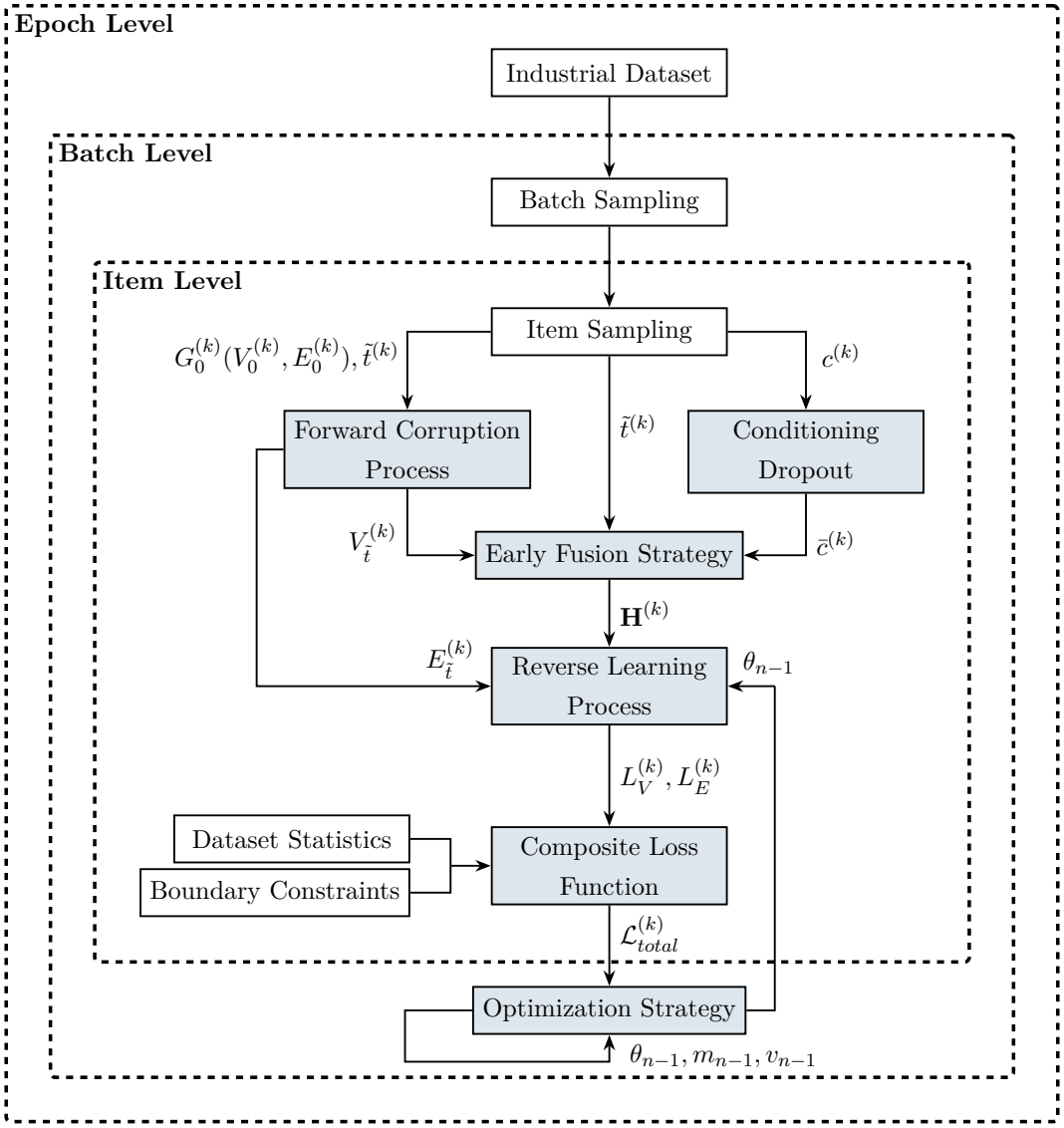


Figure 3.2: Procedural architecture of the Diffusion Training Framework.

3.1.1 Forward Corruption Process

The training framework starts with the Forward Corruption Process, which acts as the "entropic" phase of the generative pipeline. Since industrial layouts are defined by discrete elements — machine types and binary connections — this process acts as a stochastic state transition, gradually turning a functional plant into random noise.

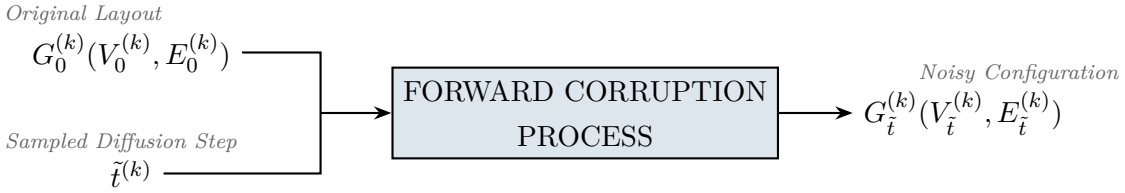


Figure 3.3: Input-output architecture of the Forward Corruption block as integrated within the Diffusion Training Framework.

As illustrated in Figure 3.3, the Forward Corruption Process requires two inputs: a ground-truth layout $G_0^{(k)}(V_0^{(k)}, E_0^{(k)})$ and a diffusion timestep $\tilde{t}^{(k)}$ — where the superscript (k) identifies the current Item index — sampled uniformly across the Diffusion Horizon ($\tilde{t}^{(k)} \sim \mathcal{U}(0, T)$). This timestep acts as a noise controller: as $\tilde{t}^{(k)}$ approaches T, the level of corruption increases. Driven by a Markov chain, the framework uses this sampled value to degrade the graph's structural integrity, yielding the corrupted configuration $G_{\tilde{t}}^{(k)}(V_{\tilde{t}}^{(k)}, E_{\tilde{t}}^{(k)})$ as its direct output.

For notational clarity, from this point onward, the mathematical formulation describes the operations applied to the single graph sample k, omitting the Item index to streamline the equations.

The dynamics of the corruption process, as proposed by Nichol et al. [32], are governed by a Cosine Noise Schedule which mitigates the rapid degradation of structural information typical of standard linear schedules. Within this discrete framework, the noise injection is parameterized by β_t, representing the marginal probability of corruption at the generic timestep t.

To achieve a gradual and controlled degradation up to the sampled $\tilde{t}$, the gov-

erning equation directly defines the cumulative signal retention, $\alpha_{\tilde{t}}$, as:

$$\alpha_{\tilde{t}} = \frac{f(\tilde{t})}{f(0)}, \quad \text{where} \quad f(\tilde{t}) = \cos\left(\frac{\tilde{t}/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

where $s > 0$ is a small offset introduced to prevent the noise level from being excessively high at $t = 0$.

This retention parameter effectively quantifies the "memory" of the original layout over time; specifically, the term $\alpha_{\tilde{t}}$ represents the probability that a single component remains unchanged from the start up to the extracted step $\tilde{t}$. While the total corruption probability is given by $1 - \alpha_{\tilde{t}}$, the marginal corruption one β_t — which defines the noise injected at the generic timestep t — is derived from the posterior relation:

$$\beta_t = \min\left(1 - \frac{\alpha_{\tilde{t}}}{\alpha_{\tilde{t}-1}}, 0.999\right)$$

As $\tilde{t}$ approaches T, the value of $\alpha_{\tilde{t}}$ decays to zero following a distinctive "S-shaped" curve. As demonstrated by Vignac et al. [37], this characteristic ensures the imposition of a cautious action on the corruption process: noise is introduced gradually in the initial phase, preserving critical topological features when the signal is strongest, and accelerates only in later stages. Furthermore, the schedule automatically scales the noise based on the ratio $\tilde{t}/T$. Regardless of the specific value of the Diffusion Horizon T, this guarantees that the system evolves from a state of negligible perturbation at $\tilde{t} = 0$ to a state of total randomness (pure noise) at $\tilde{t} = T$, ensuring consistent boundary conditions for the diffusion process.

With the cumulative retention probability $\alpha_{\tilde{t}}$ established for the sampled timestep, the layout corruption framework is practically implemented as a stochastic substitution process. Guided by the defined schedule, the Uniform Transition Kernel is applied to the ground-truth configuration: each graph element is either preserved (with probability $\alpha_{\tilde{t}}$) or replaced by a random noise category (with probability $1 - \alpha_{\tilde{t}}$).

The specific corruption logic is executed in a single computational step, leveraging the property of the diffusion kernel to transition directly from the clean data to the noise level at $t = \tilde{t}$, following two distinct pathways depending on the considered element:

- **Node Categorical Transition**
  Let $\mathcal{C} = \{\text{Machine, Buffer, Assembly, Disassembly}\}$ be the set of valid node categories. For any given node i, let $\mathbf{v}_0^{(i)} \in \mathcal{C}$ represent its original label in the ground-truth graph. The corresponding corrupted state $\mathbf{v}_{\tilde{t}}^{(i)}$ is directly sampled according to the following mixture distribution:

$$\mathbf{v}_{\tilde{t}}^{(i)} \sim \begin{cases} \mathbf{v}_0^{(i)} & \text{with probability } \alpha_{\tilde{t}} \quad \text{(Signal Preservation)} \\ u \sim \mathcal{U}(\mathcal{C}) & \text{with probability } 1 - \alpha_{\tilde{t}} \quad \text{(Uniform Noise)} \end{cases}$$

  Here, $\mathcal{U}(\mathcal{C})$ represents a discrete uniform distribution over the set $\mathcal{C}$. Therefore, when the corruption branch is activated, the node ignores its history and is assigned any valid component type with equal probability (25%).
- **Edge Binary Transition**
  Similarly, for any given pair of nodes (i, j), let $\mathbf{e}_0^{(ij)}$ represent the original edge state in the ground-truth adjacency matrix E_0. The corresponding corrupted state $\mathbf{e}_{\tilde{t}}^{(ij)}$ is directly sampled according to the following mixture distribution:

$$\mathbf{e}_{\tilde{t}}^{(ij)} \sim \begin{cases} \mathbf{e}_0^{(ij)} & \text{with probability } \alpha_{\tilde{t}} \quad \text{(Signal Preservation)} \\ b \sim \text{Bernoulli}(0.5) & \text{with probability } 1 - \alpha_{\tilde{t}} \quad \text{(Random Topology)} \end{cases}$$

  In the noise branch, the existence of an edge is resampled with probability 50%, representing a state of maximum entropy for a binary variable. In the limit of $\tilde{t} \to T$, this process drives the adjacency matrix towards a completely random graph, where every possible connection is equally likely, entirely decoupling the topological structure from the functional logic.

The execution of these independent transitions results in the generation of the tuple $G_{\tilde{t}}^{(k)}(V_{\tilde{t}}^{(k)}, E_{\tilde{t}}^{(k)})$. This noisy representation constitutes the actual input fed into the neural backbone, which is tasked with reversing the induced entropy based on the sampled time index $\tilde{t}$.

3.1.2 Conditioning Dropout Strategy

Classifier-Free Guidance (CFG) allows the model to operate simultaneously in conditional and unconditional regimes through a Conditioning Dropout Strategy, which systematically "hides" the score from the model by replacing it with a null sentinel value during a fix percentage p_{uncond}.

To enable CFG, the model must master two tasks: generating layouts based on a specific Conditioning Score c and understanding the general rules of layout design without any guidance. To achieve this balance, a Conditioning Dropout Strategy is integrated into the training process. This prevents the model from becoming overly dependent on the score c; without it, the network might ignore the basic structural logic and physical constraints that define a functional industrial layout.

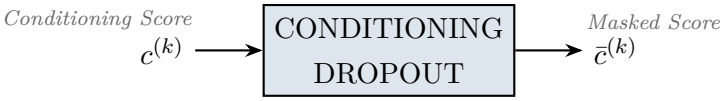


Figure 3.4: Input-output architecture of the Conditioning Dropout block as integrated within the Diffusion Training Framework.

As illustrated in Figure 3.4, this module operates by filtering the Conditioning Score $c^{(k)}$, which is derived from the simulation phase detailed in Appendix A.

This strategy is implemented by applying a stochastic mask to $c^{(k)}$ before it enters the neural network. For each Item (k), a binary indicator $m^{(k)}$ is sampled from a Bernoulli distribution:

$$m^{(k)} \sim \text{Bernoulli}(1 - p_{uncond})$$

where p_{uncond} represents the dropout probability.

This mask modulates the conditioning vector, which is subsequently integrated into the Early Fusion process according to the following switching logic:

- **Conditional Regime ($m^{(k)} = 1$):** the model preserves the original performance score, setting $\bar{c}^{(k)} = c^{(k)}$. This allows the network to observe the direct relationship between the layout's topology and its simulated efficiency.
- **Unconditional Regime ($m^{(k)} = 0$):** the score $c^{(k)}$ is suppressed and replaced by a sentinel value, setting $\bar{c}^{(k)} = -1.0$. Since the performance metrics are normalized within the range $[0, 1]$, choosing a negative value ensures a total lack of ambiguity. This "out-of-range" signal effectively hides the performance target, forcing the model to learn the structural rules of a valid layout without any external guidance.

Crucially, this stochastic dropping is a fundamental prerequisite for the inference mechanism. By alternately exposing the network to the presence and absence of the conditioning signal, the shared parameters θ learn to model both the baseline structural rules (unconditional) and the performance-driven refinements (conditional). This dual expertise allows the model to mathematically isolate the specific features that contribute to a high score, enabling the guidance mechanism to amplify them via logit extrapolation during the sampling phase.

3.1.3 Early Fusion Strategy

Standard diffusion models often operate in a local manner, refining individual node attributes without effectively linking them to the global structure. While proficient at the micro-level, this approach lacks a macro-level strategy to align specific details with the high-level requirements of the entire layout. Consequently, the model may optimize individual components in isolation, failing to ensure their functional and contextual coherence within the complex environment of an industrial plant.

To overcome the limitations of local graph operations, the framework adopts an Early Fusion Strategy. This approach is fundamental to the diffusion process: it ensures that the global context — represented by the temporal state $\tilde{t}^{(k)}$ and the Masked Score $\bar{c}^{(k)}$ — is not merely an auxiliary input, but an integral part of the graph's identity. Therefore, the entire denoising trajectory remains context-aware from the very first layer.

Beyond its conceptual importance, this strategy is technically necessary to translate heterogeneous data into an "algorithmically friendly" format. Standard neural backbones, such as Transformers, require a unified and consistent input repre-

sentation to operate effectively. Early Fusion achieves this by mapping different signals (scalar time, scalar scores, and categorical node labels) into a single, high-dimensional feature space.

As illustrated in Figure 3.5, this module takes the noisy node definition $V_{\tilde{t}}^{(k)}$, the diffusion timestep $\tilde{t}^{(k)}$, and the processed score $\bar{c}^{(k)}$ as inputs, merging them into a unified feature matrix $\mathbf{H}^{(k)}$.

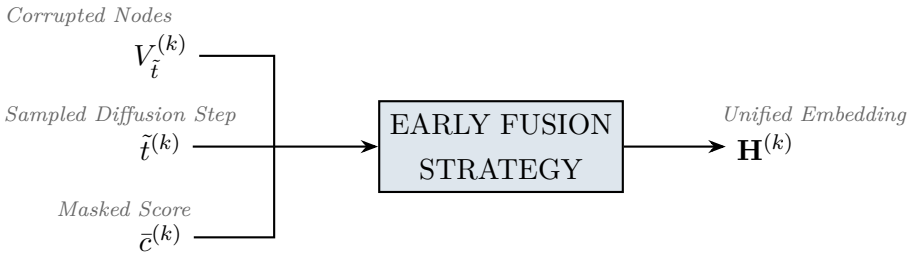


Figure 3.5: Input-output architecture of the Early Fusion Strategy block as integrated within the Diffusion Training Framework.

This integration strategy is executed through two internal logical phases:

1. **Heterogeneous Feature Projection**
   Since the diffusion timestep $\tilde{t}^{(k)}$ and the performance score $\bar{c}^{(k)}$ are scalar and heterogeneous in nature, they are first projected into a shared latent space of dimension d_{emb} to ensure compatibility with the neural network's architecture. The projection mechanisms are defined as follows:
   - **Time Encoding**: to capture temporal dynamics, the scalar timestep $\tilde{t}^{(k)}$ is first mapped into a high-dimensional vector via Sinusoidal Positional Embeddings. This fixed function — detailed in Appendix B.1 — decomposes the timestep into a series of sine and cosine waves across multiple frequencies, providing the network with a rich representation to distinguish between varying noise levels. The resulting embedding is then linearly transformed by the learnable matrix W_t to align it with the model's internal dimension d_{emb}:

$$\mathbf{v}_{time}^{(k)} = W_t \cdot \text{Sinusoidal}(\tilde{t}^{(k)}) \in \mathbb{R}^{d_{emb}}$$

     .
   - **Score Embedding**: the variable $\bar{c}^{(k)}$ serves as the conditioning signal for

the industrial layout, defined within the continuous range $[0, 1]$ during the conditional generation or assigned the sentinel value -1.0 to trigger the unconditional regime in the CFG process.

Since the raw scalar lacks the representational capacity to guide a deep neural architecture, it is projected into a dense feature vector $\mathbf{v}_{score}^{(k)}$ via a Multi-Layer Perceptron (MLP). This encoder, utilizing the Sigmoid Linear Unit (SiLU) activation function to capture complex, non-linear correlations between the input signal and the target graph topology, maps the input into a high-dimensional embedding space. Within this space, the "out-of-range" value -1.0 is translated into a unique, learned "null label" that the backbone recognizes as the absence of conditioning:

$$\mathbf{v}_{score}^{(k)} = \text{MLP}_{score}(\bar{c}^{(k)}) \in \mathbb{R}^{d_{emb}}$$

2. **Broadcasting and Concatenation**: before the final fusion, the corrupted node state $V_{\tilde{t}}^{(k)}$ is converted into a multi-space representation. Let $\mathcal{C} = \{\text{Machine, Buffer, Assembly, Disassembly}\}$ be the set of d_{node} valid node categories. Each node i is transformed from a discrete label into a one-hot encoded vector $\tilde{\mathbf{v}}_i^{(k)} \in \{0, 1\}^{d_{node}}$, effectively mapping the local noisy state into a high-dimensional feature space.

   Incorporating global conditioning signals into this local topology requires addressing a mismatch in scale. While the graph consists of N individual nodes, each possessing a distinct feature vector $\tilde{\mathbf{v}}_i^{(k)}$, the diffusion context ($\mathbf{v}_{time}^{(k)}$ and $\mathbf{v}_{score}^{(k)}$) is globally defined and invariant across the system.

   To align these representations, the global vectors undergo a broadcasting operation over the node dimension. This process effectively replicates the conditioning signal for every layout component, ensuring uniform guidance throughout the graph. The local and global features are then fused via channel-wise concatenation to form the unified state vector for each individual node i:

$$\mathbf{h}_i^{(k)} = \text{Concat}\left(\tilde{\mathbf{v}}_i^{(k)},\ \mathbf{v}_{time}^{(k)},\ \mathbf{v}_{score}^{(k)}\right) \quad \in \mathbb{R}^{d_{in}}$$

   where the resulting input dimension is $d_{in} = d_{node} + 2 \cdot d_{emb}$.

Finally, these individual node-level specifications are aggregated into a single

global feature tensor $\mathbf{H}^{(k)} \in \mathbb{R}^{N \times d_{in}}$ by stacking the vectors $(\mathbf{h}_i^{(k)})^T$ for all N nodes:

$$\mathbf{H}^{(k)} = \begin{bmatrix} (\mathbf{h}_1^{(k)})^T \\ (\mathbf{h}_2^{(k)})^T \\ \vdots \\ (\mathbf{h}_N^{(k)})^T \end{bmatrix}$$

The so obtained matrix $\mathbf{H}^{(k)}$ formally defines the initial state of the system, serving as the structured input for the sequence of graph transformation layers that follow.

3.1.4 Reverse Learning Process

The training phase focuses on optimizing the parameters θ of the neural network to accurately invert the diffusion process. The primary objective is to approximate the transition distribution $p_\theta(G_0|G_{\tilde{t}}, \tilde{t}, \bar{c})$, effectively recovering the structural logic of the industrial layout from the stochastic entropy introduced during the forward phase.

Mathematically, this task involves optimizing the estimator function $\epsilon_\theta(\mathbf{H}^{(k)}, E_{\tilde{t}}^{(k)})$, where the matrix $\mathbf{H}^{(k)}$ — as previously established — encapsulates the holistic configuration of the system by integrating noisy node states $V_{\tilde{t}}^{(k)}$, the temporal index $\tilde{t}^{(k)}$, and masked performance conditioning $\bar{c}^{(k)}$. Simultaneously, the adjacency matrix $E_{\tilde{t}}^{(k)}$ supplies the topological skeleton required for the message-passing operations. As illustrated in Figure 3.6, the Graph Transformer processes these inputs to generate two distinct sets of raw numerical scores, or logits: the node logits $(L_V^{(k)})$ and the edge logits $(L_E^{(k)})$.

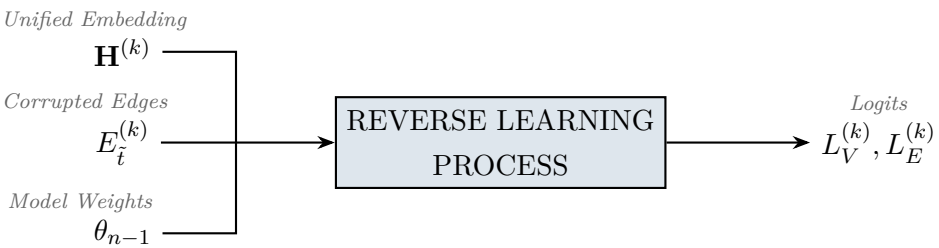


Figure 3.6: Input-output architecture of the Reverse Learning Process block as integrated within the Diffusion Training Framework.

To parameterize the neural backbone, the system adopts the TransformerConv architecture. This operator represents a specialized implementation of the Unified Message Passing (UniMP) framework proposed by Shi et al. [34], which serves as the foundational integration of Transformer-based self-attention within Graph Neural Networks.

Available in the PyTorch Geometric library, this architecture adapts the self-attention mechanism to operate directly within a message-passing scheme. Unlike general-purpose Transformers, the attention coefficients here are computed exclusively for the existing edges in the graph. This allows the network to dynamically weight the importance of neighboring components, identifying which topological relationships are most relevant for reconstructing the logical structure of the industrial system.

The expressivity of the architectural core is further enhanced by a Multi-Head Attention mechanism. In this configuration, the attention process is parallelized across n_{head} independent "heads", which allows the model to partition the hidden representation into multiple subspaces. Each head acts as a distinct filter focusing on a specific subspace of the node features, allowing the model to capture a diverse set of structural dependencies simultaneously. The insights from all heads are then combined and projected, providing a comprehensive representation that encapsulate the complex nature of factory layouts.

The adoption of this backbone is specifically designed to handle the nature of manufacturing layouts, which are discrete, sparse, and permutation-invariant structures. In this context, the functional logic of the system remains identical regardless of the arbitrary numerical order in which nodes are indexed, as the operational reasoning depends strictly on the topology rather than spatial proximity. This graph-based approach ensures that the model's predictions are governed by the logical connections between components, enabling the implementation of a Discrete Denoising Diffusion Probabilistic Model (D3PM) tailored for the rigorous constraints of industrial environments [37].

The schematic representation of the system, illustrating the internal mechanism of a single layer z and its hierarchical integration, is depicted in Figure 3.7.

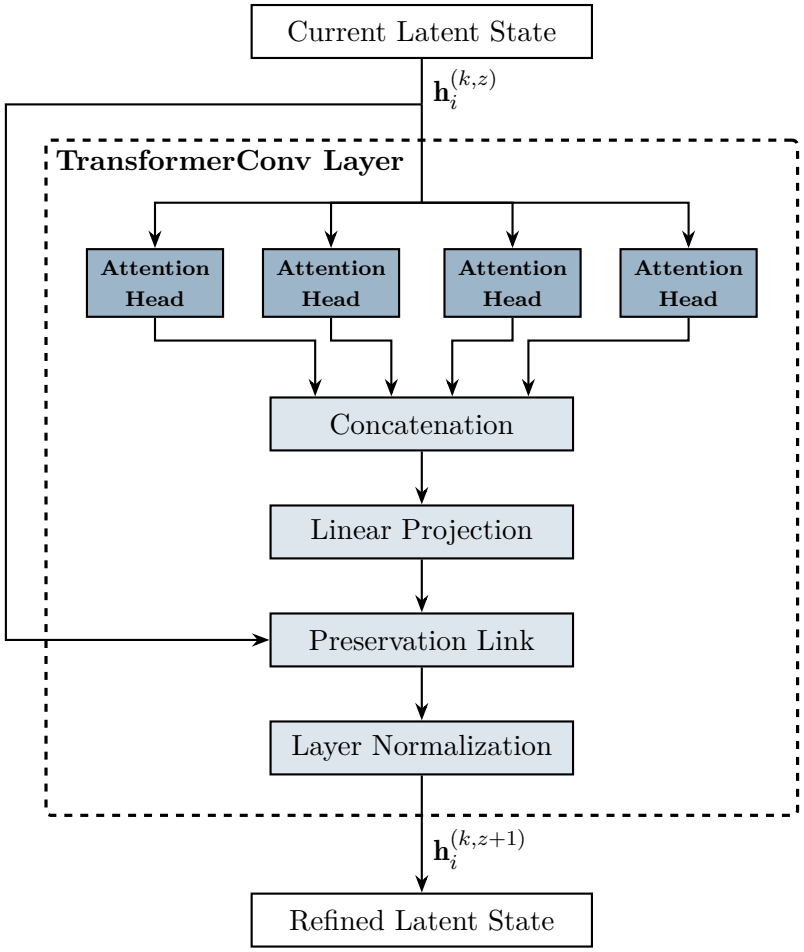


Figure 3.7: Architectural detail of the neural backbone: TransformerConv layer with multi-head attention mechanism.

The architectural flow within each TransformerConv unit follows a dual-path logic. The Current Latent State is simultaneously routed through a multi-head attention mechanism and an identity preservation branch. While the four Attention Head units parallelize the extraction of topological relationships from the neighborhood, the parallel shortcut ensures the persistence of the node's intrinsic features. These signals are integrated via Concatenation, synthesized through a Linear Projection, and subsequently merged with the initial state at the Preservation Link. The final processing by the Layer Normalization block produces the Refined Latent State, concluding the internal operation of a single Transformer-based Layer.

This structure is fundamentally recursive: the state produced at the exit of layer z serves as the direct input for layer $z + 1$. Through Z successive iterations, the

model performs a progressive refinement of the graph representation, allowing for a sophisticated integration of global topological dependencies and local component features. Within this framework, the trainable parameters θ are embedded as weight matrices governing the attention mechanisms, the linear transformations, and the self-update projections.

The process begins from the initialized unified state vector for each individual node $\mathbf{h}_i^{(k)}$, which integrates nodes, time, and performance goals attributes. As information passes through the layers $z \in \{0, \dots, Z-1\}$, each node updates its state using an anisotropic propagation rule [34]:

$$\mathbf{h}_i^{(k,z+1)} = \mathrm{LN}\left(W_1 \mathbf{h}_i^{(k,z)} + \mathrm{CONCAT}_{r=1}^{n_{head}} \left(\sum_{j \in \mathcal{N}(i)} \mathcal{A}_{i,j}^{(r)} W_2^{(z)} \mathbf{h}_j^{(k,z)} \right) \right)$$

The mathematical terms involved in the equation represent the primary operational stages of the framework:

- **Self-Update** ($W_1 \mathbf{h}_i^{(k,z)}$): corresponds to the information flowing through the Preservation Link. It operates exclusively on the current representation to preserve the node's functional identity and conditioned constraints, preventing them from being neutralized by external topological data.
- **Multi-Head Weighted Aggregation** ($\mathrm{CONCAT}_{r=1}^{n_{head}}(\dots)$): constitutes the core message gathered from the neighborhood $\mathcal{N}(i)$, where the aggregation is split across n_{head} parallel channels. Each individual Attention Head unit r computes its own coefficients $\mathcal{A}_{i,j}^{(r)}$ to dynamically weigh the importance of connections within its specific feature subspace. This allows the model to filter structural noise and capture relevant logistical dependencies from different "perspectives" simultaneously.
- **Numerical Stabilization** (LN): denotes the Layer Normalization operator acting on the sum of the two previous paths. This phase ensures the numerical stability of the system, maintaining latent features within consistent scales across the Z iterations of the backbone.

The final execution of the neural backbone layers produces the categorical logits $L_V^{(k)}$ and $L_E^{(k)}$, which serve as the structural foundation for the subsequent optimization and layout reconstruction. In the context of deep learning, these logits represent the raw numeric scores generated by the model's final layer before they are normal-

ized into probabilities (via a Softmax function). Operating within an unbounded real-number space, these values quantify the model's "confidence" in each discrete category, where higher magnitudes indicate a stronger prediction for a specific class. The logits $L_V^{(k)}$ and $L_E^{(k)}$ are defined as:

- **Node Logits** ($L_V^{(k)} \in \mathbb{R}^{N\times 4}$): for each of the N vertices in the graph, the backbone produces a vector of d_{node} raw scores corresponding to the categorical set $\mathcal{C} = \{\text{Machine, Buffer, Assembly, Disassembly}\}$. This output models a multi-class classification task where the highest logit determines the predicted functional identity of the component, ensuring each element is assigned its correct operational role.
- **Edge Logits** ($L_E^{(k)} \in \mathbb{R}^{N\times N\times d_{edge}}$): the system's connectivity is formalized by a three-dimensional tensor that maps every possible pair of nodes (i, j) within the $N \times N$ adjacency matrix. By predicting $d_{edge} = 2$ scores per entry, the model performs a binary classification between the absence (0) or existence (1) of a logistical flow, allowing it to reconstruct the global topology while filtering out stochastic noise.

This high-dimensional output space allows the estimator to capture the complete structural logic of the plant. While $L_V^{(k)}$ focuses on the individual identity of the components, $L_E^{(k)}$ captures the global topology, ensuring that the reconstructed layout is not only correctly typed but also functionally interconnected. These tensors serve as the direct inputs for the multi-objective loss function.

3.1.5 Composite Loss Function

To parameterize the training objectives, the framework defines a formal error metric to evaluate the model's performance. Formally, a loss function $\mathcal{L}$ is a mathematical operator that maps the high-dimensional output of the network to a scalar value, representing the cost associated with a specific prediction error [8]. In this context, the objective is defined as:

$$\mathcal{L}_{total}^{(k)} = \mathcal{L}(f_\theta(G_{\tilde{t}}, \tilde{t}, \bar{c}), G_0)$$

where f_θ is the neural estimator and G_0 is the ground-truth layout. Once the data has been processed by the neural network, the training phase concludes with

the calculation of this Composite Loss Function. This step is essential because the loss value measures the discrepancy between the model's predictions and the target layouts. This feedback is then used to update the shared parameters θ, allowing the model to progressively learn and improve its generative performance.

As shown in Figure 3.8, this module acts as a supervisor that combines multiple signals to guide the learning process. It evaluates the structural logits $(L_V^{(k)}, L_E^{(k)})$ — which encapsulate the model's predictions for nodes and edges — integrating them with Dataset Statistics to ensure statistical alignment and Boundary Constraints to enforce the plant's functional and engineering logic. These components are then aggregated into the total objective function, $\mathcal{L}_{total}^{(k)}$, which represents the final scalar value used to compute the gradients needed to optimize the network parameters.

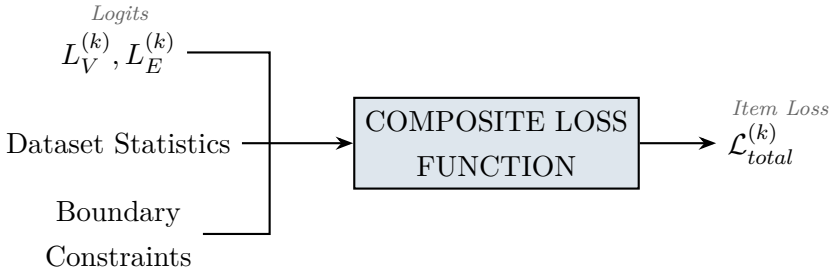


Figure 3.8: Input-output architecture of the Composite Loss Function block as integrated within the Diffusion Training Framework.

This composite metric is designed to balance three competing goals: accurate topological reconstruction, statistical fidelity to the training data, and strict adherence to industrial engineering rules. To achieve this, the total loss $\mathcal{L}_{total}^{(k)}$ is formulated as a weighted sum of three distinct loss contributions:

$$\mathcal{L}_{total}^{(k)} = \mathcal{L}_{reconstruction}^{(k)} + \lambda_{KL}\mathcal{L}_{regularization}^{(k)} + \lambda_{semantic}\mathcal{L}_{semantic}^{(k)}$$

Each of these terms addresses a specific requirement of the generation process:

1. **Reconstruction Loss**
   The primary diffusion model objective is to learn the conditional probability distribution $p_\theta(G_0|G_{\tilde{t}}, \tilde{t}, \bar{c})$, enabling the model to recover the original graph structure from the noisy state. This factor is composed by two sub-terms:

$$\mathcal{L}_{reconstruction}^{(k)} = \mathcal{L}_{node}^{(k)} + \mathcal{L}_{edge}^{(k)}$$

- **Node Classification Loss ($\mathcal{L}_{node}$)**: this term governs the accuracy of node typing. For each node i and each category $\chi \in \mathcal{C}$, let $\tilde{\mathbf{v}}_{0,i\chi}^{(k)} \in \{0,1\}^{d_{node}}$ be the ground-truth one-hot label representing the original node type for the k-th graph and the χ-th category. The corresponding predicted probability $\mathbf{P}_{V,i\chi}^{(k)} \in [0,1]$, defined over the node set V, is obtained by applying the Softmax function to the entire logit vector $L_V^{(k)}$ of node i and then selecting the entry corresponding to the category χ [36]. Given the mutually exclusive nature of the categories, the standard Categorical Cross-Entropy loss is employed to minimize the divergence between the predicted distribution and the ground-truth label:

$$\mathcal{L}_{node}^{(k)} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{\chi\in\mathcal{C}} \tilde{\mathbf{v}}_{0,i\chi}^{(k)} \log(\mathbf{P}_{V,i\chi}^{(k)})$$

  The categorical cross-entropy employed for node typing ensures the operational integrity of the industrial layout by isolating and evaluating the model's confidence in the correct functional identities. This mechanism is mathematically governed by the ground-truth one-hot label $\tilde{\mathbf{v}}_{0,i\chi}^{(k)}$, which acts as a binary selector; by nullifying the contribution of incorrect categories, it forces the loss function to focus exclusively on the probability assigned to the true class of each component. As this predicted probability $\mathbf{P}_{V,i\chi}^{(k)}$ diverges from the target, the negative log-likelihood applies an exponential penalty, generating potent gradients that rapidly steer the model's parameters toward the correct operational logic. This logarithmic penalty is further refined by size-invariant normalization through the inclusion of the $1/N$ factor, which stabilizes the error signal relative to the total number of nodes. Such stabilization ensures that the learning process remains consistent regardless of the plant's scale, preventing larger, more complex layouts from dominating the gradient updates and allowing the model to generalize effectively across both compact work cells and expansive factory departments.
- **Weighted Edge Prediction Loss ($\mathcal{L}_{edge}^{(k)}$)**: this term measures the model's ability to reconstruct the adjacency matrix. For any node pair (i,j), let $\mathbf{e}_{0,ij,k} \in \{0,1\}$ be the ground-truth binary label, one-hot encoded for classes $k=1$ (edge presence) and $k=0$ (edge absence). The corresponding pre-

dicted probability $\mathbf{P}_{E,ij,k} \in [0,1]$ is obtained by applying the Softmax function to the edge logit vector $L_E^{(k)}$ associated with the pair (i,j), thereby normalizing the raw network outputs into a valid probability distribution over the two classes.

Since valid connections are extremely rare in industrial layouts, the resulting sparsity often leads a standard loss function to converge toward a trivial, fully disconnected solution. To mitigate this class imbalance, a weighted Binary Cross-Entropy strategy is employed. A static weight $w_{pos} > 1$, calculated as inversely proportional to the dataset edge density, is applied to the positive class to prioritize the recovery of sparse connections. The loss is formally defined as:

$$\mathcal{L}_{edge} = -\frac{1}{N^2} \sum_{i,j} \sum_{k=0}^{1} w_k \cdot \mathbf{e}_{0,ij,k} \log(\mathbf{P}_{E,ij,k})$$

This function extends the logarithmic penalty mechanism by introducing a balancing factor w_k to manage the extreme sparsity of connections in industrial layouts. While the $1/N^2$ term ensures scale-invariant stability relative to the plant dimensions, the static weight w_{pos} selectively amplifies the gradients originating from existing edges ($k = 1$). This targeted intervention prevents the model from converging toward a trivial, fully disconnected solution, forcing the optimization to prioritize the recovery of critical logistical flows over the simple classification of empty space.

2. **Regularization Loss**

In order to align the generated samples with the global statistics of the training distribution, the framework incorporates a regularization term based on the Kullback-Leibler (KL) divergence. This term acts as a statistical anchor, ensuring that the proportions of different components — such as the ratio between machines and buffers — remain consistent with the empirical distributions found in actual industrial plants. By aligning the model's output with the training data's global statistics, this regularizer prevents distributional drift. Without this constraint, the model might produce layouts that are locally coherent but statistically implausible thereby ensuring the generated samples preserve the structural diversity and class balance inherent in the industrial domain.

To address the distribution matching between the generated layouts and the real-world data, two key probability profiles are defined. These profiles rely on a set of discrete categories $\mathcal{C}$, which adapts to the specific data modality: for node classification, $\mathcal{C}$ corresponds to the four functional types $\{0, 1, 2, 3\}$, whereas for edge prediction, it represents the binary connection states $\{0, 1\}$. The comparison is governed by the following definitions:

- **Empirical Marginal Probability** ($\mathbf{P}_{dataset}(\ell) \in [0, 1]$): represents the static frequency of class ℓ pre-calculated across the entire training dataset. This acts as a global reference profile, capturing the "average" composition of a valid industrial plant.
- **Predicted Marginal Probability** ($\hat{\mathbf{P}}_{batch}(\ell) \in [0, 1]$): is defined as the empirical marginal distribution of class ℓ, computed across the current batch of predicted graphs.

The regularization loss is computed separately for nodes and edges and then summed. By minimizing the divergence between the batch predictions and the dataset marginals, this term ensures that the model maintains the same class proportions found in the training data, effectively preventing the generator from producing statistically biased layouts:

$$\mathcal{L}_{KL}^{(k)} = \sum_{\ell \in \mathcal{C}} \hat{\mathbf{P}}_{batch}(\ell) \log \left(\frac{\hat{\mathbf{P}}_{batch}(\ell)}{\mathbf{P}_{dataset}(\ell)} \right)$$

3. **Semantic Constraints Loss**

   To enforce functional validity according to strict engineering rules, the framework incorporates a specialized set of Semantic Constraints. These differentiable penalty terms are calculated directly on the predicted probability tensors, effectively guiding the optimization gradients toward valid regions of the topological search space:

$$\mathcal{L}_{semantic}^{(k)} = \lambda_{mask}\mathcal{L}_{mask}^{(k)} + \lambda_{system}\mathcal{L}_{system}^{(k)} + \lambda_{connectivity}\mathcal{L}_{connectivity}^{(k)}$$

   These constraints act as an essential input to the Composite Loss Function, ensuring that the generated layouts do not just mimic data patterns but adhere to the physical and operational logic of a manufacturing plant.

Each component of this domain-driven guidance addresses a specific layer of industrial logic:

- **Forbidden Connection Penalty** ($\mathcal{L}_{mask}^{(k)}$): this component addresses hard structural violations that are strictly prohibited by manufacturing logic, such as self-loops or direct Buffer-to-Buffer connections. Let $M_{forbidden} \in \{0,1\}^{N\times N}$ denote a pre-computed binary mask derived from the node types, where an entry $\mathbf{M}_{ij} = 1$ indicates that a directed link from node i to node j is invalid. The loss is computed as the Mean Squared Error (MSE) of the predicted edge probability for class $k = 1$ (edge presence) on these forbidden links, driving their likelihood toward zero:

$$\mathcal{L}_{mask} = \frac{1}{\sum_{i,j}\mathbf{M}_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbf{M}_{ij} \cdot \left(\mathbf{P}_{E,ij,1}\right)^2$$

- **System Type Guidance** ($\mathcal{L}_{system}^{(k)}$): this term governs the global network topology by monitoring the soft degrees of Buffer nodes. Let $d_{in}(i) = \sum_{j=1}^{N}\mathbf{P}_{E,ji,1}$ and $d_{out}(i) = \sum_{j=1}^{N}\mathbf{P}_{E,ij,1}$ be the continuous input and output degrees of a buffer i, calculated by summing the predicted probabilities for the "edge presence" class. A minimal tolerance threshold δ is utilized to identify buffers acting as sources or sinks.

  The formulation of the loss function is adapted to the specific requirements of the system type:

  - **Open System**

    An Open System is required to facilitate material flow that enters from and exits to the external environment. The objective is to ensure the existence of at least one global input source ($d_{in} \approx 0$) and at least one global output sink ($d_{out} \approx 0$). If the total count of such nodes is less than one, a linear penalty is applied via the ReLU operator:

$$\mathcal{L}_{sys}^{(open)(k)} = \text{ReLU}\left(1 - \sum_{i\in V_B} \mathbb{1}_{\{d_{in}(i)<\delta\}}\right) + \text{ReLU}\left(1 - \sum_{i\in V_B} \mathbb{1}_{\{d_{out}(i)<\delta\}}\right)$$

    where $\mathbb{1}_{\{\cdot\}}$ represents a soft indicator function that returns 1 if the condition is met and 0 otherwise. This term acts as a differentiable counter: if the summation of indicators (the total count of sources or sinks) is zero,

the ReLU function activates the penalty to force the model to generate the missing topological feature.

- **Closed System**
  In Closed Systems, external input and output are prohibited, as material must circulate entirely within the internal network. The loss function imposes a quadratic penalty on any Buffer acting as a source or a sink to force the generation of a recirculating topology:

$$\mathcal{L}_{sys}^{(closed)(k)} = \sum_{i \in V_B} \left[\text{ReLU}(\delta - d_{in}(i))^2 + \text{ReLU}(\delta - d_{out}(i))^2 \right]$$

- **Input-Only System**
  This configuration is designed to receive material from the external environment without providing final output nodes. While input sources are permitted, the formation of output sinks is strictly penalized through a quadratic penalty on the outgoing soft degree:

$$\mathcal{L}_{sys}^{(in)(k)} = \sum_{i \in V_B} \left[\text{ReLU}(\delta - d_{out}(i))^2 \right]$$

- **Output-Only System**
  An Output-Only System does not receive external flows but must allow for the evacuation of products. The loss function prevents any Buffer from acting as an external input source by penalizing null incoming soft degrees:

$$\mathcal{L}_{sys}^{(out)(k)} = \sum_{i \in V_B} \left[\text{ReLU}(\delta - d_{in}(i))^2 \right]$$

- **Connectivity Regularization** ($\mathcal{L}_{connectivity}$): this term is enforced to ensure the generated layout forms a cohesive functional unit, discouraging the formation of disjoint components. The optimization leverages the spectral properties of the soft Graph Laplacian, $L = D - \hat{A}$, by targeting the maximization of its second smallest eigenvalue, λ_2 (the Fiedler value). As λ_2 serves as a differentiable proxy for the algebraic connectivity of a graph, its inclusion in the loss function effectively steers the generative process toward globally integrated topologies [28].
  Since a strictly positive Fiedler value ($\lambda_2 > 0$) mathematically guarantees

that the graph is connected, the loss function is designed to maximize λ_2 up to a predefined target margin τ (e.g., 0.5). To complement this global spectral constraint, a local degree-based penalty is applied to ensure functional viability, penalizing any node i whose aggregate edge strength falls below a minimum safety threshold δ:

$$\mathcal{L}_{connectivity} = \text{ReLU}(\tau - \lambda_2) + \frac{1}{N}\sum_{i=1}^{N} \text{ReLU}(\delta - (d_{in}(i) + d_{out}(i)))$$

The weighting coefficients λ_{KL}, $\lambda_{semantic}$, and the internal guidance parameters λ_{mask}, λ_{system}, and $\lambda_{connectivity}$ serve as critical hyperparameters that balance the trade-off between generative freedom and structural compliance. These factors are essential for normalizing the gradient magnitudes across different numerical scales — ranging from the standard cross-entropy of node classification to the spectral penalties of connectivity. By carefully tuning these lambdas, the framework prevents any single objective from dominating the optimization landscape, ensuring that the model learns to prioritize topological accuracy, statistical fidelity, and engineering validity simultaneously.

3.1.6 Optimization Strategy

The network's parameters θ are refined through a multi-objective optimization process that minimizes the aggregate loss functions. As illustrated in Figure 3.9, this phase constitutes the core learning mechanism, where the gradients derived from the error signals are backpropagated to align the model's weights with the structural requirements of industrial layouts.

Crucially, this optimization is conducted independently for each system type. Since the objective functions — specifically the Semantic Constraints — are strictly tailored to the target topology, this separation is vital to ensure that the model parameters are calibrated for one specific set of rules at a time. By isolating the training sessions, the framework avoids the "gradient interference" that would occur if a single global model attempted to simultaneously satisfy the divergent and often conflicting requirements defined by the system-type tailored loss functions.

Figure 3.9: Input-output architecture of the Optimization Strategy block as integrated within the Diffusion Training Framework.

To update the model parameters, the framework considers mini-batches rather than individual graphs, with dimensions controlled by the Batch Size (B) hyperparameter. During this phase, the individual total losses $\mathcal{L}_{total}^{(k)}$ are aggregated to provide a more stable gradient. Specifically, for each iteration n, the system identifies a subset of indices $\mathcal{B}_n$ representing the B samples selected for that specific step. These individual losses are then averaged to compute the Batch Loss ($\mathcal{L}_{batch}^{(n)}$), which serves as the actual feedback signal for the optimizer:

$$\mathcal{L}_{batch}^{(n)} = \frac{1}{B} \sum_{k \in \mathcal{B}_n} \mathcal{L}_{total}^{(k)}$$

To compute the final update, the system requires two additional inputs: the Current Weights (θ_{n-1}), which serve as the starting point for the update, and the

Past Moments (m_{n-1}, v_{n-1}), which represent the accumulated "memory" of previous gradients.

The optimization problem is particularly challenging because the "terrain" of the loss function is highly irregular, featuring steep slopes governed by strict engineering constraints and flatter regions related to topological fine-tuning. To navigate this complex landscape, the Adam (Adaptive Moment Estimation) optimizer was chosen. At each iteration n, the algorithm first computes the gradient of the batch loss function with respect to the current parameters:

$$g_n = \nabla_\theta \mathcal{L}_{batch}^{(n)}(\theta_{n-1})$$

Using g_n as the raw input, Adam estimates the First Moment (Momentum) and the Second Moment (Adaptive Scaling) to determine a parameter-specific learning rate. These mechanisms allow the system to smooth the optimization path via inertia and maintain stability by dampening oscillations in high-variance directions.

A comprehensive mathematical breakdown of the moment estimation process and the relative bias-correction logic is provided in Appendix B.2.

The final phase of the strategy produces the updated weights (θ_n) and the associated moments (m_n, v_n) for the next iteration, following the update rule:

$$\theta_n = \theta_{n-1} - \alpha \cdot \frac{\hat{m}_n}{\sqrt{\hat{v}_n} + \delta}$$

In this equation, α is the Learning Rate, while δ is a small constant for numerical stability. Mathematically, this formulation implies that the update magnitude is normalized individually for every single weight in the neural backbone.

This adaptive behavior is vital for the proposed framework. Since the loss landscape includes discrete engineering requirements — such as mandatory connectivity between machines and buffer — it often features sharp discontinuities and irregular gradients. By evaluating the average behavior of a mini-batch and adjusting the step size for each parameter based on its historical behavior, the optimization strategy can navigate these complex transitions more robustly, filtering out the noise of individual samples and maintaining stable progress even when the mathematical "terrain" becomes unpredictable.

The algorithm's convergence behavior is highly sensitive to its initialization. Specifically, the hyperparameters β_1 and β_2 regulate the exponential moving averages (memory length), while α determines the learning rate. The specific configurations of these parameters, along with the theoretical rationale for their selection, are detailed in Section 4.4.

3.2 The Generative Pipeline

The Generative Pipeline is the end-to-end framework that transforms abstract user requirements into functional industrial layouts. This inference strategy bridges the gap between the probabilistic nature of the diffusion model and the deterministic rigidity required by industrial engineering. By integrating the standard Reverse Diffusion Process with a discrete projection and validation framework, the system ensures that every output satisfies all structural and functional requirements. The sequence of stages and the interaction between the modules — from initial generation to final quality control — are illustrated in Figure 3.10.

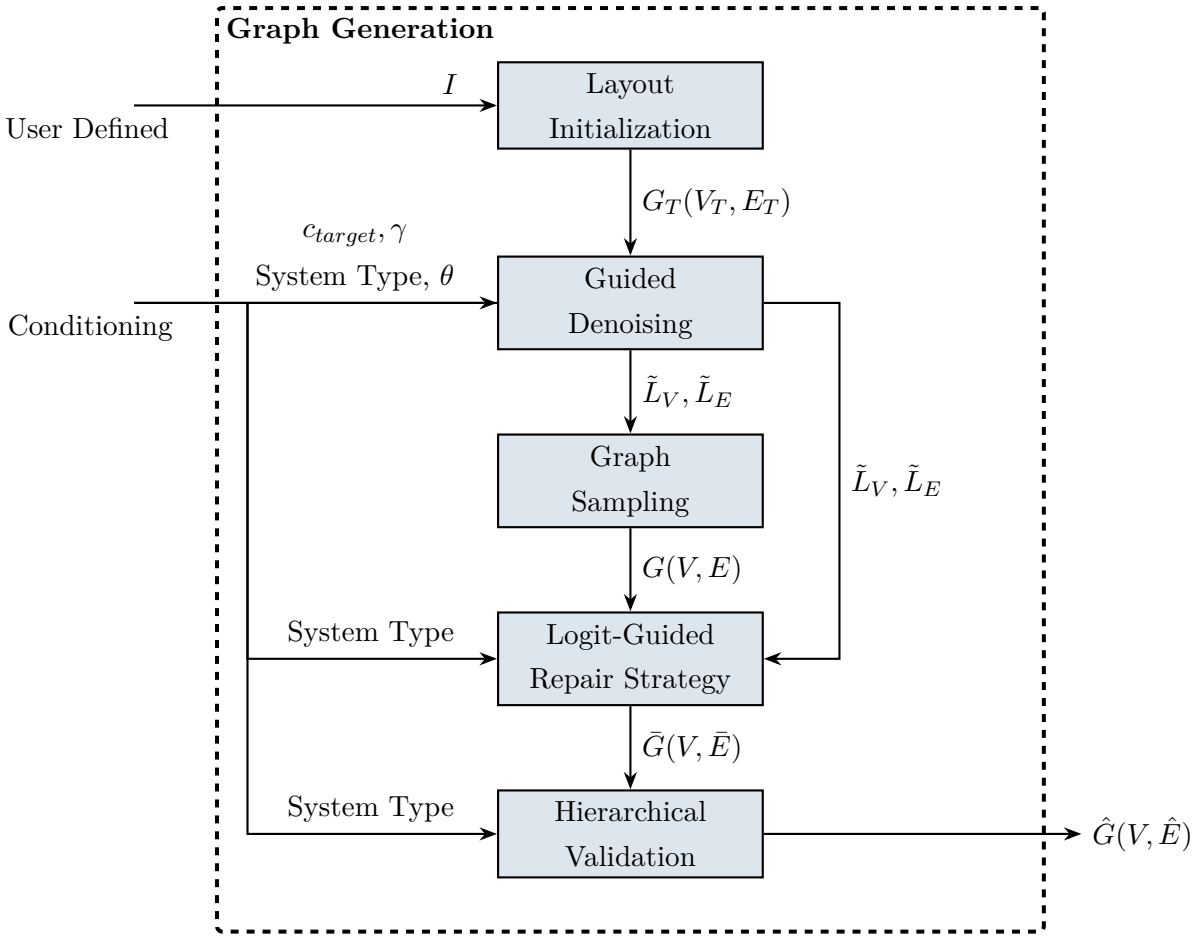


Figure 3.10: Procedural architecture of the Generative Pipeline.

As shown in the diagram, the workflow operates through four sequential stages: Layout Initialization, Guided Denoising, Logit-Guided Repair, and Hierarchical

Validation. This progression guides the latent design from an initial noisy state $G_T(V_T, E_T)$ to a final validated graph $\hat{G}(V, \hat{E})$. By processing the structural logits $(\tilde{L}_V, \tilde{L}_E)$ through a dedicated repair strategy and a multi-level validation check, the pipeline ensures that the generated layout is ready for performance evaluation while strictly adhering the plant's functional and engineering logic.

3.2.1 Layout Initialization

As illustrated in the conceptual block diagram in Figure 3.11, the sampling process begins with the initialization of the structural constraints based on the specific industrial requirements of the plant.

Figure 3.11: Input-output architecture of the Layout Initialization block as integrated within the Generative Pipeline.

These requirements are formalized into a target inventory tuple $I = (N_M, N_B, N_A, N_D)$, representing the specific count of Machines, Buffers, Assembly, and Disassembly units. The selection of these terms is not arbitrary but directly reflects the functional primitives of the industrial system. The inventory acts as a hard structural constraint, ensuring that the generative process remains grounded in the physical reality of the facility. From this tuple, the node state tensor V_T is deterministically constructed by concatenating the one-hot encoded vectors $\tilde{\mathbf{v}}_i$ corresponding to the requested component counts.

By initializing the node states in this deterministic manner, it is possible to constrain the generation process, ensuring that the resulting graphs are strictly compliant with the provided specifications. This approach guarantees that the model explores only the topological space relevant to the user's inventory, maintaining the functional integrity of the requested plant configuration.

To ensure permutation invariance and prevent any positional bias, the order of nodes within V_T is randomized via random permutation, forcing the model to rely solely on topological features rather than fixed indices. Simultaneously, sampling each edge $\mathbf{e}_{ij,T}$ from a uniform categorical distribution $\mathcal{U}\{0, 1\}$, the adjacency matrix E_T is initialized in a state of maximum entropy, reflecting the theoretical limit of

the Forward Corruption Process as $\tilde{t} \to T$. This initialization results in the construction of a purely random graph $G_T(V_T, E_T)$, which serves as the starting point for the generation process.

3.2.2 Guided Denoising

The core of the generation phase is the Guided Denoising process, which iteratively reverses the diffusion entropy. This stage acts as the operational engine of the framework, where the Discrete Denoising Diffusion Probabilistic Model effectively converts the initial input noise into a structured industrial layout. Unlike standard unconditional generation, in this phase the model actively steers the design toward high-performance configurations using CFG.

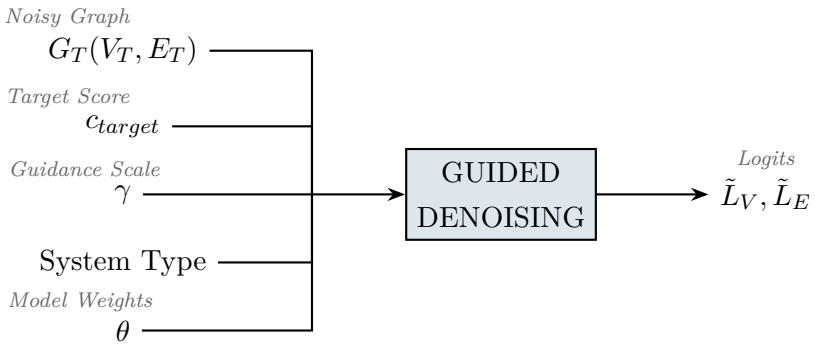


Figure 3.12: Input-output architecture of the Guided Denoising block as integrated within the Generative Pipeline.

As illustrated in Figure 3.12, this stage integrates several user-defined requirements with the model's learned logic to drive the reconstruction. Beyond the initial noisy state G_T, the denoising trajectory is steered by a set of global control parameters that define the identity and operational targets of the plant:

- **Target Performance Score** (c_{target}): a normalized scalar value that serves as the active control signal, instructing the model to synthesize a topology consistent with a specific efficiency level.
- **Guidance Scale** (γ): a hyperparameter that dictates the intensity of the Classifier-Free Guidance; higher values force the model to adhere more strictly to the performance target by amplifying the features associated with high scores.

- **System Type**: a categorical setting — such as Open, Closed, Input-Only, or Output-Only — that defines the global topological macro-category, ensuring the logistical flow respects the intended plant architecture.
- **Pre-trained Weights** (θ): these are the internal parameters learned by the neural network during the previous training process. As a result of the independent training conducted for each system type, the framework does not utilize a single universal model; instead, it dynamically loads the specific weight configuration that matches the target system topology. This operation serves as the functional trigger for the inference phase, enabling the Graph Transformer architecture to access the specialized knowledge required to recognize and reconstruct valid industrial patterns from the initial noise. By aligning the model's internal parameters with the observed configuration, the system ensures that the denoising trajectory remains grounded in the correct operational domain, effectively preventing any structural interference between divergent manufacturing paradigms.

By synthesizing these inputs, the model iteratively removes entropy to transform the initial random state into a functional industrial layout. In this context, it is fundamental to distinguish this approach from the traditional guidance systems described in Section 1.3.3. While standard methodologies typically focus on modeling and estimating the noise error to define the optimal sampling trajectory toward a target, the method implemented here is not oriented toward noise reconstruction. Instead, it is aimed directly at the structural synthesis of the graph.

Within this framework, the logits do not serve to recover the corruption introduced during the noising process; rather, they act as direct estimators of the layout's connectivity and node roles, specifically tailored to satisfy the performance scores and conditioning signals imposed as objectives. Consequently, the reverse process is redefined as a targeted search within the architectural design space, where each step refines the probability of existence for specific industrial patterns.

During this process, the system executes the reverse diffusion over T timesteps. At each denoising step t, the model predicts two sets of these guided logits (L), which act as confidence scores for the graph's fundamental components: node logits (L_V) determine the probability of each functional category, while edge logits (L_E) define the likelihood of connections between node pairs.

To steer the generation toward the desired results, the model performs dual

parallel passes to estimate these values:

- **Conditional Pass** (L^{cond}): the model predicts the logits based on the imposed target c_{target}. To achieve this, the scalar c_{target} is projected through a Multi-Layer Perceptron (MLP) into a high-dimensional embedding as in the training process.
- **Unconditional Pass** (L^{uncond}): the model predicts the logits using a "null" token ($\bar{c} = -1.0$). In this pass, the performance embedding is replaced by a fixed sentinel value, forcing the network to rely purely on basic structural rules and industrial design priors, providing a baseline for a valid layout.

The final guidance signal is computed by extrapolating the difference between these two parallel predictions. For each potential component — whether a node category or an edge connection — the modified logit $\tilde{L}$ is calculated as:

$$\tilde{L} = L^{uncond} + \gamma \cdot (L^{cond} - L^{uncond})$$

The Guidance Scale γ is crucial here: it defines how aggressively the model should prioritize the performance benchmark over the generic structural baseline. This operation shifts the probability mass toward regions of the sample space that are strongly correlated with the Target Performance, effectively suppressing suboptimal or generic structures.

Once these guided logits are obtained, the framework handles nodes and edges with different strategies to ensure both performance and validity. For the nodes, the constraint is enforced by treating the user-defined inventory V_T as a hard template. Instead of allowing the model to freely alter the number of components, the system uses the guided node logits $\tilde{L}_V$ exclusively to assign the most appropriate roles to the fixed set of available machines and buffers. This ensures that the generated plant always aligns with the specific physical requirements specified by the user.

The structural validity of the connection is instead managed through a differentiable filtering process applied to the edge logits $\tilde{L}_{E,ij,k}$. Immediately following the CFG extrapolation, the framework applies a Strict Projector. This module acts as a mathematical gate that shapes the probability distribution by enforcing "hard" structural constraints within the continuous logit space. By suppressing illegal activations — specifically targeting the logits associated with the "edge presence" class ($k = 1$) — before the graph is generated, the Projector ensures that the subsequent

sampling starts from a distribution already aligned with domain logic.

The Projector operates through the following specialized masking stages, shaping the edge logits $\tilde{L}_{E,ij,k}$ before they are normalized into the final edge probabilities $\mathbf{P}_{E,ij,k}$ via the Softmax function:

- **Diagonal Zero-Masking (Self-Loops):** to prevent a node from establishing a logistical flow with itself, the Projector applies a negative infinite mask to these entries, ensuring that the probability of a self-loop is effectively zeroed out before any discretization occurs:

$$\tilde{L}_{E,ii,1} \to -\infty \quad \text{resulting in } \mathbf{P}_{E,ii,1} \approx 0$$

- **Categorical Sequence Filtering (Buffer-to-Buffer):** manufacturing logic strictly prohibits the placement of two storage units in direct sequence. The Projector utilizes the node type definitions from V_T to identify pairs where both nodes i and j are labeled as Buffers. It then suppresses the corresponding edge logits for the "edge present" class:

$$\tilde{L}_{E,ij,1} \to -\infty \quad \forall (i,j) \in V_{Buffer} \times V_{Buffer}$$

  This ensures that the probability of a connection between two storage units is zeroed out during normalization ($\mathbf{P}_{E,ij,1} \approx 0$).
- **Soft Cardinality Weighting:** finally, the Projector prepares the logits for the sampling phase by acting as a confidence filter to prevent nodes from establishing excessive connections. By "sharpening" the logit distribution, the module increases the gap between high-confidence edges and uncertain ones, concentrating the probability mass on a few selected arcs. This operation preemptively discourages violations of the node's physical I/O cardinality before the graph topology is materialized, ensuring a more stable and realistic transition to the discrete sampling stage.

To ensure the reliability of the extraction phase, this projection process is executed iteratively for up to 20 attempts at each sampling step. The system monitors the generated structure and terminates the loop as soon as a valid configuration is identified; however, if no perfectly compliant candidate is found by the 20th iteration, the module discards the graph and the process retries from scratch.

3.2.3 Graph Sampling

Once the guided logits $\tilde{L}_V$ and $\tilde{L}_E$ have been refined by the Strict Projector, the continuous probability distributions must be converted into a discrete graph structure $G(V, E)$. This process, termed Graph Sampling (illustrated in Figure 3.13), is the final stage of the denoising iteration where the latent design is transformed into a concrete industrial layout.

Figure 3.13: Input-output architecture of the Graph Sampling block as integrated within the Generative Pipeline.

For the edges, the system applies the Softmax function to the projected logits $\tilde{L}_{E,ij,k}$ calculating the probability for each class.

The conversion follows a stochastic sampling approach. For each potential connection, the final existence of an edge e_{ij} is determined by drawing a sample from the resulting categorical distribution $\mathbf{P}_{E,ij} = [P(e_{ij} = 0), P(e_{ij} = 1)]$ using Multinomial Sampling:

$$e_{ij} \sim \text{Categorical}(\mathbf{P}_{E,ij})$$

For the nodes, since the inventory V_T is fixed, the system performs a categorical assignment based on the guided node logits $\tilde{L}_V$. Each node is mapped to its final functional identity — Machine, Buffer, Assembly, or Disassembly — to ensure that the total counts perfectly match the user-defined template.

The result of this stage is a raw discrete graph $G(V, E)$, which possesses the required components and a performance-optimized topology, serving as the input for the final phases.

3.2.4 Logit-Guided Repair Strategy

Although the Graph Sampling produces a concrete layout, the resulting graph $G(V, E)$ is still considered a "raw" output. Due to the stochastic nature of the sampling process, the materialized graph may contain topological inconsistencies, such as isolated nodes or interrupted flows. To resolve these issues, a Logit-Guided Repair Strategy is implemented, as illustrated in the block diagram of Figure 3.3.

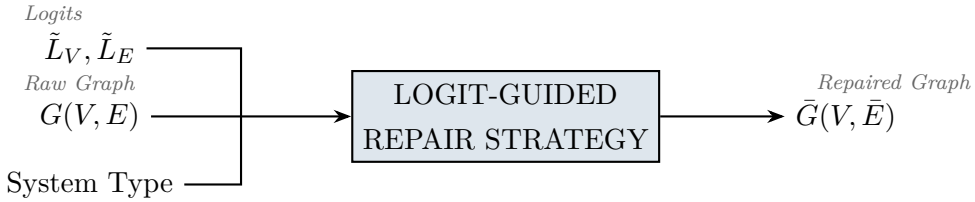


Figure 3.14: Input-output architecture of the Logit-Guided Repair Strategy block as integrated within the Generative Pipeline.

The repairer takes two primary inputs: the raw discrete graph $G(V, E)$ and the guided edge logits $\tilde{L}_E$ produced during the diffusion process. By utilizing the original logits, the repairer ensures that any structural modification maintains the highest possible affinity with the underlying industrial patterns learned by the model. Instead of adding or removing connections randomly, the strategy prioritizes the links with the highest confidence scores (probabilities) to bridge gaps or resolve violations.

The final output is a refined graph $\bar{G}(V, \bar{E})$, where the connectivity has been modified to satisfy the necessary requirements. This corrective action is specifically tailored to the graph typology considered, ensuring the coherence with the specific type constraints.

The repair algorithm operates through a sequential, multi-stage refinement process designed to ensure the structural validity of the generated layout. The first three stages are universal — applied to all four layout topologies — and work through a purely subtractive logic by pruning any incorrect or illegal connections predicted by the model. The final stage, instead, is tailored specifically for closed-loop systems, shifting to a constructive approach to ensure the global connectivity required for recirculation.

1. **Structural Sanitization:** the first pass enforces inviolable "hard zeros" on the adjacency matrix E. Regardless of the model's predictions, the algorithm explicitly zeroes out connections that are physically impossible in the domain logic:
   - **Self-Loops:** the diagonal terms are forced to zero ($e_{ii} \leftarrow 0$).
   - **Buffer-to-Buffer Constraints:** a binary mask identifies all Buffer nodes, and any edge corresponding to the Buffer-to-Buffer connections immediately removing them ($e_{Buffer \to Buffer} \leftarrow 0$), ensuring structural validity by preventing consecutive storage nodes in the layout flow.
2. **Probabilistic Conflict Resolution:** since material flow must be strictly unidirectional, any bidirectional edge pair ($i \leftrightarrow j$) represents a conflict. The system preserves the direction with the higher likelihood by comparing the logits for the "edge present" class ($k = 1$) and prunes the weaker one:

   $$e_{ji} \leftarrow 0 \quad \text{if } \tilde{L}_{E,ij,1} > \tilde{L}_{E,ji,1}, \quad \text{else } e_{ij} \leftarrow 0$$

   This ensures the final direction aligns with the neural network's strongest preference.
3. **Logit-Ranked Priority Pruning:** the final step strictly enforces maximum cardinality constraints for each node type (e.g., a Machine is limited to one output). If a node exceeds its threshold m_{max}, the algorithm retrieves the raw logit values for class $k = 1$, ranks them by confidence, and performs a Top-m selection:

   $$\mathcal{N}_{kept}(i) = \text{top-}m_{max}\{j \mid e_{ij} = 1\} \text{ ordered by } \tilde{L}_{E,ij,1}$$

   All edges not belonging to this high-confidence set are pruned ($e_{ij} \leftarrow 0$), ensuring local I/O constraints are met while retaining the connections the model considers most probable.
4. **Global Topology Reconstruction (Closed Loop Targets):** for Closed systems, satisfying local constraints is insufficient to guarantee global recirculation: a graph may be locally valid yet fragmented into disjoint sub-systems. To address this, the module applies an active reconstruction strategy based on Strongly Connected Components (SCCs).

The algorithm first decomposes the graph into its constitutive islands $\mathcal{S} = \{S_1, S_2, \ldots, S_z\}$. If fragmentation is detected ($z > 1$), global connectivity is restored by imposing a ring topology over the components ($S_1 \to S_2 \to \cdots \to S_1$). Crucially, the specific bridging edge (u^*, v^*) linking a source component S_i to a target component S_{i+1} is selected via a constrained optimization over the model's output. The algorithm selects the best bridging edge (u^*, v^*) by scanning all potential connections between the two components. It chooses the link with the highest logit score for the "edge present" class ($k = 1$), ensuring the new connection aligns with the model's strongest prediction:

$$(u^*, v^*) = \underset{u \in S_i,\, v \in S_{i+1}}{\operatorname{argmax}} \left(\tilde{L}_{E,uv,1} \mid \Phi_{\text{valid}}(u, v) \right)$$

where the validity function Φ_{valid} enforces domain rules (e.g., prohibiting Buffer-to-Buffer connections) and strict system consistency constraints for Buffer nodes.

To ensure the repair process does not alter the fundamental functional logic of the storage units, the following eligibility criteria are applied based on in-degree (d_{in}), the number of incoming edges, and out-degree (d_{out}), the number of outgoing edges:

- **Source Eligibility:** a Buffer node u can serve as the source of a bridge only if it possesses redundant input capacity ($d_{in}(u) > 1$).
- **Target Eligibility:** a Buffer node v can serve as the target of a bridge only if it is already active within the flow ($d_{in}(v) > 0$).

This criterion ensures that the stitching of disjoint islands restores the strongest "suppressed" signal latent in the diffusion model without creating deadlocks or invalidating the pre-existing node configurations.

3.2.5 Hierarchical Validation

The final stage acts as a strict quality filter (Figure 3.15). To save computing power, the repaired graph $\bar{G}(V, \bar{E})$ undergoes four checks ordered by increasing complexity. Since some constraints are type-dependent, the System Type is provided as an input to tailor the validation to the specific plant architecture. Only graphs that pass every test are accepted as valid samples $\hat{G}(V, \hat{E})$.

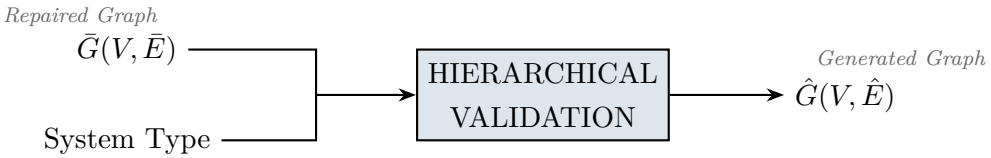


Figure 3.15: Input-output architecture of the Hierarchical Validation block as integrated within the Generative Pipeline.

The hierarchical validation pipeline is composed by the following sequential steps:

1. **Local Constraint Check:** the first filter verifies that every individual node respects the strict I/O cardinality rules defined by the dataset formalism. The check fails if any node violates the atomic definition of its component type:
   - **Machines:** must have exactly 1 input and 1 output.
   - **Buffers:** must not connect directly to other Buffers (bipartite constraint) and cannot have self-loops
   - **Assembly:** must have exactly 2 inputs and 1 output.
   - **Disassembly:** must have exactly 1 input and 2 outputs.

   Failure at this stage indicates a fundamental structural error in the component definition.
2. **System Type Verification:** this step validates the global flow boundaries against the user-requested System Type. By analyzing the in-degree (d_{in}) and out-degree (d_{out}) of all Buffer nodes, the algorithm identifies system entry and exit points:
   - **Closed Systems:** the validator asserts that the count of Input Sources and Output Sinks is exactly zero.
   - **Open Systems:** the validator guarantees the presence of at least one Input Source and one Output Sink to allow material flow from the external environment.

- **Input-Only Systems:** the validator checks for the presence of at least one Input Source while ensuring that no Output Sinks exist within the topology.
- **Output-Only Systems:** the validator ensures the presence of at least one Output Sink while strictly prohibiting any Input Sources.

3. **Connectivity Check:** This filter ensures the graph forms a single cohesive unit. It explicitly rejects layouts containing isolated components ("islands") or nodes with a total degree of zero, as these represent non-functional plant segments that cannot participate in the global production process.
4. **Topological Reachability:** the final and most computationally intensive check verifies the actual material flow feasibility using graph traversal algorithms:
   - **Open Systems**: it verifies the "Backward Reachability", where every node in the graph must be reachable starting from the set of input sources, ensuring no deadlock zones exist.
   - **Closed Systems**: it checks for strong connectivity asserting that every node is part of a valid cycle and can reach every other node in the network, preventing internal accumulation points or starvation cycles.
   - **Input-Only Systems**: it verifies the "Backward Reachability", ensuring every node is reachable from the input sources to guarantee global system integration.
   - **Output-Only Systems**: it verifies the "Forward Reachability", confirming that every node in the network maintains a directed path toward at least one output sink.

Only layouts that survive this filtering pipeline are passed to the discrete event simulator for performance evaluation. This rigorous selection mechanism ensures that the computational budget of the simulation phase is invested exclusively in structurally valid designs, maximizing the overall efficiency of the experimental campaign.

4

Experimental Setup

This chapter details the operational environment and the procedural protocols adopted to validate the proposed Discrete Diffusion framework. The focus shifts from the theoretical formulation to the practical implementation, describing the computational infrastructure, the generation of the synthetic training dataset, and the specific configuration of the generative model. Furthermore, the technical evaluation metrics and the optimization strategy are defined to ensure the reproducibility and robustness of the experimental results.

4.1 Computational Environment

The experimental framework is implemented in Python, using a modular pipeline that connects deep learning with Discrete Event Simulation. The core generative process and graph-based tensor operations are managed via PyTorch and PyTorch Geometric [33, 14]. For topological analysis — such as the repair strategy and connectivity checks — the system relies on NetworkX [16]. To finalize the process, a custom headless Discrete Event Simulator evaluates the generated layouts, providing the performance data needed to validate each candidate without rendering overhead.

All experiments were conducted on a dedicated workstation to ensure reproducibility. The training and batched inference of the diffusion model were accelerated on an NVIDIA GeForce GTX 1650 with 4 GB of VRAM, utilizing CUDA 11.8 for tensor parallelization. The CPU-bound simulation and graph repair tasks were distributed across the 12 logical threads of an Intel Core processor (6 physical cores) using multi-processing pools.

4.2 Dataset Preparation

To provide the model with a clear understanding of the relationship between layout design and efficiency, a synthetic dataset of labeled industrial layouts was constructed. It is crucial to distinguish this phase from the final inference task: while the ultimate goal is to generate optimized solutions, the dataset preparation aims to explore the widest possible operational space. Consequently, the generation process here is unconditioned regarding performance, capturing a broad spectrum of both high-performing and suboptimal topologies. This variety enables the model to learn global correlations between connectivity patterns and operational efficiency.

The dataset dimension was standardized by defining a fixed number of samples for each system category, denoted as $N_{dataset} = 250$. This specific value was chosen through a careful trade-off analysis: reducing this number would result in a considerable decrease in the variety of graphs examined by the model during the training phase, potentially limiting its ability to generalize. Conversely, excessively increasing the sample count risks generating enough graphs to cover nearly the entire spectrum of the solution space. Given that the number of possible configurations is limited by the high density of applied industrial constraints, a cautious approach was taken to avoid over-saturating the dataset, as the total set of feasible solutions remains bounded.

The workflow proceeds through four sequential stages: Stochastic Generation, Logit-Guided Repair Strategy, Strict Validation, and Performance Labeling.

4.2.1 Stochastic Generation

The primary objective of the generation engine is to navigate the vast landscape of possible industrial configurations to produce diverse and varied layout alternatives. To ensure that the model explores the full breadth of the solution space rather than converging toward a single deterministic output, the process is initialized with maximum stochasticity. Specifically, the adjacency matrix starts in a state of maximum entropy, where each edge is sampled from a uniform distribution. This randomized initialization, combined with the inherent stochasticity of the diffusion paths, allows the engine to discover multiple, structurally distinct solutions for the same set of initial constraints.

This exploration is governed by a pre-trained discrete diffusion architecture. The

model was initially developed based upon a baseline dataset of graphs that adhere only to local structural constraints — specifically, the I/O connectivity of individual nodes — regardless of global semantic restrictions or specific system topologies. This choice ensures a high degree of generalization across all various system types, preventing the model from developing biases toward fixed organizational rules.

The implementation of this strategy follows a strict Constraint-to-Graph approach, designed to transform abstract user requirements into structurally valid layouts. This framework utilizes a hybrid strategy where nodes serve as fixed elements, while their connections form the variable component that the model must determine. The process begins by mapping the user's requested inventory I onto the graph; these components are treated as constrained variables, "anchored" at the start and remaining unchanged throughout the process. By treating the hardware inventory as an immutable structural foundation, the model's generative capacity is focused exclusively on discovering the optimal connectivity pattern.

To transform the initial uncertainty of the random graph into a functional layout, the target system type is converted into a dense semantic embedding. During the reverse diffusion process, this vector is injected into the neural architecture to guide the attention mechanism. In practice, the embedding adjusts the model's output scores — the logits — at each step, pushing the probability toward edge configurations that fit the requested system logic. Through this guided refinement, the model gradually organizes the initial noise into a structured directed graph that connects the anchored nodes while strictly following the target industrial typology.

4.2.2 Logit-Guided Repair Strategy

Despite the directional guidance provided by the semantic embedding, the inherent stochasticity of the diffusion process does not inherently guarantee the structural validity of every generated output. Furthermore, while the model is biased toward a specific system type through conditioning, it cannot strictly enforce hard global logical constraints during the iterative denoising phase. Consequently, ensuring these topological properties requires a subsequent verification and refinement stage.

To correct local inconsistencies and satisfy global requirements without compromising the generative quality, the pipeline incorporates a Logit-Guided Repair mechanism. This algorithm operates through a structured three-stage protocol designed to reconcile the model's probabilistic predictions with the rigid constraints

of industrial layouts:

1. **Domain Logic Sanitization:** before addressing specific connectivity, the algorithm removes patterns that violate fundamental domain rules. Specifically, it enforces the removal of self-loops and prohibits direct Buffer-to-Buffer connections.
2. **Local Constraint Enforcement:** the core of the repair process addresses the local inconsistencies. The algorithm iterates through every node to validate its I/O cardinality against its functional definition. If a violation is detected, the algorithm checks the model's confidence scores to find the most likely missing connection. This ensures the fix follows the model's logic instead of just adding a random link.
3. **Topology-Specific Imposition:** finally, global structural rules are enforced based on the target system type. This step is particularly vital for Closed-loop systems, where the algorithm applies a dedicated repair logic to address "sinks" and "sources." The repairer identifies Buffer nodes lacking either outgoing or incoming connections and establishes the necessary feedback loops to resolve these topological discontinuities. By querying the model's logits, the algorithm identifies the most probable processing nodes to link with these buffers, ensuring that new connections align with the model's highest confidence predictions. While this refinement effectively repairs local flow interruptions, its primary goal is to ensure that local paths are recirculating, thereby significantly increasing the probability that the layout will satisfy the final Strict Validation as a single, cohesive unit.

4.2.3 Strict Validation

Following the repair phase, every candidate graph undergoes a rigorous hierarchical validation. While the repair mechanism effectively corrects the inconsistencies, this phase operates in a binary mode: any graph failing even a single validation criterion is definitively discarded and regenerated. This ensures that the final dataset contains only perfect, simulation-ready layouts.

The validation protocol enforces compliance across four distinct logical dimensions:

- **Inventory Integrity Check**: as a primary sanity check, the algorithm verifies that the final node distribution strictly matches the requested inventory vector. Although the "Node Pinning" strategy is designed to ensure this by construction, this step acts as a failsafe to detect any potential corruption in the node feature tensor during the generation or repair steps.
- **Exact Local Constraint Compliance**: the graph is checked against strict I/O rules for each component type. A single deviation in these local connection results in immediate rejection.
- **Global Connectivity Verification**: the algorithm ensures the system constitutes a single cohesive component, prohibiting the existence of isolated subgraphs or detached islands of nodes. This is critical for simulation, as disconnected components would result in undefined flow metrics.
- **Semantic Topology Validation**: finally, the graph is evaluated against the rules of the requested system type. This ensures that the dataset labels (system type) remain perfectly consistent with the actual graph topology.

Only samples that simultaneously satisfy all four criteria are admitted into the final dataset. This strict filtering guarantees that the subsequent performance labeling phase is conducted exclusively on operational layouts, eliminating noise derived from structural invalidity.

4.2.4 Performance Labeling

Validated layouts are subjected to performance evaluation through a Discrete Event Simulation (DES). This simulation measures two key metrics: System Throughput and Total Energy Consumption, capturing the system dynamics to provide a high-fidelity representation of its actual behavior under realistic operating conditions. Further details regarding the underlying DES engine and its logic are provided in Appendix A.

To derive a consistent training signal, these heterogeneous physical metrics are integrated according to the framework established in Section 2.4. Specifically, the final score S is obtained by balancing normalized results through the weighting coefficients w_X and w_E. These parameters function as tunable preferences, dictating the specific performance profile the model should internalize during learning. As a result, this approach allows the framework to adapt to different strategic goals without

requiring any modifications to the underlying generative or simulation pipelines.

4.3 Evaluation Metrics

To quantitatively assess the proposed framework, a technical evaluation protocol was established. Unlike industrial metrics that focus on plant efficiency, these metrics are specifically designed to evaluate the generative fidelity of the model. The primary objective is to verify the network's capacity to generate layouts that are not only structurally valid and diverse but also consistent with the requested performance targets. To this end, the assessment relies on the following four technical indicators:

- **Validity Rate**: this metric measures the model's adherence to the manufacturing domain's fundamental rules by calculating the ratio of generated layouts that satisfy all topological and operational constraints. It is formally defined as:

$$\text{Validity Rate} = \frac{1}{n_{gen}} \sum_{k=1}^{n_{gen}} \mathbb{1}(\text{valid}(G^{(k)}))$$

  where n_{gen} is the total number of generation attempts and $G^{(k)}$ represents the k-th synthesized layout. The logical function valid$(\cdot)$ verifies essential industrial properties while the indicator function $\mathbb{1}(\cdot)$ assigns a value of 1 to compliant graphs and 0 to those violating any constraints.
  A high Validity Rate demonstrates that the network has successfully mastered the underlying structural logic, producing architectures that are both statistically consistent with the learned distribution and strictly compliant with all physical and logical requirements.
- **Uniqueness**: this metric evaluates the structural diversity within a generated batch by identifying the percentage of topologically distinct layouts. It is defined as the ratio of unique graphs to the total number of generation attempts:

$$\text{Uniqueness} = \frac{|\{\text{WL}(G^{(k)}) : G^{(k)} \in \mathcal{G}_{valid}\}|}{n_{gen}}$$

  where $\mathcal{G}_{valid}$ is the subset of layouts that passed the validity check. The operator $|\{\cdot\}|$ calculates the cardinality of the set of unique structural hashes produced by the Weisfeiler-Lehman (WL) algorithm. By using this hashing technique, the metric ensures that variety is measured based on the actual

connectivity and logic of the plant rather than the arbitrary indexing of the nodes.

A high Uniqueness score indicates that the model possesses a wide creative range and can effectively explore different architectural combinations without falling into "mode collapse" — a common generative failure where the network repetitively produces the same output.

- **Novelty**: this metric quantifies the model's ability to synthesize structural configurations that are distinct from those already present in the source dataset. It is calculated as the proportion of valid graphs that do not appear in the training set:

$$\text{Novelty} = \frac{|\{G^{(k)} \in \mathcal{G}_{valid} : \text{WL}(G^{(k)}) \notin \mathcal{H}_{train}\}|}{n_{gen}}$$

where $\mathcal{H}_{train}$ represents the complete set of unique structural hashes extracted from the training data. By identifying valid layouts $G^{(k)}$ whose topological hash $\text{WL}(\cdot)$ is absent from the training set, the metric serves as a primary indicator for detecting overfitting and memorization [39].

Low values signal that the network is simply reproducing training samples, whereas high Novelty confirms that the model has successfully learned the underlying generative rules required to create original factory environments.

- **MAD Score (Mean Absolute Deviation)**: this metric represents the conditioning accuracy of the Classifier-Free Guidance by measuring how precisely the model respects the requested performance targets. It is defined as:

$$\text{MAD} = \frac{1}{n_{valid}} \sum_{k=1}^{n_{valid}} |\bar{c}^{(k)} - S(G^{(k)})|$$

.

In this context, n_{valid} denotes the number of valid samples produced, while $\bar{c}^{(k)}$ is the specific Target Score used as input conditioning for the model. The term $S(G^{(k)})$ represents the actual performance score obtained via Discrete Event Simulation of the generated layout $G^{(k)}$. This value quantifies the model's precision in translating abstract industrial requirements into concrete structural architectures; a lower deviation indicates a higher fidelity in the conditioning process.

4.4 Hyperparameter Configuration

The model's parameter configuration was defined through a hybrid approach, integrating theoretical principles, established literature benchmarks, and empirical testing to optimize both structural design and generative logic. These specifications establish the core framework of the model, which remains invariant throughout both the training and inference phases. This static foundation is crucial for ensuring uniform behavior across different system types and for providing a stable baseline for subsequent performance evaluations. To provide a structured overview, these fixed specifications are organized into three functional areas: Training Dynamics, Network Architecture, and the Diffusion Process.

4.4.1 Training Dynamics

This category defines the hyperparameters governing the optimization loop and the stochastic regularization strategies employed to stabilize the learning trajectory.

- **Training Epochs**
  Based on an initial exploratory phase, the experimental design targeted three strategic training intervals: $\{200, 600, 1000\}$ epochs. This range represents a deviation from standard diffusion frameworks, which typically require much longer schedules — often exceeding thousands of epochs — to ensure full convergence [37]. However, preliminary analysis identified this reduced window as the optimal trade-off for these specific dataset constraints.
  Training for fewer than 200 epochs leads to underfitting, where the model fails to learn the complex relationships between nodes, resulting in inconsistent and invalid samples. Conversely, extending the training beyond 1000 epochs increases the risk of overfitting. Given the limited dataset size ($N_{dataset} = 250$), prolonged optimization encourages the model to memorize specific training examples rather than learning the general rules of the distribution, which is detrimental to the novelty of the generated graphs.
- **Batch Size (B)**
  The batch size was tested within the set $B \in \{8, 16, 32\}$. This choice prioritizes the benefits of "Small-Batch" training, as described by Keskar et al. [21]. According to their research, Large-Batch optimization often converges toward

"sharp minima" — narrow regions in the loss landscape where the model's accuracy can drop sharply with even small changes. In contrast, smaller batches introduce helpful "noise" into the training process. This noise acts as a natural stabilizer, guiding the model toward "flat minima," which result in better robustness and superior generalization.

As a result, the upper limit was set at $B = 32$, following the findings of Masters and Luschi [30], who identified the $[2, 32]$ range as optimal for maximizing performance. Meanwhile, the lower limit of $B = 8$ was set to ensure training stability. Dropping below this threshold would make the training updates too volatile and unstable, as the model would be trying to learn from too few examples at once.

- **Learning Rate (η)**

  The optimization process is controlled by a fixed learning rate $\eta = 2 \times 10^{-4}$. This setting strictly follows the architectural benchmarks established by Vignac et al. [37] in the DiGress framework. In Discrete Denoising Diffusion, where the model works with categorical variables — such as discrete machine types and connections — rather than continuous noise, the training process is naturally more unstable. Vignac et al. [37] demonstrated that a learning rate of this magnitude offers the best balance between stability and speed for graph transformers. It allows for efficient updates in the early stages of training while preventing the erratic oscillations often caused by higher rates (such as 10^{-3}) when learning discrete structures.

- **Conditioning Dropout (p_{uncond})**

  To enable the Classifier-Free Guidance mechanism, a stochastic masking strategy is adopted during the training phase. Specifically, the conditioning signal c is dropped with a probability $p_{uncond} = 0.2$. This value was selected based on the findings of Ho and Salimans [19], who demonstrated that a 20% masking rate is optimal for teaching the model both conditional and unconditional distributions.

  Beyond enabling guidance, this mechanism acts as a vital regularizer. By forcing the network to occasionally reconstruct the graph structure without any semantic hints, it prevents the model from relying too heavily on labels. This discourages memorization and helps preserve the model's ability to generalize, which is especially important given the limited size of the dataset.

- **Adaptive Optimization** (β_1, β_2)
  Gradient optimization is managed by the Adam algorithm, using exponential decay rates for moment estimates set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. This configuration follows the standard specifications established by Kingma and Ba [22]. In their research, the authors demonstrated that this specific set of hyperparameters is highly robust for non-stationary goals and problems with sparse gradients, which are common in high-dimensional generative tasks. Consequently, these values were kept fixed to ensure stable optimization and to separate the training stability from the randomness of the diffusion process, removing the need for further tuning of these specific terms.

Sensitivity Analysis and Parameter Selection

Building upon the discrete candidate sets identified in the preceding Section 4.4.1, a Full Factorial Design was executed to empirically determine the optimal combination of Batch Size and Training Epochs. The optimization process was strictly guided by the technical evaluation protocol established in Section 4.3, specifically leveraging the multi-objective framework to balance structural integrity, generative diversity, and target adherence.

The quantitative outcomes of this analysis are visualized in Figure 4.1. The heatmaps illustrate the impact of training dynamics on the four performance indicators, highlighting the necessity of a conjoint analysis to interpret the trade-offs between generative diversity and conditioning adherence.

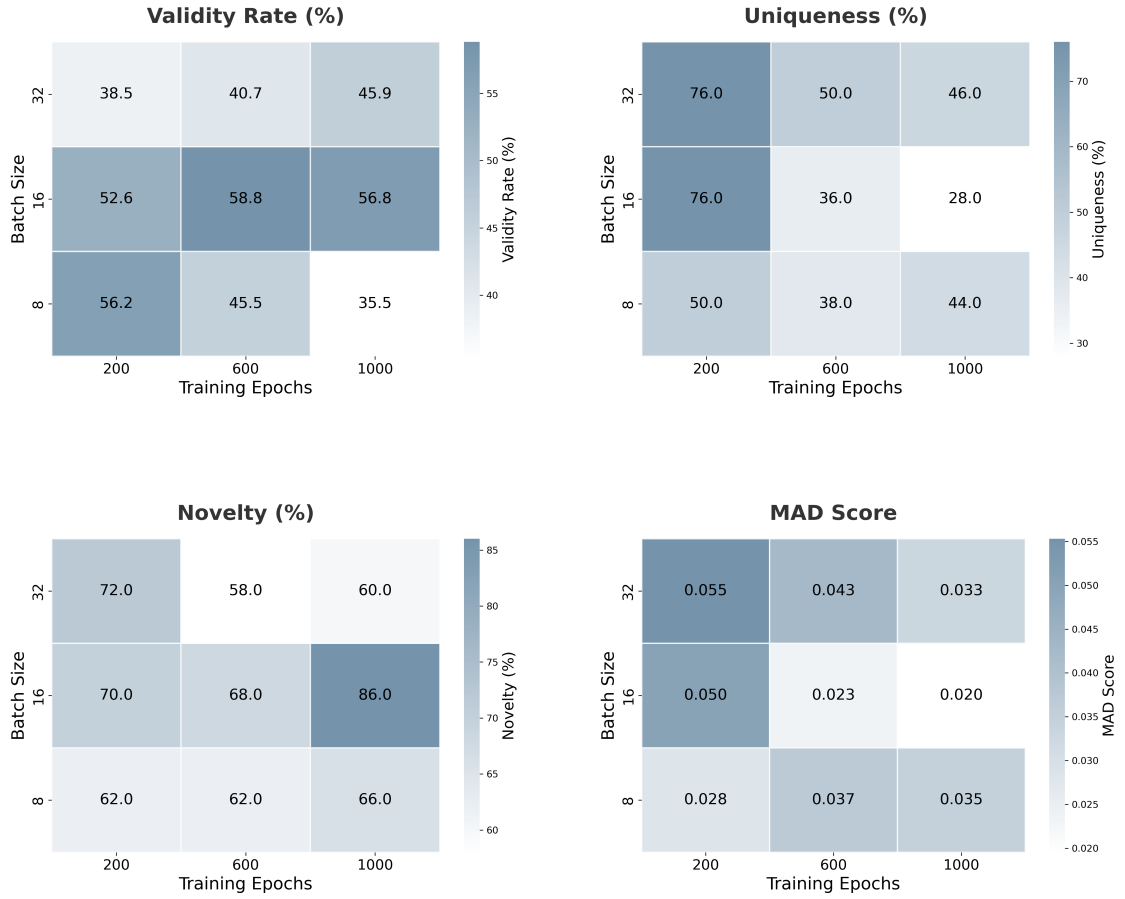


Figure 4.1: Sensitivity analysis of the training hyperparameters, illustrating the comparative impact of Batch Size and Training Epochs across Validity Rate, Uniqueness, Novelty, and MAD Score

A preliminary inspection of the heatmaps reveals clear trends in model behavior across different configurations. The Validity Rate remains relatively stable as the number of epochs increases, especially for Batch Sizes of 16 and 32. In contrast, Uniqueness shows a significant downward trend as training progresses; this decline indicates a move toward mode collapse, where the model begins to lose its ability to generate diverse layouts. Novelty does not follow a specific pattern, showing fluctuations throughout the training process. The MAD Score follows a trend similar to Uniqueness but with a positive outcome: the deviation seems to decrease as more epochs are completed, showing that the model learns to follow the requested targets more accurately. Crucially, the stable trends seen with Batch Sizes 16 and 32 are not

shared by Batch Size 8, which shows highly unstable behavior. This volatility is due to the small batch size, which causes high variance in the gradients and prevents the model from reaching a stable convergence. Consequently, configurations using Batch Size 8 are considered unreliable and have been excluded from the final selection.

The selection of the optimal configuration is based on a multi-objective evaluation that prioritizes structural robustness and conditioning accuracy. While early stopping (e.g., 200 epochs) maximizes diversity, it does not guarantee sufficient generative reliability. This limitation is clear when looking at the Batch Size 32 setup at 200 epochs. Although it achieves high Uniqueness (76.0%), it suffers from a high MAD Score (0.055) and a suboptimal Validity Rate (38.5%). This suggests that while the model generates diverse solutions, it struggles to respect topological constraints and performance targets.

In contrast, using a Batch Size of 16 for 600 epochs provides the best operational balance. This setup achieves a dominant Validity Rate of 58.8% — the highest among stable configurations — and a remarkably low MAD Score of 0.023. This significant reduction in deviation compared to the 200-epoch baseline shows that longer training allows the model to better internalize the relationship between topology and performance.

As a result, the combination of Batch Size 16 and 600 epochs was chosen as the optimal operating point. Although this choice leads to lower Uniqueness (36.0%) than early-stage models, the trade-off is strategic. The selection prioritizes valid and strictly conditioned layouts over maximum entropy. By maximizing the Validity Rate and minimizing the MAD, the framework becomes less dependent on the repair pipeline, ensuring the output is not only novel (68.0%) but also structurally robust.

For this factorial design, the target was restricted exclusively to Throughput. This isolation strategy decouples the assessment from multi-objective trade-offs, focusing on the model's ability to map topological features to flow dynamics. Operationally, this was enforced by setting the weights to $w_X = 1$ and $w_E = 0$, creating a single-objective landscape. Furthermore, due to the significant computational time required for the analysis, the investigation was narrowed to the Open system topology and validated on a single representative sample.

4.4.2 Network Architecture

The architectural hyperparameters were calibrated to achieve an optimal balance between representational capacity and structural generalization.

- **Embedding Dimension (d_{emb})**
  The dimensionality for temporal and score projections was fixed at $d_{emb} = 16$. This choice follows the architectural design of established conditional diffusion models, such as DiffWave [24], where scalar conditioning signals are projected into compact, low-dimensional embeddings (e.g., 16 or 128) before being integrated into the main hidden layers. By setting d_{emb} to a quarter of the model's hidden dimension ($d_{model} = 64$), the framework ensures that the global context provides a clear directional bias without saturating the latent space. This prevents the model from disregarding the local node features, maintaining a balanced influence between the target performance score and the structural graph topology.
- **Hidden Dimension (d_{model})**
  The latent dimension size was fixed at $d_{model} = 64$, balancing representational power with the ability to generalize through a "dual-bound" logic. From an upper-bound perspective, limiting the dimension is essential due to the small sample size ($N_{dataset} = 250$); adopting standard high-capacity values (e.g., $d_{model} = 256$) would create excessive trainable parameters, leading to the memorization of specific training examples rather than true structural learning [37, 20]. Conversely, the lower bound is dictated by the requirements of the Multi-Head Attention mechanism. As discussed by Dwivedi & Bresson [11], maintaining a sufficient subspace dimension for each head (d_k) is a prerequisite for learning complex relationships between nodes; reducing d_{model} below 64 would severely weaken the expressivity of the individual Attention Heads.
- **Number of Attention Heads (n_{head})**
  The attention mechanism is parallelized across $n_{head} = 4$ independent heads, defining the number of distinct "perspectives" the model can adopt simultaneously when analyzing the topological neighborhood of a node. In a Graph Transformer architecture, this multi-head approach allows the network to learn and attend to different types of structural relationships in parallel, such as identifying upstream material sources versus downstream processing units.

The choice of 4 heads is strategically linked to the model's hidden dimension; by setting $n_{head} = 4$, each head operates on a subspace of $d_k = 16$. According to Dwivedi & Bresson [11], maintaining an adequate subspace dimension for each head is a known prerequisite for the effectiveness of Graph Transformers, as it ensures each head has sufficient numerical expressivity to represent complex relationships. Furthermore, following the benchmarks discussed by Shi et al. [34], this setup helps manage the anisotropy of graph structures, allowing the model to assign different importance scores to neighbors based on their specific functional roles. Finally, as observed in frameworks dealing with discrete graph generation like DiGress [37], maintaining a moderate number of heads for a dimension of $d_{model} = 64$ serves as a form of architectural regularization, preventing the attention mechanism from over-parameterizing the relationships within the limited training set ($N_{dataset} = 250$) and guiding the model toward more stable, global topological patterns.

- **Number of Layers (Z)**
  The model backbone is structured with $Z = 2$ sequential Graph Transformer layers to balance structural expressivity with architectural stability. This shallow configuration is primarily intended to mitigate the over-smoothing effect, a common phenomenon in Graph Neural Networks where excessive depth causes node representations to converge into indistinguishable values, leading to the loss of functional identity between categories such as Machines and Buffers [26]. By limiting the depth to two layers, the framework constrains the receptive field — the range within which a node aggregates information — to a 2-hop radius. This is strategically sufficient because industrial layout logic is primarily defined by local dependencies: a functional unit's validity depends on its immediate neighbors (1-hop) and the nodes directly connected to them (2-hop), such as the standard Buffer-Machine-Buffer sequence. This localized focus allows the model to capture the essential connectivity rules without being "confused" by distant, irrelevant nodes, which could otherwise lead to signal dilution or over-smoothing [17]. Furthermore, this lightweight setup is essential for maintaining computational tractability during the reverse diffusion phase; since the model must perform $T = 500$ sequential inferences to synthesize a single layout, a reduced number of layers prevents excessive overhead without compromising the ability to reconstruct valid and optimized

topologies from discrete noise.

4.4.3 Diffusion Process

The configuration of the Diffusion Process determines the temporal dynamics of the generative mechanism, specifically defining the schedule by which discrete noise is injected into the graph structure and the trajectory used to subsequently remove it. These settings follow established benchmarks for discrete state-space diffusion, balancing the required number of steps with the need to preserve the original design logic.

- **Diffusion Horizon (T)**
  The forward and reverse processes are discretized into $T = 500$ time steps. While major works on diffusion models typically utilize a baseline of $T = 1000$ steps to ensure stability, using so many steps makes the generation process significantly slower [18, 37].
  The reduction to $T = 500$ is made possible by leveraging the efficiency of the Cosine Noise Schedule [32]. Standard linear schedules often destroy information too abruptly — particularly toward the end of the forward process—forcing the model to use more steps to learn the reverse transition. In contrast, the cosine mapping "softens" the noise injection at the boundaries of the horizon, maintaining a higher Signal-to-Noise Ratio (SNR) for a longer duration.
  In this context, the SNR represents the balance between the original industrial logic (the "signal") and the randomness introduced by the diffusion. Under a linear schedule, the SNR drops precipitously, leaving the model to process "information-poor" noise for a large portion of the cycle. The cosine schedule ensures the SNR decreases more gradually, meaning that even at intermediate steps, significant structural information remains discernible. This improved signal preservation allows the model to approximate the data distribution with high fidelity using half the sampling trajectory of standard baselines. Consequently, $T = 500$ provides an optimal trade-off, doubling the inference speed while ensuring the structural validity of the generated graphs is not compromised.
- **Noise Schedule Offset (s)**
  The noise progression follows a cosine schedule, incorporating a specific offset

to guarantee numerical stability at the boundaries of the time horizon. In accordance with the benchmarks established by Nichol & Dhariwal [32], this offset is set to $s = 0.008$. This parameter is fundamental to avoid mathematical errors at the very start ($t = 0$). It ensures the forward process begins with almost no noise, preventing layout's structure from being destroyed immediately.

The specific configurations for the model architecture and the training process, which remained constant across all experimental campaigns to ensure comparability, are summarized in Appendix B.3.

5

Results Analysis

The experimental evaluation provides a quantitative and qualitative assessment of the framework's ability to synthesize industrial layouts that are both structurally valid and operationally efficient. By systematically comparing the guided generative process against unconditioned baselines, the following sections demonstrate how the model internalizes complex topological rules and navigates the trade-offs between competing industrial objectives. The analysis explores the performance across distinct system topologies, validates the necessity of individual architectural components through ablation studies, and evaluates the sensitivity of the model to various guidance intensities and weighting configurations.

5.1 Results per System-Type

The experimental evaluation covers all four system types: Open, Closed, Input-Only, and Output-Only configurations. The primary objective is to verify the model's capacity to integrate topology-specific structural constraints without compromising operational efficiency.

To quantitatively assess the quality of the generated solutions, the Score function S was tuned to reflect a production-oriented approach. Specifically, the weighting coefficients were fixed at $w_X = 0.75$ for Throughput maximization and $w_E = 0.25$ for Energy Consumption minimization. Beyond these industrial indicators, the model's generative fidelity was measured through the technical metrics detailed in Section 4.3, focusing on its ability to synthesize valid, diverse, and novel structures.

The inference parameters — specifically the Guidance Scale (γ) and the Target Score (c_{target}) — were independently calibrated for each system type. This individual tuning was necessary to align the conditioning signal with the specific operational boundaries and structural limits of each layout. Furthermore, the results presented

hereafter stemmed from an iterative refinement phase of the Composite Loss Function. The balance between objectives was governed by the regularization term λ_{KL} and the semantic weight $\lambda_{semantic}$. A more granular calibration was performed on low-level penalties: λ_{mask} for forbidden connections, λ_{system} for global compliance, and $\lambda_{connectivity}$ for structural cohesion. The specific numerical values assigned to these hyperparameters are detailed in Appendix B.4. This tuning was essential to guide the model toward a solution space that satisfies both high structural validity and superior performance targets.

Finally, to ensure an objective and unbiased statistical analysis, a Uniqueness filtering was applied to each generated dataset before plotting the results. In cases where the model synthesized identical layouts, only one unique instance was retained for the performance distributions. This process prevents the statistical metrics from being artificially biased by repetitive structures, providing a more transparent view of the model's actual generative range and its ability to explore the high-performance frontier.

5.1.1 Open System

The experimental validation on the Open System topology focused on the model's ability to maximize the production rate while minimizing the Energy Consumption under strict guidance. The process followed rigorous configuration parameters: a Target Score of $c_{target} = 0.8$ was set to orient the diffusion trajectory toward the upper bound of the Throughput distribution, while the Guidance Scale was fixed at $\gamma = 1$ to focus the exploration on this optimal region.

Furthermore, the reduced generation time required for Open System configurations allowed for a more extensive experimental campaign compared to other topologies. This computational efficiency enabled a deeper analysis of the underlying generative mechanisms through the processing of five independent datasets for both the unguided and guided categories. Due to the significantly higher computational cost associated with other system types, such an exhaustive investigation was specifically prioritized for this topology as a representative benchmark of the framework's capabilities.

Operational Space Analysis

The effectiveness of this strategy is clearly shown in the performance space in Figure 5.1 where the distribution of the unguided baseline, alongside the original Training dataset distribution, is compared with the layouts synthesized by the Guided diffusion model, revealing a significant shift toward regions of higher efficiency.

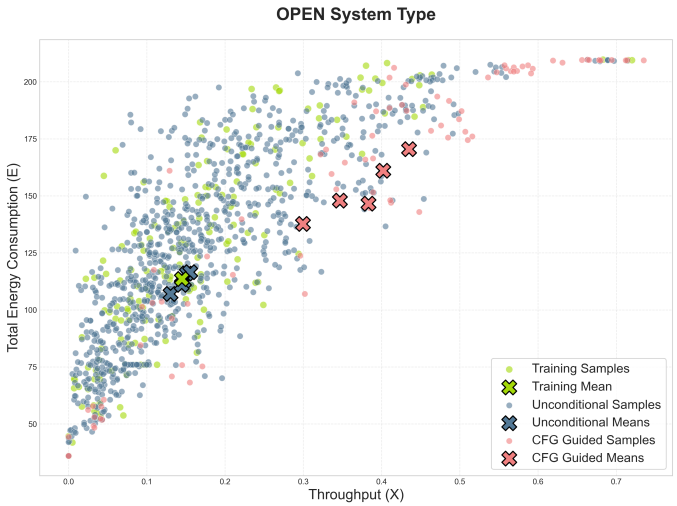


Figure 5.1: Performance distribution in the Throughput-Energy space for the Open topology: a comparative analysis of the original Training dataset, the Unconditional baseline, and CFG Guided configurations.

Notably, the Unconditional generation forms a cluster that aligns closely with the original Training data, demonstrating that the baseline used for comparison correctly captures the underlying statistical distribution of the domain. In contrast, the CFG-driven samples exhibit a significant divergence from the unguided set, shifting primarily toward the upper-right quadrant. This performance transition is further validated by the trajectory of the mean values — highlighted by the "X" markers — which illustrates how the guidance effectively steers the model toward the high-performance frontier.

The gain in Throughput is accompanied by a physiological increase in Total Energy Consumption, accurately reflecting the priority established by the weighting coefficients ($w_X = 0.75, w_E = 0.25$). By emphasizing production, the generative agent identifies layouts that employ active resources more intensively and sustain demanding operational states. These results confirm that the model does not produce physically implausible solutions; rather, it successfully navigates the engineering

trade-off between productivity and energy efficiency, pushing the system toward its optimal operational region.

This optimization trajectory is further clarified in Figure 5.2, where convex hulls and iso-score lines map the shifting operational boundaries. To interpret the logic, it is important to note that the direction of increase for the iso-score curves runs diagonally toward the lower-right corner. This gradient represents the path where the gain in Throughput outweighs the increase in Energy cost.

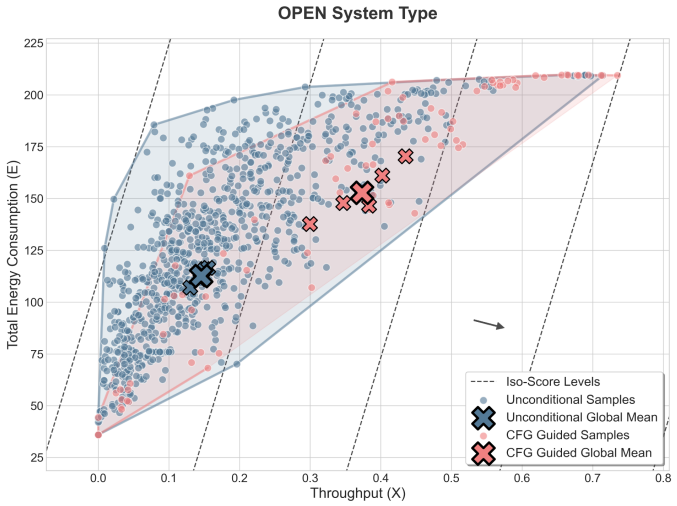


Figure 5.2: Performance distribution in the Throughput-Energy space for the Open topology: a comparative analysis of Unconditional and CFG Guided configurations, displaying individual sample clusters, statistical means, and associated convex hulls over the iso-score functional landscape.

The blue hull illustrates the broad variability of Unconditional generation. Within this boundary, many layouts gravitate toward the ”inefficient complexity” of the upper-left region — where high Energy costs are incurred without a proportional increase in Throughput. Conversely, CFG exploration suppresses these unfavorable directions. The near absence of the Guided set from the upper-left quadrant shows that the model has learned to penalize such paths, concentrating instead on a compact density of elite solutions that maximize industrial yield.

Statistical Performance Analysis

To finalize the analysis of the Open system, Figure 5.3 provides a statistical summary of the performance metrics across the different generative runs. These strip plots, defined over the mean values of the respective datasets, confirm that the shift toward higher performance is not limited to a few outliers but is a consistent property of the entire generated batch.

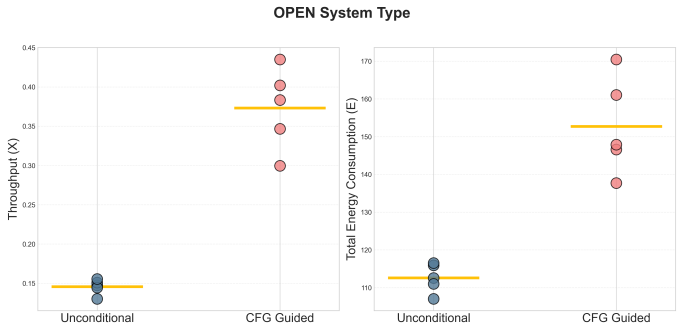


Figure 5.3: Statistical distribution in the Throughput-Energy space for the Open topology: a strip plot comparison between Unconditional and CFG Guided datasets, based on the mean performance values of independent generative runs.

Regarding Throughput, the CFG achieves a substantial mean increase of 156.10% over the baseline (0.373 vs 0.146 units/time), a result validated by a p-value of 7.94×10^{-3}. Energy Consumption reflects the growth in operational intensity with a mean increase of 35.63% (152.74 vs 112.62 units). While the baseline Energy use is often decoupled from output, the Guided model ensures that every Energy increment is effectively transformed into a proportional gain in Throughput, acting as a selective mechanism that permits higher expenditure only when strictly necessary for production.

To ensure that the performance maximization did not compromise the structural quality of the output, the generative metrics reported in Table 5.1 were evaluated.

Metrics	Unconditional	CFG Guided
Validity Rate	$17.12\% \pm 2.34\%$	$65.32\% \pm 4.08\%$
Uniqueness	100.0%	$33.60\% \pm 3.13\%$
Novelty	100.0%	$58.80\% \pm 7.08\%$

Table 5.1: Generative quality metrics for the Open system topology. Values represent the mean and the confidence interval for a significance level $\alpha = 0.10$, calculated across five independent generative runs.

The generative metrics presented in Table 5.1 reveal a significant divergence between the Unconditional approach and CFG generation, a contrast that stems from both the internal sampling dynamics and the specific composition of the Training datasets.

To ensure a robust assessment, these metrics were calculated by averaging the results of five independent generative sessions. By specifying the mean confidence interval, the analysis accounts for statistical variance, ensuring that the reported performance reflects the stable behavior of the models rather than isolated instances or outliers.

A primary observation concerns the impact of the guidance mechanism on the Validity Rate. The CFG samples achieve a remarkably higher success rate of 65.32%, far surpassing the 17.12% recorded by the Unconditional model. Although the latter model was initialized to favor valid configurations, its purely stochastic nature often prevents it from converging toward structures that satisfy all topological constraints. In contrast, the CFG guidance effectively steers the generative process toward "feasible" regions of the latent space, drastically reducing both computational overhead and the number of attempts required to secure valid samples.

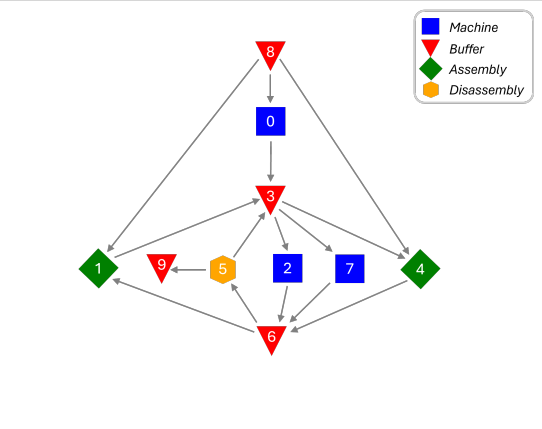
Regarding structural diversity, the 100.0% Uniqueness of the Unconditional model is a direct consequence of its unconstrained, purely random evolutionary mode, where producing identical configurations is statistically improbable. Conversely, the lower Uniqueness in the CFG model (33.60%) reflects the selective pressure of target optimization. As the guidance pushes the generation toward a narrow optimal frontier, the model naturally favors a specific set of elite topological patterns, leading to an expected structural overlap between solutions.

Finally, the differences in Novelty and the baseline Validity Rate are partially

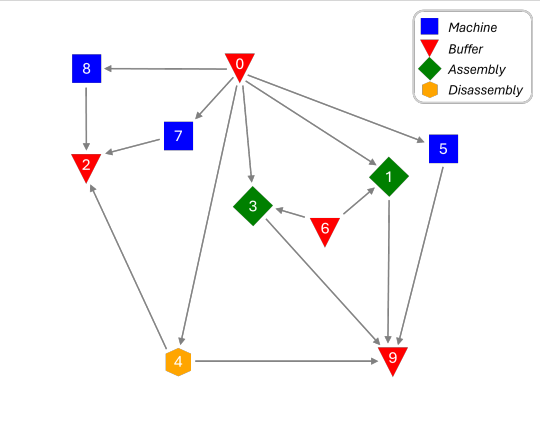
explained by the underlying training strategy. The Unconditional model was trained on a dataset that respected only local constraints to minimize the influence of complex, predefined global structures during the learning phase. Consequently, the Unconditional model produces almost entirely unseen configurations, achieving 100.0% Novelty because it lacks the performance conditioning that, in the guided case, aligns synthetic results with the high-performing trajectories identified during training.

Performance-Driven Architectural Patterns

To gain a better understanding of how the model translates theoretical concepts into practical industrial designs, it is helpful to compare the structural features of two generated layouts: a low-performance solution (Figure 5.4a) and a high-performance alternative (Figure 5.4b).



(a): Layout A.



(b): Layout B.

Performance Metric	Layout A	Layout B
Throughput	0.029	0.699
Total Energy Consumption	103.57	209.27

Figure 5.4: Comparative analysis of Open configurations: visual graph topologies and corresponding Throughput-Energy metrics for low-performance (Layout A) and high-performance (Layout B) samples.

Even though both designs share the same number of nodes, their internal organization and material flow logic are fundamentally different. In the low-performance

layout, the structural impediment is primarily induced by a restrictive buffer configuration that severely constrains material flow. Specifically, buffer 9 serves as the only output for the entire system and is exclusively connected to the Disassembly unit (node 5). Consequently, whenever the Disassembly element is inactive, the entire production flow comes to a halt. This single, restrictive connection acts as a strict limit on the system's capacity, explaining the significantly low Throughput.

In contrast, the high-performance layout (Figure 5.4b) adopts a more balanced and flexible architecture, featuring a symmetrical distribution of two input buffers (0 and 6) and two output buffers (2 and 9). Furthermore, the connectivity toward the output stage increases to seven distinct paths. This configuration enhances routing flexibility by distributing the workload across multiple parallel paths, thereby preventing single-point dependencies and sustaining high Throughput.

The most obvious difference is the logic behind the designs: Layout A is complicated and indirect, while Layout B is simple and straightforward. While the first graph has a messy structure that makes it hard for materials to find a path, the high-performance version focuses on linear patterns. This is seen in the frequent use of "Input → Machine → Output" sequences. In these patterns, a single step — using an Assembly, Disassembly, or Machine node — can move a product directly from the start to the end of the process. By removing unnecessary middle steps and creating these direct paths, the model simplifies the structure and maximizes productivity, showing that it has learned to favor direct efficiency for industrial use.

5.1.2 Closed System

The analysis of the Closed System topology required addressing the inherent complexity of recirculating flows. Unlike linear configurations, where performance follows a cumulative logic, closed loops are susceptible to non-linear phenomena such as congestion or deadlocks. Therefore, the optimization objective focused on balancing high Throughput (cycle frequency) with flow stability and Energy efficiency. To steer the model toward this optimal operational region, a Target Score of $c_{target} = 0.9$ was established, supported by a Guidance Scale of $\gamma = 5$.

Due to the long computational times required for these structures, this analysis compares a single generative set against a random baseline. It is important to note that, because of the inherent randomness of the diffusion process, this specific result might slightly deviate from the model's average output trend. However, it remains

significantly indicative for the purpose of this study, providing a reliable basis for conducting the performance analysis.

Operational Space Analysis

The resulting operational landscape is visualized in Figure 5.5.

Figure 5.5: Performance distribution in the Throughput-Energy space for the Closed topology: a comparative analysis of the original Training dataset, the Unconditional baseline and CFG Guided configurations.

The comparison reveals a sharp distinction between the baseline distributions and the Guided output. The Unconditional generation aligns closely with the original Training data, confirming its ability to properly capture the statistical distribution of the learning domain. This baseline, however, reveals a significant efficiency gap: although random configurations occasionally reach high Throughput, they do so at the cost of disproportionate Energy Consumption, failing to find a sustainable equilibrium.

Conversely, the CFG samples cluster within a specific region of the performance space, shifting the focus toward a more balanced operational profile. This clustering indicates that the model, by prioritizing stringent structural validity and robust connectivity, intentionally gravitates toward ”conservative” yet functionally stable architectures. As a result, while these configurations may not reach the peak intensities seen in the Training set, they ensure superior plant cohesion and functional reliability, successfully navigating the complex constraints inherent to recirculating

systems.

The structural impact of this optimization is further detailed in the correlation analysis (Figure 5.6).

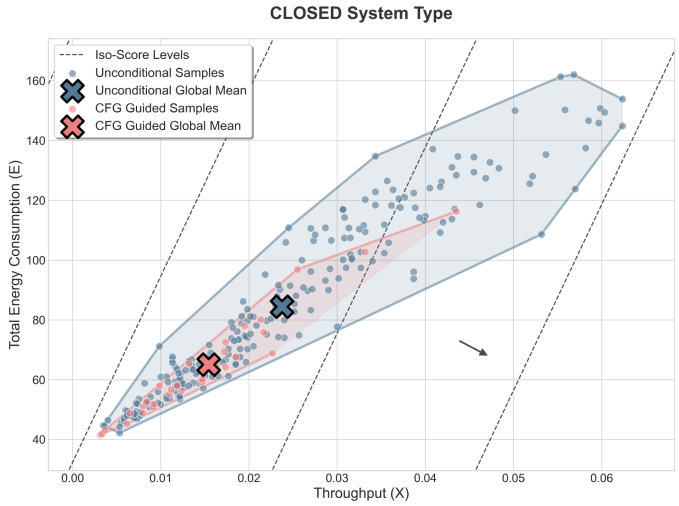


Figure 5.6: Performance distribution in the Throughput-Energy space for the Closed topology: a comparative analysis of Unconditional and CFG Guided configurations, displaying individual sample clusters, statistical means, and associated convex hulls over the iso-score functional landscape.

While the Open System exhibited a significant shift toward superior performance, the Closed System follows a distinct trajectory. As illustrated, the random baseline actually outperform the generative model along the high-Throughput axis, reaching intensities that the model currently fails to replicate. Although the CFG generation adheres to the linear correlation established by the Score weights, it encounters a ”performance barrier” at a more moderate level.

This indicates a struggle to synthesize the highly complex, high-speed architectures present in the random set. The structural intricacy required to sustain such elevated recirculation rates likely renders these specific graphs statistically more difficult to generate while ensuring mandatory connectivity constraints. As a result, the mean performance remains anchored within a more conservative operational zone. In this instance, the model appears to have prioritized the synthesis of stable, valid loops over the pursuit of the extreme performance peaks observed in the baseline.

Statistical Performance Analysis

The quantitative gap between the random baseline and the generated output is evident in the box plots in Figure 5.7.

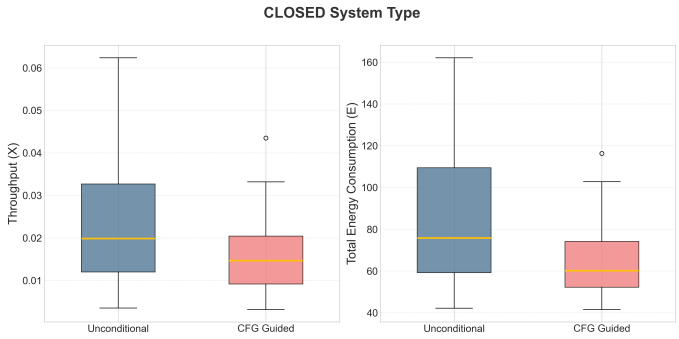


Figure 5.7: Statistical distribution in the Throughput-Energy space for the Closed topology: a box plot comparison between Unconditional and CFG Guided datasets.

The statistical data reveals a definitive shift: the generated batch exhibits a 35.00% reduction in Throughput (0.0154 vs 0.0237 units/time) compared to the random baseline. Similarly, Energy costs decrease by 22.96% (65.04 vs 84.43 energy units). Both variations are statistically significant, with p-values of 9.97×10^{-6} for productivity and 7.60×10^{-7} for energy. In this specific scenario, the reduced power consumption is not a result of efficiency optimization but a direct consequence of diminished operational activity. The generated systems are "slower" than the random examples that surpass them, resulting in lower production rates despite the decreased energy demand.

The generative metrics reported in Table 5.2 provide the key to understand these results.

Metrics	Unconditional	CFG Guided
Validity Rate	26.15%	12.6%
Uniqueness	100.0%	70.0%
Novelty	100.0%	98.0%

Table 5.2: Generative quality metrics for the Closed system topology.

The Validity Rate highlights the extreme difficulty of ensuring a continuous recirculating loop. Interestingly, the CFG Guided model shows a success rate of 12.6%, which is lower than the baseline. This suggests that as the model is pushed toward specific performance targets, satisfying the structural demands of a closed system becomes increasingly complex. As a result, while the unguided model generates solutions across a vast and unconstrained design space, the CFG process restricts its search to a much narrower region where valid solutions are more probable. Despite this more focused approach, the model still struggles to guarantee a high validity rate, as it must prioritize functional reliability over extreme performance values that might otherwise compromise the loop's integrity.

Structural diversity remains a consistent outcome of the framework. The 100.0% Uniqueness of the Unconditional model is a natural consequence of its unconstrained, random evolutionary mode. For the CFG Guided model, a 70.0% Uniqueness paired with 98.0% Novelty proves that the system is not merely duplicating known patterns; instead, it successfully explores the architectural space to synthesize original recirculating systems that were absent from the Training set.

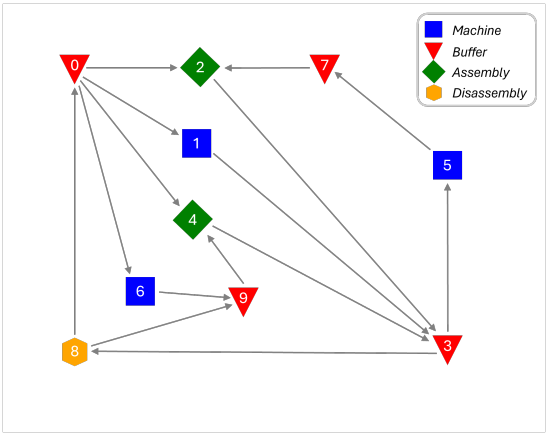
The divergence in the metrics resulting from the two paradigms further illustrates their underlying philosophies. While the Unconditional model produces almost entirely original configurations by focusing on local constraints, the CFG model effectively navigates the trade-off between innovation and structural stability. In doing so, it attempts to align its synthetic outputs with the stable, high-performing trajectories identified during the learning phase.

Performance-Driven Architectural Patterns

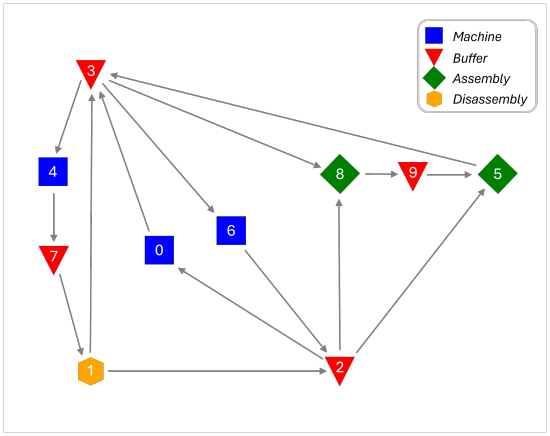
In Closed Systems, the objective evolves from linear progression to the sophisticated management of recirculation. In this context, Throughput serves as a proxy for operational intensity, measuring the aggregate operations performed by active machinery per unit of time. The fundamental challenge lies in balancing internal flow to mitigate the risks of material accumulation or shortages. Failure to maintain this equilibrium results in two critical operational issues: starvation, where machines remain idle due to a lack of components, or blocking, where operations terminate because saturated downstream Buffers cannot accept finished parts.

Unlike Open Systems, where Buffers may have distinct entry or exit roles, the inherent loop constraints of Closed Systems require all Buffers to maintain both

input and output links. While they share the same semantic definition, their operational efficiency is dictated by their degree — the specific number of incoming and outgoing edges. The strategic importance of this connectivity is demonstrated when comparing a low-performance layout (Figure 5.8a) with a high-performance alternative (Figure 5.8b).



(a): Layout A.



(b): Layout B.

Performance Metric	Layout A	Layout B
Throughput	0.006	0.085
Total Energy Consumption	56.73	190.43

Figure 5.8: Comparative analysis of Closed configurations: visual graph topologies and corresponding Throughput-Energy metrics for low-performance (Layout A) and high-performance (Layout B) samples.

The quantitative results in Figure 5.8 demonstrate that Layout B significantly surpasses Layout A in terms of productivity. The fundamental distinction between the two lies in the topological balance of their designs. In the low-performance configuration, the flow is severely restricted by an unbalanced structure where Buffer 0 has only a single incoming connection against four outgoing edges. This disproportion suggests that the Machines dependent on it — nodes 4, 1, 2, and 6 — suffer from frequent starvation events. Since nodes 4, 1, and 2 also serve as the primary inputs for Buffer 3, that Buffer remains nearly empty, triggering a chain reaction that lacks the remaining active elements (nodes 8 and 5) of materials and decelerates the entire recirculation process.

In contrast, Layout B facilitates fluid recirculation by achieving a fairer distribution of input and output connections across all Buffers. In this configuration, central Buffers 2 and 3 form a highly interconnected core, the efficiency of which is further enhanced by nodes 0 and 6 establishing a bidirectional link between these two storage units. Specifically, Machine 0 transfers materials from Buffer 2 to 3, while Machine 6 operates in the reverse direction, creating a nucleus that accelerates material transmission and significantly mitigates the risks of stalling. As a result, operations do not experience blocking phenomena.

Ultimately, this comparison demonstrates that while the model operates within conservative boundaries to ensure structural validity, it remains capable of synthesizing sophisticated architectures when steered toward the efficiency frontier. By optimizing the degree of connectivity for Buffer nodes, the generative process establishes stable internal cycles that sustain high operational intensity. This structural balance effectively prevents the congestion and deadlocks typical of poorly organized closed-loop systems, ensuring a consistent and reliable flow.

5.1.3 Input-Only System

The experimental validation of the Input-Only topology examined the model's ability to optimize the initial filling phase by streamlining inventory distribution from sources to Buffers. The generative agent was tasked with synthesizing layouts that support high intake rates while minimizing the energetic costs and routing complexities of inefficient networks. For this purpose, a high Target Score ($c_{target} = 0.8$) was set to steer the diffusion trajectory toward peak performance, with a Guidance Scale of $\gamma = 2$ to ensure strict adherence to the conditioning signals.

Due to significant computational requirements, this analysis evaluates a single generative batch against a random baseline. Although stochastic, this result serves as a highly indicative benchmark for assessing the system's optimization potential, highlighting how the model moves beyond random distribution to find structured patterns that balance throughput with operational cost.

Operational Space Analysis

The efficacy of the guidance mechanism is immediately apparent in the performance chart displayed in Figure 5.9.

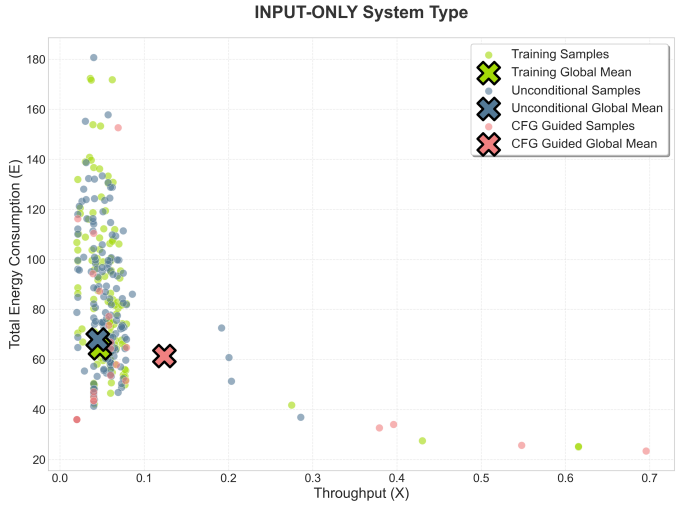


Figure 5.9: Performance distribution in the Throughput-Energy space for the Input-Only topology: a comparative analysis of the original Training dataset, the Unconditional baseline and CFG Guided configurations.

The baseline distribution exhibits a wide vertical spread, indicating that Energy Consumption varies significantly across all production levels. This unguided baseline aligns closely with the original Training data, confirming that the Unconditioned sets accurately reflects the Energy-intensive patterns and the wide variance present in the source domain.

While the model successfully synthesized configurations with Throughput values surpassing those found in the Training dataset, the relative rarity of these high-performance cases led the algorithm to prioritize Energy Consumption optimization. This strategy allowed the framework to maximize the overall Score by navigating around the physical barriers and structural constraints that render high-Throughput graphs particularly difficult to generate within this specific topology. By focusing on efficiency where gains were more accessible, the model ensured a high global performance score while maintaining structural validity.

In Figure 5.10, the relationship between the optimization score and various system components is clearly visible. This visualization highlights how the model successfully navigates the trade-offs between performance metrics, consistently favoring

paths that reduce wasteful complexity.

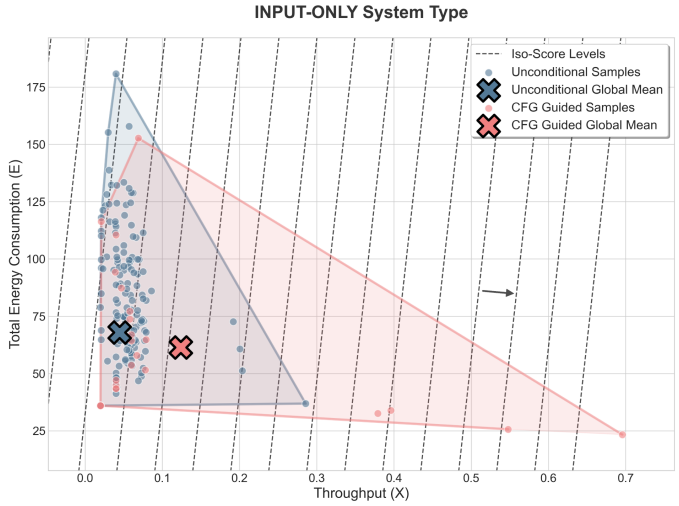


Figure 5.10: Performance distribution in the Throughput-Energy space for the Open topology: a comparative analysis of Unconditional and CFG Guided configurations, displaying individual sample clusters, statistical means, and associated convex hulls over the iso-score functional landscape.

The generated set is significantly compressed along the vertical axis, indicating a consistent reduction in Energy Consumption for identical Throughput levels compared to the baseline. Simultaneously, the distribution expands horizontally, reaching Throughput values previously untouched by the Unconditioned model.

However, the model's ability to consistently replicate these peaks is limited by the scarcity of high-performance samples in the Training data. Consequently, the agent concentrates its optimization primarily on minimizing energy waste within the standard operational range — where it possesses the most reliable information — while still attempting to push performance boundaries whenever learned patterns allow.

Statistical Performance Analysis

The quantitative extent of this optimization is detailed in the box plot comparison presented in Figure 5.11.

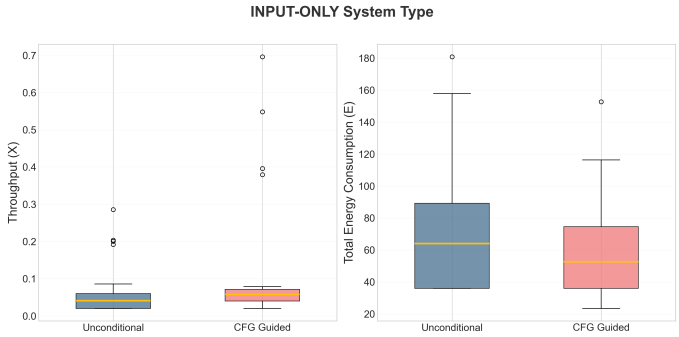


Figure 5.11: Statistical distribution in the Throughput-Energy space for the Input-Only topology: a box plot comparison between Unconditional and CFG Guided datasets.

Regarding Throughput, the Guided group exhibits a substantial increase compared to the baseline, with an average growth of 177.42%. Statistical tests confirm the significance of this result (p-value $= 0.045$). This marked improvement is primarily driven by a select few configurations that achieved exceptionally high production levels. Although these cases are infrequent, they exert a strong influence on the final average. Due to the low Uniqueness value, a substantial number of configurations are removed by the filtering process; consequently, the total number of unique graphs considered in the analysis remains limited. In the data distribution, these solutions appear as "outliers," demonstrating the model's capacity to surpass the standard operational limits of the system.

The true success of the optimization is most evident in Energy Consumption. The data reveals a statistically significant reduction of 15.09% relative to the baseline (p-value $= 0.027$). As illustrated in Figure 5.11, while the random set is characterized by numerous high-Energy outliers, the CFG layouts converge within a high-efficiency zone, confirming that the guidance effectively eliminated redundant or power-hungry paths in favor of streamlined routing.

To assess the structural quality behind these numbers, the generative metrics in Table 5.3 were evaluated.

Metrics	Unconditional	CFG Guided
Validity Rate	28.18%	17.5%
Uniqueness	100.0%	48.0%
Novelty	100.0%	94.0%

Table 5.3: Generative quality metrics for the Input-Only system topology.

The Validity Rate clearly reflects the intrinsic complexity of this configuration. The CFG Guided model achieves a success rate of 17.5%, lower than the 28.18% recorded by the Unconditional baseline. This gap suggests that introducing performance-oriented guidance makes satisfying topological requirements more challenging. While the unguided model relies on a less constrained stochastic search, the CFG guidance adopts a more "prudent" strategy. Given the scarcity of high-Throughput configurations in the original dataset, the model avoids exploring extreme or unpredictable dynamics.

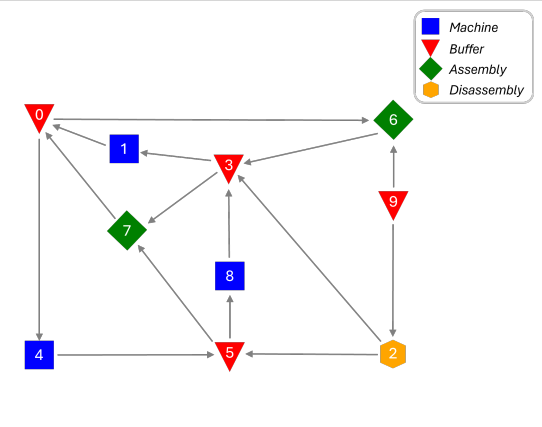
Regarding variety, the 48.0% Uniqueness indicates that the agent identified a set of "reliable" functional patterns. Despite this partial standardization, the high Novelty score of 94.0% confirms the framework is actively synthesizing original architectures rather than simply replicating Training data. This illustrates the balance between innovation and reliability: the Guided agent deliberately narrows its focus, sacrificing some variety to ensure the generated graphs follow high-efficiency pathways.

Performance-Driven Architectural Patterns

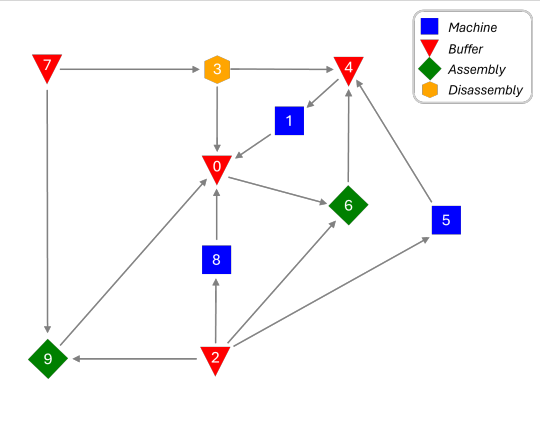
In Input-Only Systems, the objective focuses on optimizing the filling phase by streamlining the movement of inventory from sources to Buffers while simultaneously minimizing Energy costs. Within this specific topology, Throughput is defined as the total volume of items stored in the Buffers divided by the time elapsed during the filling process. The primary goal is to sustain high intake rates and ensure the system reaches capacity rapidly, bypassing the inefficiencies and power waste often associated with convoluted routing networks.

The efficiency of this topology is governed by the equilibrium between "creator" (Assembly) and "consumer" (Disassembly) units. Rather than the raw quantity of input sources, performance is dictated by the connectivity degree of each station.

Since storage elements have infinite capacity, every outgoing edge functions as a multiplier, amplifying a source's impact across the network. When these connections are established strategically, they can saturate the system even with a limited number of "consuming" units. Conversely, reduced connectivity creates a bottleneck that cannot be resolved by merely increasing the number of input sources. For instance, in a configuration featuring one source, two Assembly units, and one Disassembly unit, a technical balance exists between material addition and consumption. However, if the input station is poorly connected, Throughput will remain negligible. This result stems from the dynamic requirements of the simulation: Assembly machines necessitate multiple concurrent inputs to activate. If the primary source fails to distribute material through a sufficient number of channels, these downstream nodes remain idle, effectively stalling the overall production cycle.



(a): Layout A.



(b): Layout B.

Performance Metric	Layout A	Layout B
Throughput	0.023	0.741
Total Energy Consumption	141.20	52.49

Figure 5.12: Comparative analysis of Input-Only configurations: visual graph topologies and corresponding Throughput-Energy metrics for low-performance (Layout A) and high-performance (Layout B) samples.

The data in Figure 5.12 reveals a significant performance gap. Layout A suffers from poor connectivity: the single input source is linked to only two machines, one of which is an Assembly unit. Because this Assembly unit requires a second part

from Buffer 5 to operate, it frequently remains idle, stalling the intake. In contrast, Layout B (Figure 5.12b) achieves much higher performance by utilizing two input Buffers (7 and 2) with a combined total of six outgoing connections. This expanded connectivity allows the system to distribute material much faster, filling storage Buffers 0 and 4 almost immediately.

Notably, Layout B reaches this high Throughput while consuming significantly less Energy than Layout A. This confirms that the generative model successfully learned to eliminate wasteful complexity in favor of a lean, high-intake architecture. By prioritizing direct connections from sources to storage, the model ensures that the system reaches its target state efficiently. Ultimately, this demonstrates that in an Input-Only context, high performance is achieved through streamlined connectivity rather than structural density.

5.1.4 Output-Only System

The analysis of the Output-Only topology examined the system's "emptying" phase, where the primary objective is to clear all stored material from the Buffers. The generative agent was tasked with synthesizing layouts that fluidly move inventory from internal storage toward the output sinks, ensuring the depletion process is not hindered by bottlenecks or isolated sections. To guide the generation, a high Target Score ($c_{target} = 0.8$) was established to steer the diffusion trajectory toward optimal efficiency, while a Guidance Scale of $\gamma = 2$ ensured strict adherence to the conditioning signal.

Due to significant computational requirements, this study evaluates a single generative set against a random baseline. Although this represents a single test, the results offer a definitive indication of the model's optimization capabilities. The comparison specifically highlights how the model rationalizes the exit paths to prevent material stagnation.

Operational Space Analysis

The impact of the guidance mechanism is clearly visible in the performance chart in Figure 5.13.

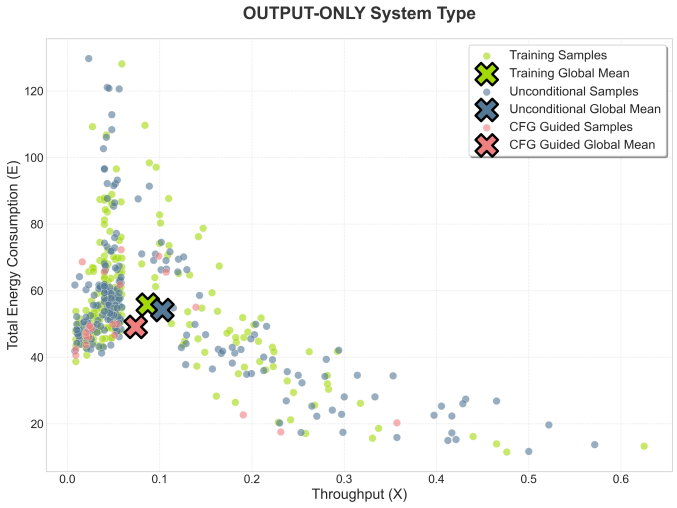


Figure 5.13: Performance distribution in the Throughput-Energy space for the Output-Only topology: a comparative analysis of the Training dataset, the Unconditional baseline, and CFG Guided configurations.

The Unconditional distribution appears as a scattered cloud, typical of a stochastic process. Since these outputs match the spatial dispersion of the original Training data, this unguided model acts as a reliable benchmark for the domain's natural variability. In this configuration, samples cluster in the lower part of the plot with high density at minimal Throughput levels, suggesting that, without guidance, the system struggles to establish the connectivity required for efficient material transport.

The Guided model, while sharing some structural traits with the baseline, explores a much more restricted area of the design space. Instead of a broad search, the model concentrates on a specific subset of configurations, favoring patterns perceived as reliable for the emptying phase. This localized behavior is likely a reflection of the high-density regions within the Training set, which exert a strong influence during inference, leading the agent to replicate stable, familiar patterns rather than navigating toward the high-performance frontier.

The relationship between the score and the system components is further detailed in Figure 5.14.

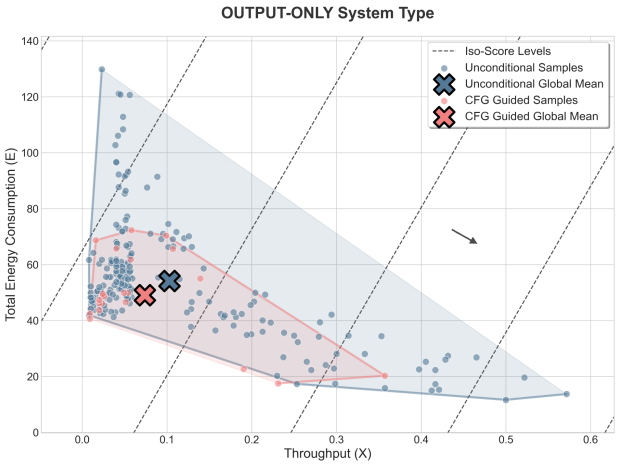


Figure 5.14: Performance distribution in the Throughput-Energy space for the Open topology: a comparative analysis of Unconditional and CFG Guided configurations, displaying individual sample clusters, statistical means, and associated convex hulls over the iso-score functional landscape.

The model restricts its exploration to a narrow set of solutions, specifically avoiding the top-left quadrant and failing to penetrate the bottom-right corner. The absence of samples in the top-left — characterized by high Energy consumption and low Throughput — demonstrates the effectiveness of the optimization logic, which successfully filters out inefficient configurations.

Conversely, the difficulty in reaching the bottom-right zone — which represents peak performance — reflects the scarcity of such configurations within the Training dataset. Because high-performance samples are rare in the original distribution, the model struggles to learn the sophisticated structural logic required to navigate these regions.

Statistical Performance Analysis

The quantitative results are visualized in the box plots in Figure 5.15.

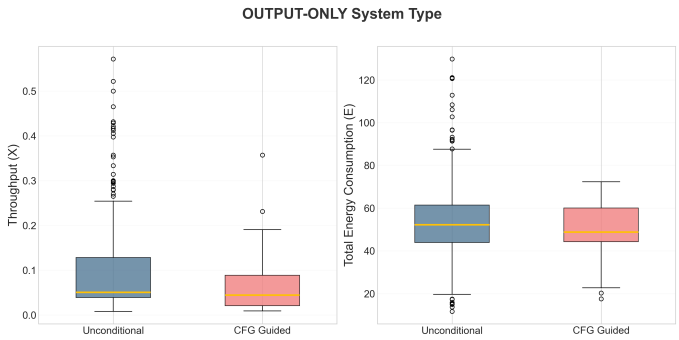


Figure 5.15: Statistical distribution in the Throughput-Energy space for the Output-Only topology: a box plot comparison between Unconditional and CFG Guided datasets.

Regarding Throughput, the guided set exhibits a 28.21% decrease relative to the baseline. While this result lacks absolute statistical significance (p-value = 0.160), it suggests that the agent prioritizes familiar, stable structures encountered during training over more complex, faster architectures.

The most notable improvement is seen in Energy Consumption, with a 9.38% reduction. Although the p-value (0.158) does not meet the threshold for full statistical significance, the box plot reveals that the generation process successfully eliminated nearly all high-energy outliers present in the random set. In this capacity, the model functioned as a selective filter, concentrating layouts within a more consistent and optimized efficiency range.

Finally, the generative metrics are reported in Table 5.4.

Metrics	Unconditional	CFG Guided
Validity Rate	23.70%	49.0%
Uniqueness	100.0%	44.0%
Novelty	100.0%	64.0%

Table 5.4: Generative quality metrics for the Output-Only system topology.

The Validity Rate underscores the topological complexity of this configuration. The CFG Guided model achieves a success rate of 49.0%, more than doubling the

Unconditional baseline. This indicates that guidance effectively maneuvers the generative process toward feasible latent regions, satisfying global constraints far better than a purely stochastic approach.

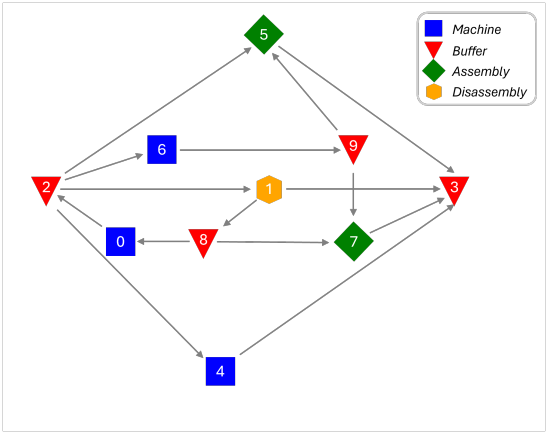
Structural diversity presents a sharp contrast: while the 100.0% Uniqueness of the Unconditional model reflects its random nature, the 44.0% Uniqueness in the CFG model indicates selective pressure toward reliable topological patterns. However, the Novelty score of 64.0% confirms that this pressure does not result in mere replication; instead, the model successfully synthesizes original architectures that were not present in the Training set. This balance of Novelty and Uniqueness suggests the agent identifies and gravitates toward functional layouts, effectively navigating complex constraints while still exploring novel, unobserved regions of the design space.

Performance-Driven Architectural Patterns

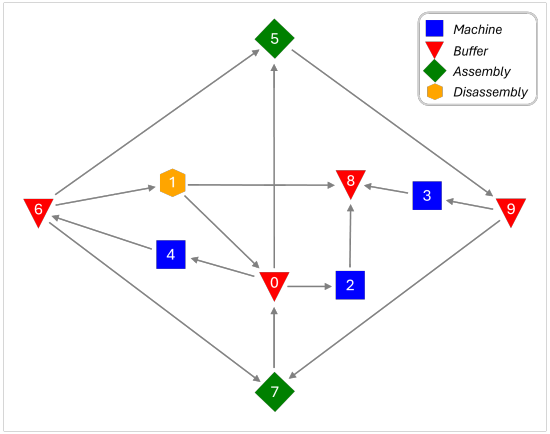
In Output-Only systems, the architectural objective centers on optimizing the "emptying" phase where the goal is to synthesize layouts that ensure all inventory stored within the Buffers is effectively transitioned to output sinks. Unlike the filling phase, performance is measured by the depletion rate of internal storage. Consequently, the model must establish clear routing logic to prevent bottlenecks or isolated segments that would otherwise cause material stagnation.

The effectiveness of this topology depends largely on the integration of "consumer" nodes — specifically Disassembly units and sinks — into the network. While internal machinery regulates the flow, the primary driver of Throughput is exit connectivity. If a layout lacks sufficient pathways to the output sinks, material remains trapped in upstream buffers, rendering the processing speed of individual machines irrelevant.

However, exit connectivity is defined not only by the quantity of links but by their strategic positioning within the graph logic. The following case study illustrates a counterintuitive dynamic: although Layout A (5.16a) possesses more direct connections to the sink than Layout B (5.16b), the latter achieves significantly higher performance due to its superior internal routing efficiency.



(a): Layout A.



(b): Layout B.

Performance Metric	Layout A	Layout B
Throughput	0.059	0.300
Total Energy Consumption	75.65	30.14

Figure 5.16: Comparative analysis of Output-Only configurations: visual graph topologies and corresponding Throughput-Energy metrics for low-performance (Layout A) and high-performance (Layout B) samples.

The low performance of Layout A stems from its highly restrictive connection architecture. In this configuration, the output buffer is linked to two Assembly units, which creates a rigid dependency on the global system state; these machines require multiple concurrent inputs to activate, often leading to operational delays.

Furthermore, Buffer 2 acts as a critical structural bottleneck, possessing four outgoing connections but only a single incoming link. This topological imbalance makes the Buffer highly susceptible to starvation. The impact is particularly severe for Machine 6, the sole feeder for Buffer 9. Since Buffer 9 is tasked with supplying both Disassembly units, this single point of failure significantly decelerates the entire depletion process.

In contrast, the high-performance architecture of Layout B features three direct connections to the output Buffer. The remaining Buffers exhibit a much more balanced distribution of input and output edges. This topological balance is fundamental to maintaining a continuous and effective flow, ensuring that material is consistently moved toward the sinks during the system's emptying phase.

5.2 Ablation Study

To validate the architectural choices defined in Section 5.1, a series of ablation experiments isolates the contribution of individual components to the overall system performance. The analysis quantifies the specific impact of the post-processing repair strategies on validity rates, determines the optimal trade-off in guidance intensity (γ), and evaluates the fidelity of the conditioning mechanism against the requested Target Scores (c_{target}).

5.2.1 Impact of Strict Projector and Repairer Strategy on System Performance

To isolate the individual contributions of the constraint-handling components, this study evaluates the specific roles of the Strict Projector and the Logit-Guided Repairer Strategy by selectively disabling them. This ablation analysis assesses the necessity of the projection mechanism during the sampling process and quantifies the impact of post-processing repair strategies on the final validity rates and performance.

The experimental results reveal a marked divergence in how the system manages structural constraints across different topologies. A clear distinction emerges between the dynamics of Open, Input-Only, and Output-Only configurations and those observed in Closed systems. This behavioral split is due to the dual nature of the Repairer (as discussed in Section 3.2.4): the tailored corrective and generative actions required for different topologies impose fundamentally diverse operational trajectories on the model.

For the first category - comprising Open, Input-Only, and Output-Only topologies - the Open system was selected as the representative case study due to its higher generation speed, which allowed for a more efficient sampling process. While Closed systems are inherently characterized by long generation times, the Open configuration is typically much faster. However, removing the Projector drastically reduces the Validity Rate, making the task of gathering a sufficient number of valid samples extremely time-consuming even for this faster representative. Hence, for both categories of analysis, a single experimental run was conducted for each ablation configuration.

To mitigate bias from identical configurations, all datasets were filtered to remove duplicate graphs based on the Uniqueness metric. This pruning process ensures that the analysis is not skewed by repeated sampling of identical structures, allowing the performance metrics to accurately reflect the model's capacity to explore the feasible manifold of the design space.

Open, Input-Only, and Output-Only Topologies

Focusing on the first category — comprising Open, Input-Only, and Output-Only topologies — the experimental evidence identifies the Strict Projector as the sole determinant of structural validity. In these linear or semi-linear flows, the post-processing Repairer proves to be structurally redundant when the projection is active. To demonstrate this redundancy, Table 5.5 reports the ablation results specifically for the Open System.

Configuration Strategy	Strict Projector	Logit-Guided Repairer	Validity Rate
Full Pipeline	ON	ON	70.4%
Projector Ablation	ON	OFF	68.5%
Repairer Ablation	OFF	ON	0.7%
Unconstrained	OFF	OFF	$\approx 0.0\%$

Table 5.5: Ablation study on structural validity for the Open system, comparing the individual and combined impact of the Strict Projector and Logit-Guided Repairer modules.

First, comparing the Full Pipeline to the Projector Ablation reveals that the Logit-Guided Repairer provides a negligible contribution when the Strict Projector is active. This minor 1.9% difference falls within the expected range of stochastic variance, confirming that the Strict Projector is the primary driver of structural validity. By enforcing constraints directly during the diffusion process, the model generates adjacency matrices that are already topologically valid, making post-processing repair largely redundant for these systems.

In contrast, disabling the Projector (Repairer Ablation) leads to a dramatic collapse in performance, with Validity dropping to 0.7%. This outcome highlights

a fundamental functional limitation: in these configurations, the Repairer's role is restricted to that of a subtractive filter. While it remains capable of pruning invalid edges to satisfy degree constraints, it lacks the constructive agency required to synthesize new connections. Without the Projector to ensure flow continuity during the diffusion phase, the Repairer cannot synthesize the missing links required to establish a functional network.

Finally, the Unconstrained configuration yields a near-zero Validity Rate. This is consistent with the probabilistic nature of generative models; without a mechanism to bound the output to a feasible topological manifold, the mathematical probability of stochastically sampling a graph that perfectly satisfies complex connectivity and flow constraints is essentially null.

The choice of constraint-handling components determines more than just the Validity Rate: it also affects the qualitative performance of the generated solutions. As shown in Table 5.6, mean variations in Throughput and Energy Consumption shift significantly depending on which mechanisms are active.

Configuration Strategy	Throughput Variation	Energy Variation
Full Pipeline	+166.23%	+38.92%
Projector Ablation	+127.33%	+24.69%
Repairer Ablation	+37.93%	+5.01%

Table 5.6: Ablation study on operational performance for the Open system, comparing the individual and combined impact of the Strict Projector and Logit-Guided Repairer modules.

The performance gap observed in the Repairer Ablation configuration highlights a fundamental limitation: the repair mechanism alone cannot compensate for the lack of active guidance during the generative phase. Without the Projector, the system becomes too random and fails to maintain a coherent structure. While the Repairer can fix small local errors, it cannot reorganize the overall graph to make it efficient. As a result, the few valid configurations produced are poorly optimized and do not reach high performance levels.

Conversely, the results for the Full Pipeline and the Projector Ablation are almost identical. The small differences in Throughput and Energy Consumption are

statistically insignificant and fall within the expected range of stochastic variation proving that the Projector is sufficient on its own to steer the model toward the high-performance solution space.

This similarity remains clear even after having filtered out duplicate graphs, a process that necessarily reduced the number of unique samples. In fact, if the analysis were conducted on the full set of generated data - effectively giving more weight to the most frequent high-performing configurations — the gap between these two strategies would narrow even further, with results differing by only a few points. This confirms that the high level of performance is a structural property derived from the Projector's logic.

Closed Topologies

The dynamics of the Closed system reveal a fundamentally different interaction between the components. As anticipated, the requirement to form a valid, continuous recirculating loop without dead-ends or disconnects proves to be a significantly more challenging constraint to satisfy. The ablation results for this topology are detailed in Table 5.7.

Configuration Strategy	Strict Projector	Logit-Guided Repairer	Validity Rate
Full Pipeline	ON	ON	12.6%
Projector Ablation	ON	OFF	4.8%
Repairer Ablation	OFF	ON	0.6%
Unconstrained	OFF	OFF	$\approx 0.0\%$

Table 5.7: Ablation study on structural validity for the Closed system, comparing the individual and combined impact of the Strict Projector and Logit-Guided Repairer modules.

A comparison between the Full Pipeline and the Projector Ablation highlights a critical finding: for cyclic topologies, the Logit-Guided Repairer is far from redundant. On the contrary, its activation yields a greater than twofold increase in the Validity Rate. This proves that while the Strict Projector is effective in guiding the generation toward a globally circular structure, it often fails to satisfy the specific degree constraints required for a functional loop. Here, the Repairer plays a decisive

role, acting as a fine-tuning mechanism that corrects these ”near-miss” configurations by pruning excess edges or resolving local conflicts that would otherwise makes the graph invalid.

The analysis of the Repairer Ablation further supports the hierarchy established in the linear case. Despite the crucial importance of the Repairer within the full pipeline, the component is virtually useless in isolation. Without the Projector to define the global cyclic skeleton during the process, the Repairer faces unstructured or fragmented adjacency matrices that cannot be successfully managed.

Finally, the Unconstrained configuration results in a null Validity Rate. This outcome is consistent with the results obtained for the Open System, confirming a universal behavior across different topologies. The simultaneous deactivation of both the projection and repair mechanisms leads to a complete collapse of the generative capability. Without active constraint enforcement to bound the diffusion output to the feasible manifold, the model acts as a purely stochastic generator, producing matrices that fail to satisfy the fundamental topological rules.

The impact of these mechanisms on operational performance is shown in Table 5.8.

Configuration Strategy	Throughput Variation	Energy Variation
Full Pipeline	-35.00%	-22.96%
Projector Ablation	-7.60%	-9.14%
Repairer Ablation	-7.04%	-8.50%

Table 5.8: Ablation study on operational performance for the Closed system, comparing the individual and combined impact of the Strict Projector and Logit-Guided Repairer modules.

Performance results for Projector Ablation and Repairer Ablation are qualitatively comparable. Given the stochastic nature of the diffusion process, these marginal deviations from the baseline suggest that, when operating in isolation, each mechanism merely reproduces the standard behavior of the training data. In these scenarios, the generative agent creates loops that preserve learned flow dynamics without triggering significant operational shifts.

However, a critical divergence occurs in the Full Pipeline, which exhibits a distinct and non-negligible performance collapse. This variation cannot be attributed to random noise; instead, it points to a deterministic structural conflict between the two modules. While the Strict Projector and the Logit-Guided Repairer function adequately on their own, their simultaneous activation creates destructive interference.

The Projector operates by locally saturating all available connection slots (node degrees), effectively "locking" the graph architecture. Consequently, the Repairer is forced to intervene on these rigid structures to ensure the loop is fully closed. Finding no remaining capacity for new connections, the Repairer triggers an aggressive logic, pruning high-probability edges to force the insertion of artificial closures.

These forced topological cuts create severe bottlenecks, leading to the sharp drop in performance that distinguishes the Full Pipeline from the more stable ablation scenarios.

5.2.2 Impact of Guidance Scale (γ)

The Guidance Scale γ serves as the primary hyperparameter governing the Classifier-Free Guidance mechanism, mathematically controlling the magnitude of the logit extrapolation between the conditional and unconditional estimates. At the generative level, this coefficient dictates the intensity of the steering effect: higher values progressively shift the probability density toward configurations that strictly maximize the Target performance metrics, effectively narrowing the search space to suppress suboptimal structures at the potential cost of sample diversity.

To assess the impact of this steering effect under high-demand conditions, the Performance Target Score was fixed at $c_{target} = 0.8$. This benchmark serves as a constant reference point, allowing for a precise observation of how different guidance intensities influence the model's ability to converge toward a high-performance objective.

To quantitatively evaluate this behavior, the analysis is structured around the systematic processing of multiple independent generations for each Guidance Scale value. The tested values were selected on an exponential scale — specifically 0.0, 0.25, 0.5, 1.0, 2.0, and 4.0 — to provide a more granular observation of the system's behavior near the zero-point while still capturing trends at higher intensity levels. For each of these settings, five independent datasets were generated; the data points

presented in the following analysis represent the average values across these sets. From this aggregated data, mean performance metrics — including Throughput, MAD, Novelty, Validity Rate, and Uniqueness — were calculated.

The statistical robustness is ensured through 90% confidence intervals, which account for the variability inherent in the diffusion process and provide a reliable measure of how consistently the model responds to the guidance signal across different random initializations.

To keep the results as clear as possible, the scoring system was simplified by temporarily removing the Energy Consumption factor ($w_E = 0$). Consequently, the performance Target was based strictly on Throughput. This was a strategic choice: by focusing only on productivity, the potential confusion caused by a combined score was avoided. This decoupling allows for a clear observation of the direct, causal link between the Guidance Scale γ and the model's ability to reach a specific performance goal. Furthermore, the analysis was performed exclusively on Open systems due to practical considerations regarding computational cost. Given the lower simulation overhead of this topology, it was possible to perform extensive testing across multiple scales without excessive runtime.

The impact of the steering mechanism on performance is clearly visible in the Throughput distribution shown in Figure 5.17. When the Guidance Scale is low ($\gamma \leq 0.5$), the results exhibit high variance, with many samples falling into low-performance zones. As γ increases toward 2.0, the mean Throughput rises toward the target, and the distributions become significantly tighter. This indicates that a stronger steering pressure forces the model to discard poor solutions and focus on high-efficiency architectures, effectively reducing the generation of low-performing outliers.

However, as the scale reaches $\gamma = 4.0$, a "saturation" effect emerges: the model tends to over-activate the guidance, leading to an over-performance that pushes the Throughput beyond the intended target. While this high intensity further collapses the variance, it introduces a systematic bias where the generated architectures exceed the desired performance levels. This suggests that at high scales, the steering mechanism prioritizes maximizing performance and ensuring consistency over a strict, centered adherence to the Target value.

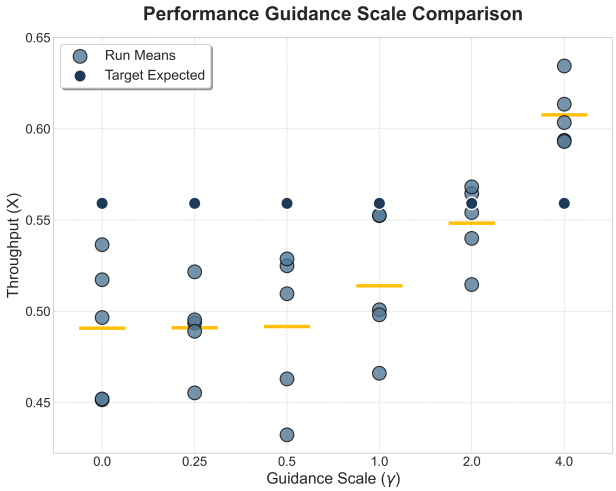


Figure 5.17: Throughput performance across tested Guidance Scale (γ) levels. The large dark blue markers represent the theoretical Throughput value corresponding to the specific Target Score imposed during the generation process.

The precision of the system is further elucidated by the MAD trend in Figure 5.18.

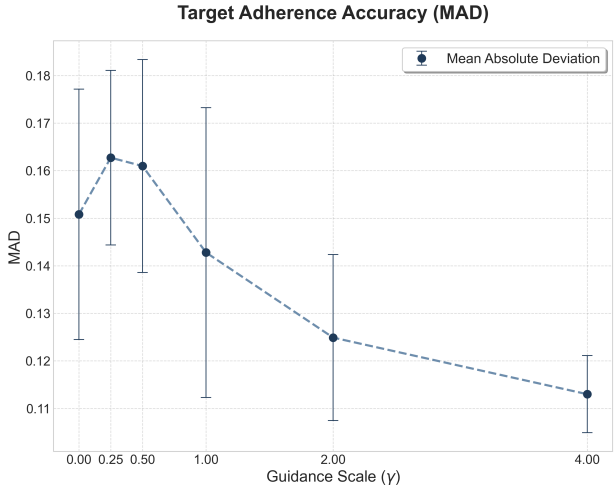


Figure 5.18: MAD performance across tested Guidance Scale (γ) levels, illustrating the distribution of deviation values obtained from independent experimental runs relative to the fixed Target Score.

With the Target Score fixed at 0.8, the deviation decreases steadily as the guidance scale increases, reaching its minimum at $\gamma = 4.0$. At first glance, this result

may appear counterintuitive: although the mean at this peak level is numerically further from the target than at $\gamma = 2.0$, the total deviation is lower.

This behavior stems from a crucial distinction between different levels of statistical variability. While the strip plots — constructed using the mean values of each dataset — might suggest that a scale of 2.0 is closer to the Target, they primarily reflect the variance between the means of different sets rather than the internal dispersion of individual samples.

Crucially, although the collective means cluster closer to the Target at lower scales, the internal variability of each generation remains significantly higher compared to the most intense guidance level. In those lower regimes, the distribution is characterized by numerous low-performing outliers that deviate sharply from the requested performance. Since the MAD is highly sensitive to such extreme values, the total deviation remains elevated even when the average of the means appears more favorable.

However, as illustrated by the Novelty trend in Figure 5.19, this focus on precision comes at the expense of originality. As the guidance strength increases, Novelty consistently declines. This occurs because the model, in prioritizing high-performance, tends to "play it safe" by falling back to frequent training patterns rather than exploring truly novel or unconventional structures.

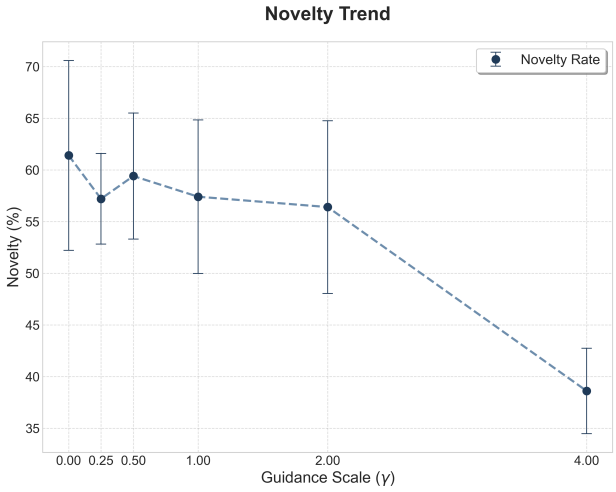


Figure 5.19: Novelty performance across tested Guidance Scale (γ) levels, illustrating the distribution of average Novelty values obtained from independent experimental runs.

Finally, the overall balance of the system is summarized by the trade-off between structural validity and batch diversity in Figure 5.20. As evident, there is a clear opposite trend: while the Validity Rate climbs (surpassing 80% at $\gamma = 4.0$), the Uniqueness of the samples drops sharply after $\gamma = 1.0$.

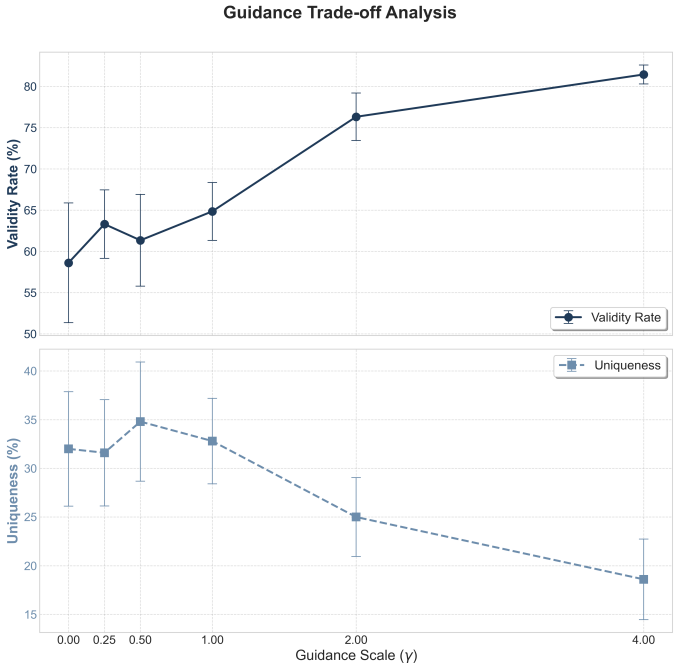


Figure 5.20: Validity Rate and Uniqueness performance across tested Guidance Scale (γ) levels, illustrating the distribution of average percentage values obtained from independent experimental runs.

This phenomenon occurs because the Guidance Scale (γ) increasingly constrains the model's exploratory "freedom." At lower values, the agent investigates a broad spectrum of architectures; however, many of these configurations remain invalid or chaotic due to insufficient directional pressure.

By increasing γ, the steering mechanism forces the model to concentrate on narrow regions of the probability space associated with high-performance and valid layouts. While this ensures that nearly all generated graphs are functional, it induces a mode collapse (or concentration) effect.

5.2.3 Impact of Performance Target Score (c_{target})

The Performance Target Score serves as the primary control signal, mathematically steering the generative process toward topological designs that satisfy specific productivity standards defined by the parameter c_{target}.

For this analysis, the Guidance Scale was fixed at 2.0. This value was selected to provide sufficient directional pressure to reveal how the system adapts to varying Target requests, while avoiding the risk of "over-stretching" the underlying architecture. By maintaining this balance, the agent can explore the design space effectively without collapsing into the repetitive or biased patterns often triggered by extreme guidance intensities.

To quantitatively evaluate the algorithm's behavior, the analysis is conducted across five discrete Target levels: 0.6, 0.7, 0.8, 0.9, and 1.0. This range was selected to observe the model's response as it moves from standard performance requirements toward the theoretical maximum of the design space. For each of these settings, five independent datasets were generated; the data points presented in the following analysis represent the average values across these sets. From this aggregated data, mean performance metrics — including Throughput, MAD, Novelty, Validity Rate, and Uniqueness — were calculated.

The statistical robustness is ensured through 90% confidence intervals, which account for the variability inherent in the diffusion process and provide a reliable measure of how consistently the model responds to the guidance signal across different random initializations.

Mirroring the methodology of the previous study, consistency is maintained by adopting identical constraints for this analysis. By setting $w_E = 0$, the investigation isolates Throughput, allowing for a clear observation of the correlation between c_{target} and performance without the interference of conflicting variables.

The effectiveness of Score conditioning is first shown by the Throughput distribution in Figure 5.21. As the requested Target c_{target} increases from 0.6 to 1.0, the entire distribution of generated graphs shifts in the desired direction. However, this movement is notably slow compared to the changes in the Target Score. While the mean values trend upward, they do not perfectly keep pace with the specific targets requested, showing a "lag" between the user's request and the model's actual output. This indicates that while the model is responsive to the input signal, there

is a limit to how quickly or aggressively it can shift its internal logic toward the requested performance range.

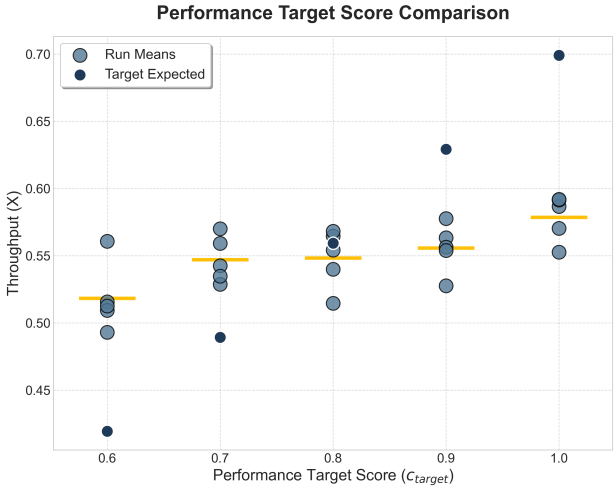


Figure 5.21: Throughput performance across tested Target Score (c_{target}) levels. The large dark blue markers represent the theoretical Throughput value corresponding to the specific Target Score imposed during the generation process.

The MAD trend illustrated in Figure 5.22 clarifies the boundaries of the model's controllability: the deviation does not decrease linearly with the Target.

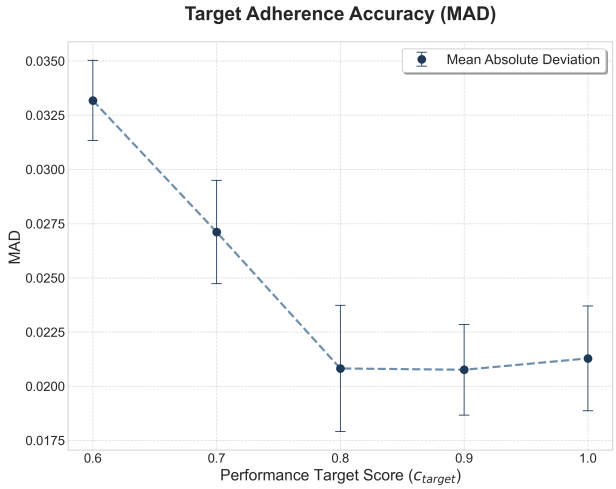


Figure 5.22: MAD performance across tested Target Score (c_{target}) levels, illustrating the distribution of deviation values obtained from independent experimental runs relative to the fixed Guidance Scale.

As evident, while performance improves significantly for targets up to 0.8, accuracy seems to degrade as the objective reaches the 1.0 threshold. This suggests that pushing the model to its absolute operational limit yields diminishing returns; the agent struggles to align the requested performance with the structural constraints of the design space, leading to a loss of precision as it approaches the edge of the feasible manifold.

An interesting phenomenon emerges when comparing the 0.8 and 0.9 benchmarks. As shown in Figure 5.21, the mean result for the 0.8 target is positioned significantly closer to the objective dot than in the 0.9 case. Despite this closer proximity of the average, the MAD remains remarkably similar between the two configurations. This apparent paradox is explained by the distinction between inter-set variability and intra-generation dispersion. While the strip plot suggests that the 0.8 results are more accurate at a collective level, these visualizations are constructed using the mean values of each generated set. Consequently, the plot reflects the variance among the averages of different batches rather than the dispersion of individual samples within those batches.

At the 0.8 target, the internal variability of each generation is significantly higher. These distributions are characterized by a wider spread and numerous low-performing outliers that deviate sharply from the requested performance. Since the MAD is highly sensitive to such extreme deviations, these ”heavy-tail” misses keep the total deviation elevated. In contrast, at the 0.9 benchmark, the model achieves greater structural consistency. Although the mean is more distant from the objective, the generation effectively eliminates the extreme outliers. This reduction in internal dispersion balances the total deviation, resulting in a MAD that is equivalent to the 0.8 case despite the less favorable average positioning.

Demanding peak performance significantly impacts the model's exploratory range, as evidenced by the Novelty trend in Figure 5.23. While this metric remains relatively stable across lower Targets, a sharp decline occurs as the objective reaches 1.0. This trend indicates that when pushed toward its theoretical limit, the agent ceases to synthesize original configurations. Instead, it regresses to a narrow subset of established, high-performing patterns extracted from the training data. At these extreme scales, the priority shifts from architectural innovation to the reliable replication of known ”optimal” structures, effectively sacrificing structural diversity to ensure objective compliance.

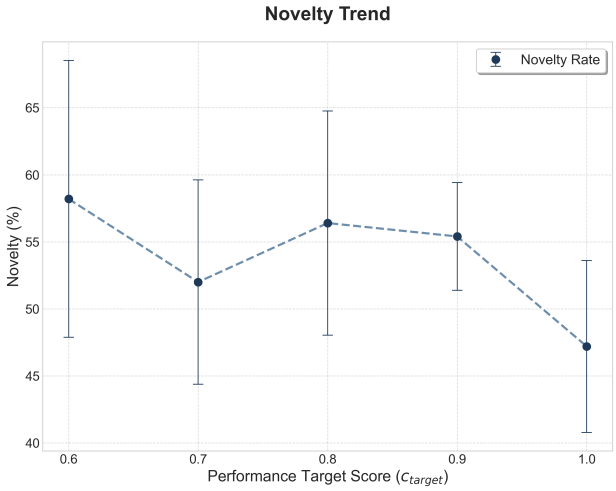


Figure 5.23: Novelty performance across tested Target Score (c_{target}) levels, illustrating the distribution of average Novelty values obtained from independent experimental runs.

The relationship between Validity Rate and Uniqueness in Figure 5.24 summarizes the overall stability of the generative process.

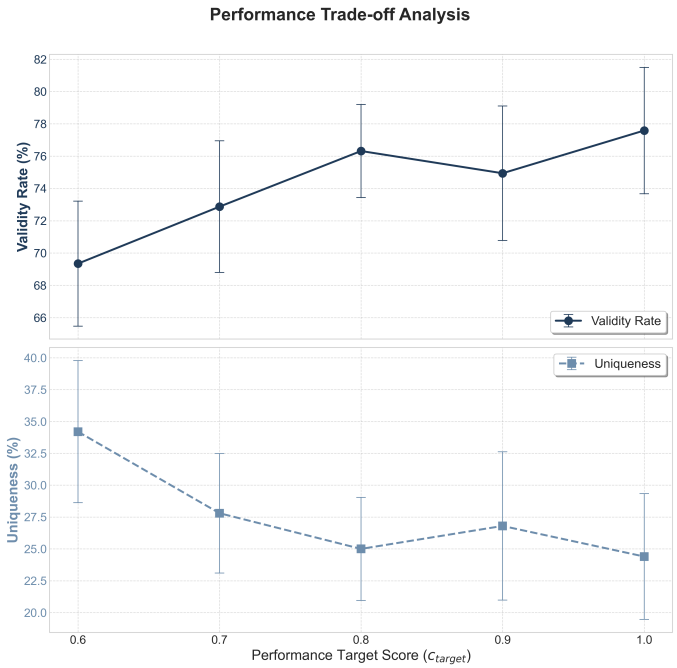


Figure 5.24: Validity Rate and Uniqueness performance across tested Target Score (c_{target}) levels, illustrating the distribution of average percentage values obtained from independent experimental runs.

Notably, the Validity Rate remains high throughout the entire range and even shows a gradual improvement as the benchmark increases. This confirms that higher performance requests do not compromise the basic structural rules of the system; in fact, the model appears to find more stable and valid configurations as the goals become more demanding.

The upward trend in validity is mirrored by a clear downward trend in Uniqueness, which falls from nearly 35% at lower targets to roughly 25% as the target reaches 1.0. This diverging behavior - where graphs become more "correct" but also more repetitive — suggests a potential mode collapse effect.

This shift indicates that the model is trading its creative diversity for reliability. By prioritizing high-performance and functional layouts, the system begins to repeatedly sample from a limited set of "safe" solutions that it has learned are effective.

5.3 Score Weighting Analysis

To validate the sensitivity of the generative process to the underlying objective function, a specific analysis was conducted to quantify how the definition of the weighting parameters w_X and w_E influences both the training landscape and the resulting topological distributions. The coefficients of the score function S serve as the primary steering mechanism during the inference phase; by altering these values, the system reconfigures the priority associated with individual configurations within the training manifold, effectively shifting the model's focus between competing industrial objectives.

Taking advantage of the high computational efficiency of Open systems, this characteristic was evaluated by comparing two distinct generative sets synthesized from the same algorithmic baseline. Specifically, an Energy-driven philosophy ($w_X = 0.25, w_E = 0.75$) was compared against a Throughput-oriented configuration ($w_X = 0.75, w_E = 0.25$). To ensure that the observed variations originated solely from the weighting logic, both configurations utilized identical inference parameters: a Guidance Scale (γ) of 1.0 and a Performance Target (c_{target}) of 0.8.

The results in Figure 5.25 show that the model effectively follows the requested priorities. By changing the weights, the generated clusters shift in different directions within the operational space, demonstrating that the system can adapt to different

industrial goals.

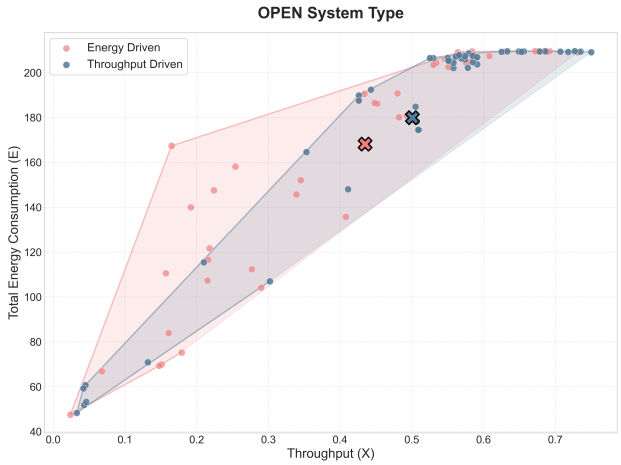


Figure 5.25: Performance distribution in the Throughput-Energy space for the Open topology: a comparative analysis of Energy-driven and Throughput-driven configurations, displaying individual sample clusters together with local means statistical indicators.

In the Energy-driven test, the average values for both energy and throughput are lower than in the production-oriented set. This confirms that the model is sensitive to the weights and tries to adapt the graph structure to satisfy the most important metric.

However, there is a difference between the theory and the actual results. Ideally, the Energy-driven layouts should cluster in the bottom-left corner (minimum energy). In reality, the model finds it difficult to populate that area. This suggests a "structural barrier": when the model tries to reduce energy too much, it often fails to create enough connections to keep the plant functional.

This is confirmed by the Validity Rate data: while the throughput-oriented guidance maintains a validity rate of 69.4%, the Energy-driven configuration drops to 36.8%. This gap indicates that when the system tries to minimize energy, it often fails to create layouts that follow basic topological rules, leading to many failed generations. The model must balance the push for energy efficiency with the need to keep a functional structure, which proves to be much harder than simply optimizing production capacity.

6

Conclusions and Future Developments

The transition toward adaptive manufacturing requires a shift from traditional heuristics to generative design capable of managing complex industrial constraints. By integrating Discrete Diffusion with engineering-guided strategies, this research provides a robust framework for synthesizing production layouts that are both structurally valid and operationally optimized.

The following sections summarize the main research achievements, discuss current limitations, and outline future developments.

6.1 Main Achievements

The primary contribution of this research lies in the development of a generative framework capable of synthesizing industrial graphs that strictly comply with the specific requirements necessary for their functional characterization. This achievement is driven by the model's ability to satisfy complex hierarchical and domain-specific constraints, ensuring that the generated topologies are not merely abstract patterns but conceptual industrial frameworks.

The structural integrity of these layouts is guaranteed by the integration of a Logit-Guided Repair mechanism and the imposition of strict topological rules during the inference phase. This approach ensures compliance with physical and logical requirements — such as node-type compatibility, flow direction, and connectivity — effectively correcting local and global violations without the need for manual post-processing.

To move beyond structural validity, the framework utilizes Classifier-Free Guidance (CFG) to steer the generative process toward high-performance configurations.

The robustness of this guidance was validated across diverse topologies, including Open, Closed, Input-Only, and Output-Only systems, demonstrating that the model can adapt its architectural logic to the specific demands of each system type.

The efficacy of this mechanism is quantitatively demonstrated in the Open System analysis, which serves as a representative benchmark of the framework's capabilities. In this configuration, the guided model achieved a mean throughput increase of 156.10% compared to the unguided baseline, rising from 0.146 to 0.373 units/-time. This performance shift is validated by a p-value of 7.94×10^{-3}, confirming with high statistical confidence that the guidance effectively steers the generation toward the optimal operational frontier. Furthermore, the model maintained a Validity Rate of $65.32\% \pm 4.08\%$ even under intense optimization pressure, proving that significant performance gains do not compromise the structural quality of the synthesized layouts.

A fundamental component of this achievement is the integration of a custom Discrete Event Simulation (DES) module, which plays a dual role in the framework. Initially, the simulator is utilized during the Performance Labeling phase to provide precise ground-truth metrics (Throughput and Energy Consumption) for the training dataset. Subsequently, it serves as the final evaluator in the generative pipeline ensuring that the generated designs actually work while meeting high-performance targets.

6.2 Limitations

Despite the positive results, there are some limitations that define the current boundaries of this approach. A major challenge is the computational intensity of the process. Although the system is faster than traditional methods for large explorations, the iterative nature of the diffusion steps requires significant time and hardware resources which limited the ability to conduct very large statistical tests for hyperparameter tuning and prevented a massive validation across all possible system types.

Furthermore, the generated layouts are currently schematic representations. While the model successfully defines the logical flow and connections, it provides an abstracted description of the industrial system. This means that, at this stage, it does not account for specific physical characteristics, such as: machine and product de-

tails, physical dimensions, and temporal scheduling. Because of these factors, the framework remains a "high-level" design tool. It provides a strong logical backbone, but translating these abstract graphs into real-world manufacturing facilities still requires further steps to include spatial and dynamic details.

6.3 Further Developments

While the current framework is robust, its practical application is limited by an abstracted representation of the system. At this stage, the model focuses on the structural logic and functional connection, omitting the specific physical features of the layout. Additionally, the management of computational complexity and the selection of guidance hyperparameters currently rely on a generalized configuration. At this stage, these parameters follow a broad definition rather than being case-specific, meaning they have not yet been fine-tuned for every individual system type. To evolve this research prototype into a large-scale industrial tool, it is essential to move beyond the current nature of the generations by optimizing algorithmic efficiency and integrating logical flow synthesis with real-world spatial and dynamic constraints.

Based on these considerations, the following sections outline the priority areas for future research, defining the fundamental steps to achieve full automation in Facility Layout Planning.

Refining Generative Performance through Automated Tuning and Scaling

The approach used in this thesis to define hyperparameters was a hybrid strategy. It was largely based on established literature benchmarks for architecture and diffusion settings, combined with a sensitivity analysis limited to a specific subset of training dynamics. To move beyond these heuristics, future developments should aim for a parameter definition that depends solely on the system itself, rather than on assumptions from external studies. A more robust optimization mechanism is needed to evaluate the complex correlations between parameters and the intrinsic impact each one has on the others. To achieve this, a Bayesian approach utilizing Gaussian Processes would be an ideal solution, enabling a sophisticated and automated exploration of the hyperparameter landscape.

Furthermore, while this research focused primarily on Open Systems, future work should include a more rigorous experimental campaign dedicated to each of the four system types. Because every topology possesses its own unique complexities, dedicated tuning is likely to result in significantly different optimal configurations for each. To properly validate these differences, it is essential to move beyond single tests and conduct extensive statistical evaluations using multiple generation sets for every system. Finally, this framework should support an increase in the dimensionality of the networks. Scaling the model from its current foundation to larger, more complex graphs would allow it to reflect the intricacies of real-world industrial facilities.

Ultimately, the main obstacle to these developments is the high computational demand. In generative AI for industrial design, a constant trade-off exists between model complexity and the resources required for processing. Due to these resource constraints, the full implementation of the three main objectives — testing all four system types, conducting deep statistical validations, and scaling the system to more nodes — was not feasible within the scope of this study. Running such large-scale experiments would have exceeded the practical time and hardware limits available. Overcoming these barriers remains a primary focus for future research, allowing the model to evolve from a research prototype into a practical tool for large-scale industrial planning.

Integration with Geometric Layout Synthesis

As established in the literature, Facility Layout Planning is divided into two main stages: Topological Design (defining material flows) and Geometric Design (placing resources in physical space). While existing research often treats the topology as a fixed input, this thesis addressed the ”pre-geometric” phase to provide an optimized logical backbone.

A key future development is the creation of an integrated framework where these two stages communicate directly. In this scenario, logical connections and the spatial distribution of machines would be determined together rather than in isolation. By incorporating physical constraints — such as machine footprints and facility boundaries — into the generative loop, the system would be guided by both flow logic and real-world feasibility.

The goal is to achieve a coordinated optimization where the best performance

is found through a trade-off between an efficient logical structure and an effective physical placement. Only through this combined approach can the framework fully address the Facility Layout Problem (FLP). This integration would transform the model from an abstract flow pattern into a complete industrial design tool, capable of determining the exact position of every element while maintaining peak operational performance.

Refining System Dynamics for Improved Physical Realism

A critical direction for future development is to increase the level of detail within the system to achieve greater realism and model validity. Enhancing the characterization of the system involves two main levels: machine properties and product specifications.

The first level aims for a more detailed representation of the system by defining machine properties individually. This involves incorporating realistic parameters such as specific failure modes (reliability) and varying processing times. In practice, this would enrich the simulation module — which is responsible for computing system performance — making the feedback provided to the generative model much more accurate and "reality-based."

The second level focuses on products. Future versions of the model should be able to handle specific product routings, ensuring that the generated logical connections satisfy the unique paths required by different manufacturing items. This is more complex because it requires the definition of new loss functions (soft constraints) and topological rules (hard constraints) to ensure specific patterns are respected.

Combining these improvements will lead to systems that are more grounded in reality. Such an evolved tool would be highly valuable in industrial applications, helping engineers evaluate potential machine reconfigurations or design new plants from scratch while accounting for the actual limits and difficulties of industrial engineering.

Bibliography

[1] Jacob Austin et al. *Structured Denoising Diffusion Models in Discrete State-Spaces*. Feb. 2023 (cit. on p. 11).

[2] Saif Benjaafar, Sunderesh S. Heragu, and Shahrukh A. Irani. "Next Generation Factory Layouts: Research Challenges and Recent Progress." In: *Interfaces* 32.6 (2002), pp. 58–76 (cit. on p. 3).

[3] Aleksandar Bojchevski et al. *NetGAN: Generating Graphs via Random Walks*. June 2018 (cit. on p. 7).

[4] Sam Bond-Taylor et al. "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.11 (Nov. 2022), pp. 7327–7347 (cit. on p. 5).

[5] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. May 2021 (cit. on p. 5).

[6] Zhou Cai, Xiyuan Wang, and Muhan Zhang. *Latent Graph Diffusion: A Unified Framework for Generation and Prediction on Graphs*. Feb. 2024 (cit. on p. 16).

[7] Hyungjin Chung et al. *Diffusion Posterior Sampling for General Noisy Inverse Problems*. May 2024 (cit. on pp. 14, 15).

[8] *Deep Learning* (cit. on p. 41).

[9] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. June 2021 (cit. on p. 13).

[10] Amine Drira, Henri Pierreval, and Sonia Hajri-Gabouj. "Facility layout problems: A survey." In: *Annual Reviews in Control* 31.2 (Jan. 2007), pp. 255–267 (cit. on p. 3).

[11] Vijay Prakash Dwivedi and Xavier Bresson. *A Generalization of Transformer Networks to Graphs*. Jan. 2021 (cit. on pp. 76, 77).

[12] PIUS J. EGBELU and JOSE M. A. TANCHOCO. "Characterization of automatic guided vehicle dispatching rules." In: *International Journal of Production Research* 22.3 (May 1984), pp. 359–374 (cit. on p. 5).

[13] Faezeh Faez et al. *Deep Graph Generators: A Survey*. Dec. 2020 (cit. on pp. 5, 6, 8).

[14] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. Apr. 2019 (cit. on p. 64).

[15] Kilian Konstantin Haefeli et al. *Diffusion Models for Graphs Benefit From Discrete State Spaces*. Aug. 2023 (cit. on pp. 10, 11).

[16] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX." In: Pasadena, California, June 2008, pp. 11–15 (cit. on p. 64).

[17] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. Sept. 2018 (cit. on p. 77).

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. Dec. 2020 (cit. on pp. 9, 26, 78).

[19] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. July 2022 (cit. on pp. 13, 72).

[20] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. *Score-based Generative Modeling of Graphs via the System of Stochastic Differential Equations*. June 2022 (cit. on pp. 10, 76).

[21] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. Feb. 2017 (cit. on p. 71).

[22] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 2017 (cit. on p. 73).

[23] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. Feb. 2017 (cit. on p. 5).

[24] Zhifeng Kong et al. *DiffWave: A Versatile Diffusion Model for Audio Synthesis*. Mar. 2021 (cit. on p. 76).

[25] Y. Tina Lee et al. "A Classification Scheme for Smart Manufacturing Systems' Performance Metrics." In: *Smart and Sustainable Manufacturing Systems* 1.1 (Feb. 2017), pp. 52–74 (cit. on p. 3).

[26] Qimai Li, Zhichao Han, and Xiao-Ming Wu. *Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning*. Jan. 2018 (cit. on p. 77).

[27] Eliane Maria Loiola et al. “A survey for the quadratic assignment problem.” In: *European Journal of Operational Research* 176.2 (Jan. 2007), pp. 657–690 (cit. on p. 3).

[28] Amine M’Charrak et al. “Connected Causal Graphs for Real-World Science.” In: () (cit. on p. 47).

[29] Félix Marcoccia et al. “NetDiff: Graph Diffusion with Improved Global Capabilities to Generate and Update Mobile Network Topologies.” In: () (cit. on pp. 14, 15).

[30] Dominic Masters and Carlo Luschi. *Revisiting Small Batch Training for Deep Neural Networks*. Apr. 2018 (cit. on p. 72).

[31] Benoit Montreuil. “A Modelling Framework for Integrating Layout Design and flow Network Design.” In: *Material Handling ’90*. Ed. by Robert J. Graves et al. Berlin, Heidelberg: Springer, 1991, pp. 95–115 (cit. on p. 4).

[32] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. Feb. 2021 (cit. on pp. 30, 78, 79).

[33] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Dec. 2019 (cit. on p. 64).

[34] Yunsheng Shi et al. *Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification*. May 2021 (cit. on pp. 38, 40, 77).

[35] Martin Simonovsky and Nikos Komodakis. *GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders*. Feb. 2018 (cit. on p. 6).

[36] Françoise Fogelman Soulié and Jeanny Hérault, eds. *Neurocomputing: Algorithms, Architectures and Applications*. Berlin, Heidelberg: Springer, 1990 (cit. on p. 43).

[37] Clement Vignac et al. *DiGress: Discrete Denoising diffusion for graph generation*. May 2023 (cit. on pp. 10, 12, 14, 31, 38, 71, 72, 76–78).

[38] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks.” In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24 (cit. on p. 5).

[39] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* Feb. 2019 (cit. on p. 70).

[40] Lantao Yu et al. *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. Aug. 2017 (cit. on p. 8).

A

Discrete Event Simulation Dynamics

This appendix details the internal mechanics of the validation environment used to label the generated datasets. The simulator operates on a time-stepped logic, resolving the state of every node at discrete intervals t. The simulation runs for a total duration $T_{sim} = 1000$ time units, during which the system state is processed to extract the ground-truth performance metrics: System Throughput and Total Energy Consumption.

A.1 Physical State Definitions

Each active processing node (Machine, Assembly, Disassembly) is modeled as a stochastic server governed by a Finite State Machine (FSM). To accurately capture the system dynamics and identifying bottlenecks, the framework distinguishes between four mutually exclusive physical states:

- **Processing:** the node is actively transforming a unit. This state is entered only when input material is available and the machine is free. The duration τ of the Processing state is stochastic and sampled from an exponential distribution $P(\tau) \sim \text{Exp}(\lambda)$. The mean processing times ($\mu = 1/\lambda$) for each unit type are defined as follows:
  - **Machine**: 12.0 time units
  - **Assembly**: 9.0 time units
  - **Disassembly**: 10.5 time units
  - **Buffer**: 0.0 time units (items are transferred instantaneously)
- **Idle:** the node is operational and empty, but inactive due to a lack of system demand or initialization. It represents a baseline availability state where the

resource is waiting for a trigger condition.

- **Starved:** the node is free and ready to process, but strictly prevented from starting a new cycle due to the emptiness of one or more input buffers. While physically similar to Idle, this state specifically highlights an inefficiency caused by upstream supply failures.
- **Blocked:** the node has completed a task but cannot discharge the finished item because the destination buffer is saturated. Unlike Starvation, the resource here is physically occupied by the workpiece, rendering it unavailable for new tasks regardless of input availability.

A.2 Material Flow and Routing Logic

The movement of units through the network is strictly governed by the graph topology $G = (V, E)$. Because the layouts are generated stochastically, nodes often have multiple incoming or outgoing connections. To manage these flows, the simulator uses a Positional Priority Protocol. This logic is deterministic and depends entirely on the numerical index assigned to each node in the graph.

The simulation resolves node states sequentially, following the order of their indices ($i = 0, 1, \ldots, N$). This means that nodes with lower indices have a static priority when claiming resources from shared buffers. This ensures that the simulation behavior is consistent and repeatable.

A.2.1 Output Allocation

When a processing node i completes a task, it must discharge the processed item into downstream buffer. Such an allocation strategy depends on the node type:

- **Standard Routing (Machine and Assembly)**: the node attempts to discharge the item in the adjacent buffer. If available space is found, the item is moved. If the connected buffer is full, the node enters the Blocked state.
- **Split Routing (Disassembly)**: the node must discharge distinct components into all its connected output buffers simultaneously. If even a single downstream path is blocked, the entire discharge process is halted, and the node remains blocked. This enforces a strict 1-to-N splitting logic.

A.2.2 Input Acquisition

Before transitioning from Idle to Processing, a node must pull materials from upstream buffers. This follows a ”pull” logic, where the machine initiates the request:

- **Standard Acquisition (Machine and Disassembly)**: the node scans its input buffer and extract a single unit from the first non-empty space. If no items are present, the node enters in Starvation state.
- **Joint Acquisition (Assembly)**: the Assembly node requires a strict synchronization of inputs. It scans all connected upstream buffers and initiates production only if a unit can be drawn from each one simultaneously (AND-logic).

A.3 Energy Consumption Model

The total Energy Consumption of the system is calculated by integrating the power profile of each node over the simulation horizon. Although four physical states are defined, the energy model simplifies them into two consumption profiles: Active (during Processing) and Idle (during Idle, Starved, or Blocked states). Table A.1 details the specific coefficients used in the experimental campaign.

Unit Type	Processing Consumption (P_{active})	Idle Consumption (P_{idle})
Machine	0.050	0.010
Buffer	0.000	0.000
Assembly	0.020	0.002
Disassembly	0.020	0.002

Table A.1: Energy consumption per time unit for each operational unit type based on state (Active/Idle).

Consequently, the Total Energy Consumption E is computed as the summation of the state-dependent power rates over all non-buffer nodes $i \in V \setminus V_B$:

$$E = \sum_{t=0}^{T_{sim}} \sum_{i \in V \setminus V_B} \left(\mathbb{I}(S_i(t) = \text{Processing}) \cdot P_{active}^{(i)} + \mathbb{I}(S_i(t) \neq \text{Processing}) \cdot P_{idle}^{(i)} \right) \cdot \Delta t$$

where $\mathbb{I}(\cdot)$ is the indicator function, $S_i(t)$ represents the state of node i at time t, and $P^{(i)}$ denotes the specific power coefficient for the unit type of node i.

B

Technical Specifications

B.1 Sinusoidal Positional Encoding

To provide a precise mathematical foundation, the Sinusoidal Positional Encoding is defined as a deterministic mapping that transforms a generic scalar input $x \in \mathbb{R}$ into a high-dimensional vector space $\mathbb{R}^D$. This technique allows neural networks to perceive the precise magnitude of a scalar value through a set of periodic basis functions.

The Encoding Mechanism

Given a generic scalar x and a target embedding dimension D (where D is an even integer), the Sinusoidal Encoding function $\Psi : \mathbb{R} \to \mathbb{R}^D$ is defined by the following mapping:

$$\Psi(x) = \left[\sin(\omega_0 x), \cos(\omega_0 x), \sin(\omega_1 x), \cos(\omega_1 x), \ldots, \sin(\omega_{\frac{D}{2}-1} x), \cos(\omega_{\frac{D}{2}-1} x)\right]$$

The angular frequencies ω_i are determined by a geometric progression:

$$\omega_i = \frac{1}{10000^{\frac{2i}{D}}}$$

where $i = 0, 1, \ldots, \frac{D}{2} - 1$. This results in a spectrum of wavelengths $\lambda_i = 2\pi \cdot 10000^{\frac{2i}{D}}$ that ranges from 2π to $2\pi \cdot 10000$.

Functional Roles and Properties

- **Multi-Resolution Decomposition:** the frequencies ω_i act as a set of analytical "rulers" with different scales. High-frequency components (small i) oscillate rapidly, providing high sensitivity to minute changes in x, allowing the model to distinguish between very close values. Conversely, low-frequency components (large i) oscillate slowly, capturing the coarse-grained magnitude and providing a stable global context.
- **Dimensionality Expansion and Normalization:** projecting a raw scalar x into $\Psi(x)$ ensures all entries are bounded within the range $[-1, 1]$, preventing gradient instability. Furthermore, it provides the network with D independent features, stretching the scalar information into a dense representation that hidden layers can manipulate more effectively.
- **Linear Shift Invariance:** a key property is that for any fixed offset Δ, the encoding $\Psi(x + \Delta)$ can be represented as a linear transformation of $\Psi(x)$. Mathematically, there exists a transition matrix M_Δ such that:

  $$\Psi(x + \Delta) = M_\Delta \Psi(x)$$

  This allows the neural backbone to learn relative relationships between different inputs, as the "distance" between values is encoded in the phase shift of the sinusoids.

Application in the Framework

In the context of this research, this mechanism is applied to scalars such as the diffusion timestep. While the raw value indicates a state, the Sinusoidal Encoding provides the temporal geometry necessary for the network to understand the specific noise regime it is operating in, ensuring that the learnable weights can effectively align the scalar signal with the complex graph topology.

B.2 Detailed Analysis of Adam Moments

The Adam (Adaptive Moment Estimation) optimizer is utilized to handle the non-convex and irregular loss landscape of the graph generation problem. The core effectiveness of this strategy lies in its ability to adapt the learning rate for each individual parameter by tracking two moving averages of the gradients.

Mathematical Formulation of Moments

At each optimization step n, the raw gradient g_n is used to update the first and second moments using exponential decay coefficients β_1 and β_2:

- **First Moment (Momentum):** this term acts as an internal "inertia" for the optimization process. By maintaining a running average of past gradients, m_n smooths out the update trajectory, allowing the model to bypass small local minima and accelerate in directions where the gradient is consistent.

  $$m_n = \beta_1 m_{n-1} + (1 - \beta_1) g_n$$

  The coefficient β_1 (typically set to 0.9) dictates the memory horizon; a higher value implies that the optimizer relies more on the historical direction than on the instantaneous gradient.
- **Second Moment (Adaptive Scaling):** this term tracks the uncentered variance of the gradients (v_n). It measures the "fluctuation" or volatility of the signal for each weight.

  $$v_n = \beta_2 v_{n-1} + (1 - \beta_2) g_n^2$$

  The coefficient β_2 (typically set to 0.999) serves as the volatility memory factor. It regulates the sensitivity of the adaptive scaling mechanism, ensuring that the learning rate is adjusted based on a long-term average of gradient variance. By doing so, it prevents the optimizer from overreacting to isolated outliers or transient noise within a mini-batch.
  This mechanism creates a damping effect that stabilizes updates in directions characterized by high curvature or sharp discontinuities. When a parameter's gradient exhibits wild oscillations, v_n increases, which subsequently reduces

the effective step size for that specific weight. This adaptive behavior is crucial for maintaining numerical stability, particularly when navigating the irregular gradients caused by strict engineering constraints.

Bias Correction Logic

Since the moments m_n and v_n are initialized as zero vectors, they are naturally biased toward zero, particularly during the initial iterations of the training process. This is because the decaying sums have not yet accumulated enough gradient information to represent the true distribution. To counteract this effect, the algorithm calculates bias-corrected estimates:

$$\hat{m}_n = \frac{m_n}{1 - \beta_1^n}, \qquad \hat{v}_n = \frac{v_n}{1 - \beta_2^n}$$

By dividing the raw moments by $(1 - \beta^n)$, the optimizer compensates for the initialization at the origin. As n increases, the terms β_1^n and β_2^n rapidly vanish, and the corrected estimates $\hat{m}_n, \hat{v}_n$ converge toward the actual moving averages. This ensures that the early stages of training, which are critical for establishing the backbone's stability, are not hindered by artificially small update steps.

B.3 Model Architecture and Training Configurations

The following table summarizes the key structural hyperparameters and training configurations discussed in Section 4.4. These values were selected to ensure consistency and comparability between the different manufacturing system typologies, providing a stable foundation for the generative process.

Parameter	Value
Epochs	600
B	16
η	2×10^{-4}
p_{uncond}	0.2
β_1	0.9
β_2	0.999
d_{emb}	16
d_{model}	64
n_{head}	4
Z	2
T	500
s	0.008

Table B.1: Summary of architectural and training parameters.

B.4 Composite Loss Function Parameters

The hyperparameter configuration of the Composite Loss Function is the result of a dedicated calibration process aimed at balancing structural compliance with generative diversity. Each λ parameter was iteratively refined to weigh the different components of the loss, ensuring that the guidance signal effectively steers the model toward valid and high-performing topologies without compromising the exploration of the design space.

Hyperparameter	Value
λ_{KL}	0.1
$\lambda_{semantic}$	1.0
λ_{mask}	1.0
λ_{system}	2.0
$\lambda_{connectivity}$	0.2

Table B.2: Numerical values for the Composite Loss Function weighting parameters.

The calibration process led to the identification of a robust set of hyperparameters that balance the competing objectives of the composite loss function. These specific values for λ_{KL}, $\lambda_{semantic}$, λ_{system}, and $\lambda_{connectivity}$ were applied uniformly across all system topologies, including Open, Closed, Input-Only, and Output-Only configurations. This transversal approach was chosen to maintain a consistent evaluation framework, ensuring that any observed variations in performance or structural validity are attributable to the topological constraints of the specific system type rather than to variations in the tuning intensity of the generative guidance.

List of Symbols

Variable	Description
$G(V, E)$	Graph representation of industrial layout with nodes V and edges E.
N	Total number of nodes defined in the graph topology.
$\mathcal{C}$	Set of functional node types (Machine, Buffer, Assembly, Disassembly) or edge classes (presence or absence).
S_{in}, S_{out}	Sets of input source nodes and output sink nodes.
T_{sim}	Total duration of the Discrete Event Simulation (DES).
T_{eff}	Effective simulation duration used for steady-state Throughput calculation.
$X_{open}, X_{closed}, X_{in}, X_{out}$	Specific Throughput formulations for each of the four system typologies.
$n_v(t)$	Cumulative count of parts collected by sink v up to time t.
$ops_v(t)$	Total operations completed by node v up to time t.
V_B	Subset of nodes belonging to the Buffer category.
C_{tot}	Total initial inventory capacity of the manufacturing system.
$P_v(t)$	State-dependent power consumption rate of node v at time t.
X, E	System-level metrics: Throughput and Total Energy Consumption.
S	Multi-objective scalar score function balancing X and E.

Continues on next page...

[1]

Variable	Description
w_X, w_E	Weighting coefficients for System Throughput and Total Energy Consumption in the Score function.
x_{node}, e_{node}	Normalized Throughput and Energy Consumtpion contribution at the single-node level.
N_{act}	Number of active processing nodes (Machines, Assembly, Disassembly).
X_{norm}, E_{norm}	Standardized Z-scores for Throughput and Energy Consumption.
μ, σ	Empirical mean and standard deviation of metrics in the reference dataset.
$G_0^{(k)}(V_0^{(k)}, E_0^{(k)})$	Ground-truth (original) layout of the k-th graph sample.
$\tilde{t}$	Sampled diffusion timestep within the Diffusion Horizon T.
$G_{\tilde{t}}^{(k)}(V_{\tilde{t}}^{(k)}, E_{\tilde{t}}^{(k)})$	Partially corrupted (noised) of the k-th graph sample at diffusion step $\tilde{t}$.
T	Diffusion Horizon.
s	Small temporal offset used in the Cosine Noise Schedule.
$\alpha_{\tilde{t}}, \beta_{\tilde{t}}$	Cumulative signal retention and marginal corruption probabilities.
$v_0^{(i)}, e_0^{(i,j)}$	Ground-truth categorical state for node i and edge (i, j).
$v_{\tilde{t}}^{(i)}, e_{\tilde{t}}^{(i,j)}$	Noised categorical state for node i and edge (i, j) at step $\tilde{t}$.
$c^{(k)}$	Raw performance score of the k-th graph, projected into d_{emb} dimensions.
$\bar{c}^{(k)}$	Processed score conditioning of the k-th graph (actual or null sentinel -1.0) for CFG.
θ	Trainable parameters and weights of the Graph Transformer backbone.

Continues on next page...

[1]

Variable	Description
$V_{\tilde{t}}^{(k)}$	Categorical feature matrix for the nodes of the k-th graph at the diffusion step $\tilde{t}$.
$\mathbf{H}^{(k)}$	Unified feature matrix of the k-th graph embedding node types, time, and performance signals.
d_{emb}	Dimension of the latent space for time and score projections.
W_t	Learnable projection matrix for temporal sinusoidal encodings.
$\mathbf{v}_{time}^{(k)}, \mathbf{v}_{score}^{(k)}$	Dense feature vectors in $\mathbb{R}^{d_{emb}}$ representing time and score signals of the k-th graph.
d_{node}	Dimension of the node feature vector (one-hot encoding of 4 types).
$h_i^{(k)}$	Latent representation of node i within the Transformer layers for the k-th graph.
d_{in}	Total input dimension of the backbone ($d_{node} + 2 \cdot d_{emb}$).
$E_{\tilde{t}}^{(k)}$	Adjacency matrix representation of the k-th graph at diffusion step $\tilde{t}$.
$\epsilon_\theta(\mathbf{H}^{(k)}, E_{\tilde{t}}^{(k)})$	Neural network prediction of the noise for the k-th graph.
$p_\theta(G_0 \mid G_{\tilde{t}}, \tilde{t}, \bar{c})$	Learned reverse diffusion transition probability.
L_V, L_E	Predicted categorical logits for node types and edge existence.
W_1, W_2	Weight matrices for the linear projections in the attention mechanism.
$\mathcal{N}(i)$	Local neighborhood of node i in the layout graph.
$\mathcal{A}_{i,j}$	Attention coefficient acting as a semantic gate between node i and j.
L	Number of stacked Transformer layers in the backbone architecture.

Continues on next page...

[1]

Variable	Description
$\mathcal{L}_{total}^{(k)}$	Total multi-objective loss function for the k-th sample.
$\mathcal{L}_{reconstruction}^{(k)}$	Combined loss for node classification and edge prediction for the k-th sample.
$\mathcal{L}_{node}^{(k)}$	Categorical Cross-Entropy loss for node type reconstruction for the k-th sample.
$\mathcal{L}_{edge}^{(k)}$	Weighted Binary Cross-Entropy loss for adjacency reconstruction for the k-th sample.
$\mathcal{L}_{regularization}^{(k)}$	KL-divergence term for distribution alignment for the k-th sample.
$\mathcal{L}_{semantic}^{(k)}$	Global semantic loss enforcing engineering validity for the k-th sample.
$\mathcal{L}_{mask}^{(k)}$	Penalty for forbidden structural connections (e.g., self-loops) for the k-th sample.
$\mathcal{L}_{system}^{(k)}$	Penalty for non-compliance with system-specific flow boundaries for the k-th sample.
$\mathcal{L}_{connectivity}^{(k)}$	Spectral penalty ensuring a connected graph via Fiedler value for the k-th sample.
$\mathcal{L}_{batch}^{(n)}$	Mean loss aggregated over a training mini-batch for the k-th sample.
λ_{KL}	Weighting coefficient for the statistical regularization term.
$\lambda_{semantic}$	Global weighting factor for all semantic constraint terms.
λ_{mask}	Local weight for the forbidden connection penalty.
λ_{system}	Local weight for the system type compliance penalty.
$\lambda_{connectivity}$	Local weight for the graph fragmentation penalty.
$\tilde{\mathbf{v}}_{0,i}^{(k)}$	Ground-truth node types for node i.
$\mathbf{e}_{0,ij,k}$	Ground-truth edge existence between node i and j.

Continues on next page...

1

Variable	Description
$\mathbf{P}^{(k)}_{V,i,\chi}$	Predicted probability of node i belonging to class χ.
$\mathbf{P}^{(k)}_{E,ij,k}$	Predicted probability for edge existence between node i and j.
w_{pos}	Positive class weight for handling edge sparsity in Binary Cross-Entropy loss.
$\mathbf{P}^{(l)}_{dataset}$	Empirical marginal probability for the Regularization loss.
$\hat{\mathbf{P}}^{(l)}_{batch}$	Predicted marginal probability for the Regularization loss.
$M_{forbidden}$	Binary mask encoding prohibited structural connections.
$d_{in}(i), d_{out}(i)$	In-degree and out-degree of node i.
δ	Small constant for numerical stability and hinge loss margins.
$\mathcal{L}^{(open)(k)}_{sys}, \mathcal{L}^{(closed)(k)}_{sys}$, $\mathcal{L}^{(in)(k)}_{sys}, \mathcal{L}^{(out)(k)}_{sys}$	Specific flow-boundary penalties for system topology for the k-th graph.
λ_2	Second smallest eigenvalue (Fiedler value) of the Graph Laplacian.
τ	Target threshold for connectivity.
m_n, v_n	First and second moment estimates of the Adam optimizer.
$\hat{m}_n, \hat{v}_n$	Bias-corrected moment estimates of the Adam optimizer.
B	Batch Size used during the training phase.
g_n	Stochastic gradient vector at optimization step n.
α	Learning rate for the weight update rule.
β_1, β_2	Decay rates for the Adam moment.
I	Target inventory vector defining the required count of each node type.

Continues on next page...

[1]

Variable	Description
γ	Guidance Scale for Classifier-Free Guidance during sampling.
c_{target}	User-defined Target Performance Score for conditional generation.
$G_T(V_T, E_T)$	Fully noised initial state for the reverse process.
L_{cond}, L_{uncond}	Predicted logits in conditional and unconditional regimes.
$\tilde{L}_{E,ij,k}$	Modified edge logits after applying Classifier-Free Guidance.
$P_{E,ij,k}$	Final edge probability between nodes i and j
m_{max}	Maximum edge probability used in the logit-guided repair mechanism.
$\mathcal{N}_{kept}$	Set of edges preserved by the connectivity-driven repair step.
$\Phi_{valid}(\cdot)$	Indicator function for the topological validity of a graph.
$\bar{G}(V, \bar{E})$	Graph state after the Repair phase.
$\hat{G}(V, \hat{E})$	Final repaired graph after the Connectivity-level Repair phase.
η	Learning Rate for parameter update.
p_{uncond}	Probability of masking the conditioning signal during CFG.
d_{model}	Internal Hidden Dimension of the Graph Transformer layers.
$N_{dataset}$	Total number of samples in the training layout dataset.
n_{head}	Number of Attention Heads in the Multi-Head Attention mechanism.
n_{gen}	Total number of generation attempts in a sampling batch.

Continues on next page...

[1]

Variable	Description
$\mathbb{1}(\cdot)$	Indicator function assigning 1 to true conditions and 0 otherwise.
$\mathrm{WL}(\cdot)$	Weisfeiler-Lehman graph hashing algorithm for structural isomorphism.
$\mathcal{G}_{valid}$	Subset of generated layouts satisfying all topological constraints.
$\mathcal{H}_{train}$	Set of Unique structural hashes extracted from the training dataset.
$S(G^{(k)})$	Actual performance Score obtained via Discrete Event Simulation.
n_{valid}	Number of valid samples produced within a generated batch.

List of Acronyms

Acronym	Description
CFG	Classifier-Free Guidance
CSP	Constraint Satisfaction Problem
D3PM	Discrete Denoising Diffusion Probabilistic Model
DES	Discrete Event Simulation
FLP	Facility Layout Planning
FMS	Flexible Manufacturing Systems
GAN	Generative Adversarial Networks
KL	Kullback-Leibler (Divergence)
KPIs	Key Performance Indicators
LGD	Latent Graph Diffusion
MAD	Mean Absolute Deviation
MLP	Multi-Layer Perceptron
MolGAN	Molecular Generative Adversarial Network
NetDif	Network Diffusion
NetGAN	Network Generative Adversarial Network
NLP	Natural Language Processing
QAP	Quadratic Assignment Problem
SCCs	Strongly Connected Components
SiLU	Sigmoid Linear Unit
SNR	Signal-to-Noise Ratio
UniMP	Unified Message Passing
VAE	Variational Autoencoders
VGAE	Variational Graph Autoencoders
WL	Weisfeiler-Lehman (Algorithm)