universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Towards Enhanced Interpretability in Generative Flow Networks with Deterministic-Heuristic Guidance Regularisation.

verfasst von | submitted by

inz. Piotr Stanisław Fic

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 645
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Data Science
Betreut von | Supervisor:	Assoz. Prof. Dipl.-Ing. Dr.techn. Sebastian Tschiatschek BSc

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Dr. Sebastian Tschiatschek, for his clear guidance, valuable feedback, and steady support throughout my thesis.

I am also grateful to my friends for their encouragement and understanding during this process. Your support helped me stay focused, balanced and motivated in the challenging moments.

Finally, I want to thank my family for their constant support and patience. Their trust in me has always gave me the strength to continue.

Abstract

Generative Flow Network (GFN) is a powerful model for generating diverse, high-reward samples in a sequential construction process. GFN has shown promising results in various domains, including molecule design and combinatorial optimisation. However, its architecture relies on a Deep Learning backbone, making its sampling policy difficult to interpret, potentially limiting its adoption in real-world and high-stakes applications.

This thesis aims to improve the interpretability of GFN. For this purpose, it introduces a dedicated regularisation mechanism to steer the GFN sampling policy toward human-understandable heuristics without modifying the GFN loss function. The method extends the training process by injecting a fraction of heuristic-generated trajectories into the training data and boosting their rewards. The regularisation strength is adjustable by two hyperparameters: the fraction of trajectories to inject and the reward boost factor.

To evaluate the method's impact and effectiveness, two task-agnostic metrics are proposed: the Regularisation Ratio (action agreement with the heuristic) and Solution Intersection (terminal solution overlap). Experiments conducted on three NP-hard graph problems (Maximum Independent Set, Minimum Dominating Set, and Maximum Cut) with a GFN based on a Graph Isomorphism Network reveal systematic patterns that enhance interpretability: high agreement with the heuristic in early stages, followed by mid-trajectory exploration, and frequent late-stage re-alignment.

The regularised GFN can generate solutions that are not only more interpretable, but also occasionally exceed the performance of the canonical GFN on individual instances. Overall, the proposed method provides a way to control the trade-off between agreement with the heuristic and exploration, enhancing the transparency and interpretability of the generation process, while maintaining the GFN's original training objective.

Kurzfassung

Das Generative Flow Network (GFN) ist ein leistungsstarkes Modell zur sequenziellen Erzeugung vielfältiger, hochwertiger Stichproben. GFN hat in verschiedenen Bereichen vielversprechende Ergebnisse gezeigt, darunter Moleküldesign und kombinatorische Optimierung. Seine Architektur basiert jedoch auf einem Deep-Learning-Backbone, wodurch seine Sampling-Strategie schwer zu interpretieren ist, was ihre Anwendung in realen und sicherheitskritischen Szenarien potenziell einschränken kann.

Diese Arbeit zielt darauf ab, die Interpretierbarkeit von GFN zu verbessern. Zu diesem Zweck wird ein spezieller Regularisierungsmechanismus eingeführt, der die GFN-Sampling-Strategie in Richtung menschlich nachvollziehbarer Heuristiken lenkt, ohne die Verlustfunktion des GFN zu verändern. Die Methode erweitert den Trainingsprozess, indem ein Teil heuristikgenerierter Trajektorien in die Trainingsdaten eingespeist und ihre Belohnungen verstärkt werden. Die Stärke der Regularisierung ist durch zwei Hyperparameter einstellbar: den Anteil der einzuspeisenden Trajektorien und den Verstärkungsfaktor der Belohnung.

Zur Bewertung der Wirkung und Effektivität der Methode werden zwei aufgabenneutrale Metriken vorgeschlagen: das Regularisierungsverhältnis (Übereinstimmung der Aktionen mit der Heuristik) und die Lösungsüberschneidung (Überlappung der Endlösungen). Experimente an drei NP-schweren Graphproblemen (Maximum Independent Set, Minimum Dominating Set und Maximum Cut) mit einem auf einem Graph-Isomorphism-Network basierenden GFN zeigen systematische Muster, die die Interpretierbarkeit verbessern: hohe Übereinstimmung mit der Heuristik in frühen Phasen, gefolgt von Erkundung in mittleren Trajektorienabschnitten und häufige Rückkehr zur Heuristik in späten Phasen.

Das regularisierte GFN kann Lösungen erzeugen, die nicht nur besser interpretierbar sind, sondern in einzelnen Instanzen auch gelegentlich die Leistung des ursprünglichen GFNs übertreffen. Insgesamt bietet die vorgeschlagene Methode eine Möglichkeit, den Kompromiss zwischen Übereinstimmung mit der Heuristik und Exploration zu steuern, wodurch die Transparenz und Interpretierbarkeit des Generationsprozesses verbessert werden, während das ursprüngliche Trainingsziel des GFNs erhalten bleibt.

Contents

List of Tables

List of Figures

List of Algorithms

1. Introduction

1.1. Background

Artificial Intelligence (AI) is currently a rapidly developing field of science, with powerful solutions to problems in various domains. The main idea that laid the foundations for future growth was to create algorithms that can solve problems by learning from data, without explicit rule-based instructions provided by humans. Multi-layer neural networks with a large number of trainable parameters became state-of-the-art solutions for many tasks and created a new sub-field named Deep Learning. Intuitively, these models create an inner representation of the input data, which replaces manually defined feature engineering and preprocessing techniques. Recurrent Neural Networks, such as Long Short-Term Memory [HS97] networks, revolutionised modelling of sequential data, such as text, by enabling the neural networks to incorporate information about the parts of the sequences and relationships between them. The recent prominent advances were initiated by leveraging the power of the Attention mechanism and Transformer architecture [VSP+17]. Attention itself overcomes performance issues of the recurrent models by parameterised learning of the structural relationships from full sequences, without recurrent processing. Based on the Transformer architecture, models like BERT [DCLT19] were invented and trained to learn a generalised domain data representation, in this case, natural language in textual format. Later on, they can be additionally fine-tuned to solve a particular task.

Deep Generative Models explore the generative capabilities of Deep Learning models. One of the initial breakthroughs in this field was the Variational Autoencoder [KW+19] neural network. The core concept of this model is to generate new observations from the so-called latent space, which is the internal embedding of the input data. A different approach, Generative Adversarial Networks [GPAM+20], train two neural networks simultaneously: Generator and Discriminator, allowing them to compete in the form of a zero-sum game. The Generator tries to output samples indistinguishable from the input data distribution, while the Discriminator, in opposition, learns to spot the artificially generated observations. The previously mentioned techniques are well-suited to tasks such as image generation, but in some domains, it's more intuitive to treat the generation task as a sequential process of decisions with a background in Markov Chain Decision Processes. With such definition, the problem can be solved with Reinforcement Learning solutions. The agent, based on the current state of the object and the reward function, chooses the next action to take, for example, in a molecule generation task it can be adding a new atom or bond to the current object. From the Transformer architecture mentioned in the previous section, a family of models called Generative Pre-trained Transformers, with the milestone-setting GPT-3 [BMR+20], evolved in the field of Natural Language Processing. These models sample the

next sequence token based on the previously generated output and the contextual input.

Another innovative approach to generative modelling is Generative Flow Network (GFN) [BJK+21]. This model combines methods from Reinforcement Learning, Deep Learning, and Energy-Based Models. The core functionality of GFN is to learn a sampling policy, which generates samples in a sequential process with a probability proportional to the reward distribution. This policy can be used to sample a diverse set of high-reward observations. The GFN has several positive characteristics. The sampling policy approximates the distribution of the entire reward space, rather than optimising only towards the most rewarding solution. This solves a common problem of collapsing into sampling distribution modes, where only a limited set of high-reward but not diverse samples is produced. Moreover, the GFN framework enables approximated computation of various probabilistic quantities, such as the normalising constant, often intractable in other methods. The obtained sampler is also computationally efficient, as the main cost is the network training time, while the generation happens in a single forward pass of a sequence of actions. The sequential construction process naturally adapts to various domains, including graph data, molecular design, and decision-making processes. Furthermore, it allows modelling problems where various trajectories can lead to obtaining the same terminal state. Enhancing the understanding of GFNs generation process is the primary goal of this research work.

1.2. Problem Statement

The previously introduced complex Generative Models, among them GFNs, come with challenges such as scalability, efficiency, or interpretability. The core building block of GFNs is a deep neural network trained to learn the sampling policy, which is based on an architecture suitable to the task domain, for example, Graph, Recurrent, or Transformer neural networks. These models themselves are complex in terms of the number of parameters, layer design, or objective and activation functions. On top of that, the GFN framework adds the Flow-Network concept and the reward-based objective function, which will be introduced in more detail in the next Chapter 2. Although a layperson can relatively easily understand the sequential generation process of GFNs, the whole architecture places them in a "black-box" category. The lack of interpretability is a major barrier to wide adoption of GFNs in real-world applications.

All methods belonging to the Artificial Intelligence domain should progress towards interpretability of the models [GBY+18], also known under the term Explainable Artificial Intelligence (XAI), because when applied in real-world scenarios they contribute to decision-making processes and might operate in high-impact scenarios, such as medical diagnosis. Deep learning models are currently among the most complex and, at the same time, the hardest to explain. One of the key goals of XAI is to improve understanding of the algorithms and the trust in their solutions. Despite the measurable quality of the models' output, explaining the reasoning behind it is crucial in many applications and desirable in any of them. Taking the medical or banking domain as an example, the prediction should be supported with factors that led to it, such as significant input variables that directed the model towards a particular output. Without interpretations of the predictions, the

usage of AI models might be unsafe and non-compliant with the latest legal regulations. XAI helps to detect possible problems with the models like bias, unfairness, or harmful content generation. Because Machine Learning (ML) is based on historical data, in various cases models might reflect biases and inequalities embedded in the data. Interpreting the model gives more control and understanding to its user and can be used to improve and correct it when not behaving according to expectations.

Generative Flow Networks are a relatively new research direction in the early stages of development. Because of that, the subject of interpretability has not yet been significantly explored in the case of GFNs. Importantly, as stated in previous sections, GFN is a highly complex model which would benefit from research towards its explainability. The interdisciplinary design of GFN makes it more difficult to use any of the existing XAI frameworks. The popular methods, for example LIME, are designed to explain predictions of classical supervised learning models and typically analyse output change in response to perturbation of the input. In contrast, GFN does not make a prediction on a static input, but rather through a sequence of dependent actions, which shifts the challenge to interpreting the entire generation process. As this model was already showcased to be capable of solving problems in domains requiring high safety of the decisions, e.g., drug discovery, the lack of clear understanding of the factors decisive for the shape of sampling trajectories is an obstacle to wide adoption in real-world applications.

1.3. Research Objective

The research objective of this master thesis is to improve interpretability of the GFN model and its generative process. An overview of the research methodology consists of the following steps:

1. **Comparing similarity of GFN solutions to baseline algorithms.** This point aims at developing a task-agnostic method for comparing the similarity of the solution generation process between the GFN and reference algorithms (e.g., heuristic methods).

2. **Regularising GFN.** The motivation for this step is to steer the GFN sampling policy with heuristic algorithms guidance in order to increase the similarity of the generation process to the human-understandable heuristics. With that assumption, regularising GFN is intended to make its solution more interpretable while possibly allowing for weaker performance, as a natural outcome of regularisation.

3. **Evaluating regularisation effectiveness and its outcome on GFN solutions.** In this step, the goal is to assess the impact of the regularisation on the solution trajectories. An expected outcome is a change of the GFN sampling patterns, which could replicate characteristics of the baseline algorithms.

4. **Analysing the impact of the regularisation on the GFN interpretability.** The goal of this step is to assess the contribution of the regularisation toward the

GFN interpretability. The expected outcome is a set of empirical observations and conclusions about directions of further research.

The remainder of this thesis is organized as follows: Chapter 2 reviews the current state of research in the field of GFN. Following this, Chapter 3 describes the proposed methodology and the methods designed to achieve the research objectives. Chapter 4 details the model training procedure, parameter selection, and intermediate results, while Chapter 5 presents the final results within the context of the research objectives. Chapter 6 discusses the impact of the proposed methods on GFNs interpretability, and finally, Chapter 7 concludes the thesis by summarizing the research and its findings.

2. Related Work

This chapter describes research in the field of GFNs related to this thesis, beginning with the initial model invention, its further development and improvements. Next, I describe current applications of GFNs in theoretical and real-world scenarios. Then I continue with the interpretability and Explainable AI (XAI) fields, describing the state of the research for the relevant domains of Deep Learning and Generative Models. Finally, I introduce the research opportunities in XAI for GFNs.

2.1. Generative Flow Network

2.1.1. Model Introduction

GFN was first proposed in [BJK+21]. The authors introduced the model's definition and proved its validity. The GFN was tested on a simple hyper-grid optimisation problem and a more complex real-world application of molecule generation task. [BJK+21] confirmed the advantages of GFN over methods like MCMC, stated current limitations, and in general, opened a new research field of GFNs development.

GFNs generate samples in a sequential construction process. Starting from an initial state s_0, they take a sequence of actions that modify the sample. Each state corresponds to an intermediate state of the sample being generated. The next action is sampled from a set of allowed actions $\mathcal{A} = \{a_0, a_1, ..., a_n\}$ by a forward-sampling policy $P_F(s_{t+1}|s_t)$, that is optimised during the training process. A practical example follows below. The GFN model also enables learning a backwards-sampling policy $P_B(s_t|s_{t-1})$, that, based on the current state of an object, can output plausible actions that led to it. The consecutive object states resulting from GFN actions form a trajectory $\mathcal{T} = (s_0, s_1, s_2, ...)$. Because of the stochastic nature of this process, many trajectories might lead to the same object at the final, so-called terminal state s_T. The construction process ends when GFNs sample a dedicated terminal action or reach a predefined stopping criterion, such as a maximum trajectory length. The final output s_T is scored with a non-negative reward function $R(s_T)$. The reward function can be defined and implemented depending on the application of GFNs. The key aim of GFNs' training process is to obtain a policy P_F that samples objects with a probability proportional to $R(s_T)$, i.e. $\mathbb{P}_{x \sim P_F}(x) \propto R(x)$.

An example of a GFN setting in a particular task of generating drug molecules, using the introduced terminology, is the following. A set of actions may contain additions and removals of atoms and bonds, and the reward function may be a molecule binding affinity score approximated by an external model. The consecutive states in a trajectory would correspond to a partially built molecule.

GFNs use a neural network as their backbone to optimise the stochastic policy P_F. Depending on the task, the sampled object structure (e.g. number of nodes in a graph) differs at each step of the trajectory, hence a suitable neural network must handle inputs of varying shape. Possible choices are, e.g. Graph Neural Networks (GNNs) [SGT+09] or Transformers. The key difference in this aspect from the previously described Deep Generative Models is the fact that GFNs are trained to learn the sampling policy P_F proportional to R and not to optimise the rewards on a finite dataset. This highlights a key difference: while typical Deep Learning models risk overfitting, GFN models may suffer from undertraining. The more iterations of the training process that lead to evaluating various trajectories with R and the more complex network architecture, should lead to more accurate sampling policy estimation. However, typically computational resource constraints and time constraints require a trade-off with a risk of underfitting.

2.1.2. Flow Network and Objective Functions

The novel part of the GFN is a Flow Network-based model of the generation process trajectories. A Flow Network is a graph with sources, sinks and intermediate nodes connected with edges which weights correspond to the amount of flow between nodes. The flow-matching constraint states, that the sum of the flow entering a node must equal the sum of the outgoing flow. The intuitive interpretation of a pipe network that carries water flow is an appropriate analogy for this model. In the GFN context, the Flow Network consists of a single source node (state s_0) and multiple sinks (terminal states). Importantly, the amount of flow reaching each sink equals the value of $R(s_T)$ calculated on the corresponding output object. Defining the flow function from state s to s' as $F(s \to s')$, the forward policy can be defined as $P_F(s'|s) = \frac{F(s \to s')}{\sum_{s''} F(s \to s'')}$.

Naturally for most tasks building an entire Flow Network is not feasible because of its often exponential size resulting from the size of objects and actions space. Hence, various objective functions approximating the original problem were proposed. One of the initial objectives used in [BJK+21] is the Flow-Matching loss of a trajectory τ:

$$L_{\theta,\epsilon}(\tau) = \sum_{s' \in \tau \neq s_0} \left(\log \left[\epsilon + \sum_{s,a:T(s,a)=s'} \exp F_\theta(s,a) \right] - \log \left[\epsilon + R(s') + \sum_{a' \in A(s')} \exp F_\theta(s',a') \right] \right)^2$$

where $T(s,a) = s'$ denotes a transition from state s to s' and $A(s')$ a set of valid actions in a given state. For the stability of neural network training, the model F_θ with set of parameters θ outputs the logarithm of the flow function $F_\theta(s,a) = \log(F(s'|s))$ instead of the policy itself. The epsilon terms ϵ solve numerical issues of logarithms of near-zero values.

The objective function evolved over time. Firstly [BLD+23] introduced the Detailed Balance loss, which avoids one of the summations and indirectly yields the flow-matching

condition by operating on sampling policies, instead of flows. The Detailed Balance loss for a single transition is:

$$L_\theta(s, s') = \left(\log \frac{F(s)P_F(s'|s)}{F(s')P_B(s|s')}\right)^2$$

Related works introduced other improvements like Trajectory Balance loss [MJB+22] or Subtrajectory loss [MRBK+23]. The main goals were to achieve better convergence, a broader diversity of generated samples and improved learning from long trajectories or in large action spaces. Those challenges were also addressed in [MRBK+23], where a sub-trajectory balance objective was formulated. It enables optimising the model on partial sequences from the trajectories and tries to solve the trade-off between a local and global focus of the optimisation process. Another development was made towards enabling learning not only from the terminal states' rewards but also from intermediate states' evaluations. In [PZC+22] a variation named Generative Augmented Flow Network introduced intrinsic motivation to enable learning from an augmented, intermediate state- and edge-based rewards to avoid stucking in single modes of the reward distribution. This solution enables better exploration of a sparse space which results in more diverse sample generation, better convergence and performance. Further advances were made in [PMZB23] where GFN was adapted to utilise explicit intermediate rewards computations in cases such as an additive reward function. This setup allows training even on incomplete trajectories, improves localised optimisation and speeds up convergence. The solution used in [ZDM+23] is a transition-based training, originally proposed in [DGE+22]. In this approach a buffer $B = \{(s, s_{i+1})_k\}_{k=1}^{|B|}$ is sampled from full-length trajectories. It serves as training batch supplier, under an intermediate learning signals loss [PMZB23].

2.1.3. Extensions and Applications

A further contribution from [BLD+23] introduced multiple concepts building on the previously set foundations, such as conditional GFN, continuous action space and more, which extended the framework and properties of the model. The conditional GFN allows conditioning the sampling policy on internal information, e.g. current state of the molecule being generated, or external information, like a target binding for the protein that is constructed by GFN.

To better control GFN training process and balance the exploration and exploitation trade-off, a temperature-conditional version of the model was introduced in [KKZ+23]. This method directly scales the sampling policy's logits and simultaneously learns a family of models that correspond to reward functions raised to different powers. In this way, flexible control of generative policy is enabled by adjusting the temperature parameter. Another extension of GFN presented in [JRHG+23] was a multi-objective version of the model, which according to the authors is particularly useful in tasks like drug discovery or material design, where often the solutions are evaluated by multiple, sometimes conflicting objectives. They showed the effectiveness of the method in generating diverse Pareto-optimal candidates on various tasks like molecule or protein design. This concept was

further developed in [RBPB23], where authors steered the generation of molecules, such that they are placed in their region of interest on multi-objective reward space. This approach again aims to increase the diversity of generated samples and exploration of the Pareto front.

Generative Flow Network is a relatively new model, therefore most of the work has focused on extending and improving the core model formulation and proposing its new variations. However, certain practical applications of GFNs were tested and the performance was compared to the existing state-of-the-art solutions. A vast number of publications come from the biological domain, where tasks such as protein or drug design are naturally suitable to solve with GFN's sequential generative process. In [JBHG+22] several biological sequences design tasks were solved in an active learning setting where GFN played the role of candidates generator. The experiments showed greater diversity and scores of the produced samples than in previously existing approaches. GFN conditioned on external information was used in [SPE23], in the drug design task, where the target binding protein structure was used as the conditional information. The authors concluded that applying GFN outperformed other methods in terms of protein pocket docking score (related to conditional information), but also the novelty of the molecules design and generation time. In other domains, GFN was applied to the maximum likelihood estimation in discrete compositional Latent Variable Model [HMJ+23]. The GFN was used as an approximated sampler of intractable E-step of the Expectation Maximisation framework in grammar induction and image representation tasks. Another usage was proposed in [LJD+23], where GFN served as a dropout mask generator under the assumption of treating the dropout as a latent variable and GFN as the approximation over the posterior distribution of dropout masks. The experiments showed an advantage of GFN over related methods and improved performance on downstream tasks.

The Combinatorial Optimisation domain, consisting of often NP-hard problems, is a good candidate for applying GFN. One of the research works [ZDM+23] approached CO problems with the design of Markov decision processes and training the GFN to sample from the solution space. The NP-hard problems on graphs approached in this paper were: maximum independent set, maximum clique, minimum dominating set, and maximum cut. The state space corresponds to a set of graph vertices and a single action is the addition of a new vertex to it. Because of the use-case-specific long trajectories, authors used a version of transition-based objective, where learning happens on randomly sampled single transitions without knowing full trajectories. They also used the intermediate reward signal method from [PMZB23] mentioned above. The empirical experiments confirmed the advantage of GFN over related learning-based methods, with a specific benefit of diversity of generated solutions.

2.2. Interpretability and Explainability

The next topic highly related to the content of this master thesis is interpretability and explainability of ML models in general and more specifically in the domain of generative models. It is becoming constantly more significant and widely used [GBY+18]. One of the reasons is the growth of models' size and complexity, which leads to more ambiguous and less traceable internal representations of problems, especially in Deep Learning. The models' outputs become difficult to trace back to particular input characteristics, which is why they are commonly referred to as black boxes. This poses a significant challenge, particularly in real-world applications where explanations for AI decisions are highly demanded and in some cases might be required by legal means. To begin with, there is a group of ML models considered to be natively interpretable, such as linear and logistic regression or decision trees [CPC19]. We assume so, because the model's prediction can be supported by an exact link between input data and the model's prediction function. For example, the coefficients of a linear model correspond to the influence of particular input variables. This property has been used to develop a proxy model approach, in which a simpler model is developed to approximate a black box and project the hidden representation of the problem onto an interpretable one. One of the most popular of this kind of method is the LIME framework [RSG16]. It approximates a complex model by building a local linear proxy around the data point of our interest. This allows for finding out about the most significant input variables that influence the prediction's outcome for the observation. The method is also model-agnostic; it can be used for any model because it does not rely on any of its characteristics. In a similar approach decision tree proxies can be used. They provide a meaningful interpretation by the decision splits defined in tree nodes. However, as decision trees can grow significantly in depth the method has limitations in scalability.

The Deep Learning models have their distinguishing characteristic of multiple layer-based architectures. Therefore some interpretability techniques were designed especially for this setting. One of them is called Salience Mapping and relies on repetitive testing of a neural network with partially modified input data to find out which part of the input has an actual influence on the network's output. This method was the most rapidly adapted for Convolutional Neural Networks, where the importance of particular pixels of the image could be examined. The computations are usually based on the changes of values in gradients [STY17] or activation functions [SGK17] concerning the input perturbations. Some of the networks' architectures like attention-based ones, by design, provide a good level of interpretability. The attention weights can be studied to determine which parts of the input had the largest influence on output, in other words, what the network attends to. This is widely used in Natural Language Processing, where attention maps are created to look for the most decisive input tokens, but it can also be applied in Image Processing tasks. Finally, the goal of an explainable prediction can be directly formulated as a training objective for neural networks. In work on image classification [HAR+16], the model was trained to support its prediction with a natural language explanation.

The quick development of Large Language Models enhanced new studies on interpretability methods for those models gathered in [ZCY+23]. Several of them rely on the concept

mentioned previously, such as inner attention mechanism analysis. Adversarial attacks, which by slight input perturbations look for a drastic change in model output. Counterfactual explanations similarly look for a change of input that leads to a prediction of the opposite class, but in this case, the degree of input change should be noticeable to humans. In the domain of Deep Learning for drug design, again the same tools like counterfactuals or gradient-based explanations were adapted by adjusting them to graph environments. Some works tried to implement explainability into training procedures by focusing on certain substructures that lead to highly rewarded molecules [GS23]. In the domain of reinforcement learning, most of the XAI methods were developed more use-case specific and in the tasks of robotic or games agents [HCDR21]. One class of the method is named State Representational Learning, where a meaningful, lower-dimensional state representation is computed to investigate which features of the environment are important for an agent when making actions. Another approach is Hierarchical Reinforcement Learning, where a task is divided into a few sub-agents with specific sub-goals and a main agent that bases its actions on the optimal ones for the sub-agents. This forms a specific form of human-interpretable environment representation. One more idea is a reward function decomposition into several summing-up components with meaningful definitions. In summary, the reinforcement learning domain has not yet developed a model-agnostic XAI method.

In the domain of GFNs, explainability has not been directly addressed yet. Some intermediate results were presented in [JRHG+23] or [RBPB23], but they focused rather on very specific aspects of GFNs, for example, the goal-conditioned GFN might show us some insights into the model's behaviour when changing the point of interest in reward-space. The observations in those papers are mostly relevant for scientific purpose or for domain experts that can derive deep insights useful for the model's usage or adaptation to a specific task. They also do not provide a full picture approach to GFNs interpretability, but are limited to specific parts of the model's behaviour. There is a significant place for improvement in methods that could introduce human understandable interpretations of GFNs solutions concerning the problem domain. The possible directions are incorporating already existing methods to particular parts of GFNs framework or finding novel approaches that would be capable of capturing all nuances of the GFNs mechanics.

3. Methodology

This chapter presents the methodology used in this thesis. The methodology was designed to provide a comprehensive analysis of the GFN model in the context of combinatorial optimisation tasks. The main goal was to understand how the regularisation of the GFN model can affect its performance and how much it can benefit the model's interpretability. The methodology is presented in chronological order of the steps taken to achieve the research objectives. The following sections will explain them in detail.

1. Section 3.1: Reproduction of an unregularised GFN (canonical GFN) solution on the subset of CO problems.
2. Section 3.2: Computation of reference solutions from selected heuristic algorithms for the chosen CO problems.
3. Section 3.3: Regularisation of the GFN to steer its generative process to be more similar to the baseline algorithms.
4. Section 3.4: Comparative study of the canonical and regularised GFN, and the heuristic solutions.
5. Section 3.5: Empirical analysis of the impact of the regularisation on the GFN's interpretability.

3.1. Canonical GFNs on Graph Combinatorial Optimisation Tasks

In the first step, the canonical GFNs solutions were reproduced on the CO problems presented in [ZDM+23]. This research validated the quality and correctness of the GFNs application for these types of tasks. The reproduction ensured that the experimental setting, training process and intermediate solutions were a valid basis for the key experiments of this thesis.

The chosen graph CO problems are examples of NP-hard problems. Typically, they are defined as optimisation problems to find an optimal solution to a given graph and objective function. The considered graphs are undirected and unweighted, as in [ZDM+23]. Using the notation of a graph $G = (V, E)$, where V is a set of vertices and E a set of edges, the problems can be defined as follows:

1. **Maximum Independent Set (MIS)**: The task is to find a subset of vertices $V' \subseteq V$ such that no two vertices in V' are adjacent. The objective is to maximise the size of V'.
$$\max_{I \subseteq V} |I| \quad \text{subject to} \quad \forall u, v \in I, (u, v) \notin E$$

2. **Minimum Dominating Set (MDS)**: The task is to find a subset of vertices $V' \subseteq V$ such that every vertex in V is either in V' or adjacent to a vertex in V'. The objective is to minimise the size of V'.
$$\min_{D \subseteq V} |D| \quad \text{subject to} \quad \forall v \in V \setminus D, \exists u \in D \text{ such that } (u, v) \in E$$

3. **Maximum Cut (MC)**: The task is to find a partition of the vertices V into two disjoint sets S and $V \setminus S$ such that the number of edges between S and $V \setminus S$ is maximised.
$$\max_{S \subseteq V} |E(S, V \setminus S)|$$

The datasets for the problems were generated using the RB Model [XBHL07] for graph generation, a random graph model that creates graphs with controlled structural properties. This model was selected because it produces graphs with varying degrees of complexity while maintaining realistic structural characteristics that are representative of real-world combinatorial optimisation problems. The RB Model generates graphs by randomly connecting vertices with a specified probability, creating instances that are neither too trivial nor impossibly complex to solve by algorithmic methods.

Following the same procedure as in [ZDM+23], the training dataset consisted of 4000 graph instances with vertex counts varying between 200-300. This range was chosen to provide sufficient problem complexity while remaining computationally tractable for GFNs training. The test set consisted of 500 additional graphs generated using the same parameters. The evaluation results across all three combinatorial optimisation problems are presented in Chapter 5.

3.2. Baseline Heuristic Algorithms

For each of the graph problems, a respective heuristic algorithm was selected as a reference solution. The selected algorithms preferably should be:

- **Intuitive:** The algorithms should represent a human intuitive approach to solving the problem, which will benefit the derived interpretability.
- **Deterministic:** Heuristic algorithms often use some form of random search strategy to find an approximated solution. In the context of the comparison to stochastic GFNs, the deterministic nature of the heuristic algorithms is needed for reproduction and meaningful insights. Randomised strategies will not enhance the interpretability.
- **Step-wise:** The algorithms should follow a step-wise procedure, as GFNs do.

- **Efficient:** The heuristics will be used during the training process, therefore they should be relatively fast to compute and implementable in GPU-based Deep Learning frameworks.

Based on the above criteria, the following greedy algorithms were chosen:

1. **MIS:** Greedy Maximum Independent Set, Algorithm 1
2. **MDS:** Greedy Minimum Dominating Set, Algorithm 2
3. **MC:** Greedy Max-Cut, Algorithm 3

The selected algorithms were chosen as arguably the simplest of available heuristic algorithms for the given problems. In all cases, most of the algorithms present in the literature are more complex, for example using randomised strategies, in order to guarantee solutions closer to the optimal one. However, for the given methodology and above-mentioned criteria, the basic greedy algorithms are a better fit.

Input: Graph $G = (V, E)$
Output: Independent Set $I \subseteq V$
$I \leftarrow \emptyset$;
$G' \leftarrow G$;
while G' *is not empty* **do**
  Select $v \in G'$, such that $v = \arg\min_{v \in G'} \text{degree}(v)$;
  $I \leftarrow I \cup \{v\}$;
  Remove v and all its neighbours from G';
end
return I;

Algorithm 1: Greedy Maximum Independent Set (MIS Greedy)

Input: Graph $G = (V, E)$
Output: Dominating Set $D \subseteq V$
$D \leftarrow \emptyset$;
$G' \leftarrow G$;
while G' *is not empty* **do**
  Select $v \in G'$ with the maximum number of uncovered neighbours in G';
  $D \leftarrow D \cup \{v\}$;
  Remove v and all its neighbours from G';
end
return D;

Algorithm 2: Greedy Minimum Dominating Set (MDS Greedy)

Input: Graph $G = (V, E)$, indexed vertices $v_1, v_2, \ldots, v_n$
Output: Partition (S, S')
$S \leftarrow \emptyset$;
$S' \leftarrow \emptyset$;
Place the vertex v_1 in S;
for $i = 2$ ***to*** n **do**
  `// Place` v_i `in the set that maximises the cut`
  if $|E(S, S' \cup \{v_i\})| > |E(S \cup \{v_i\}, S')|$ **then**
    $S \leftarrow S \cup \{v_i\}$;
  end
  else
    $S' \leftarrow S' \cup \{v_i\}$;
  end
end
return (S, S');

Algorithm 3: Greedy Max-Cut Algorithm (MC Greedy)

3.3. Regularisation of GFNs

This section describes the details of the regularisation process applied to the GFN model. The regularisation aim is to make the GFN model generation process more similar to the heuristics. This means, for example, making the same actions given a particular state of the graph or deriving similar solutions. The purpose of regularised GFNs is to obtain models which incorporate heuristic behaviour, that can be further used to derive solutions for interpretability enhancement. The regularisation is designed based on the observation that the GFN is primarily trained to learn the optimal policy P_F that is built on top of the flow network and reinforced by the reward function. In my approach, I did not want to negatively affect the desired properties of the GFN model, for example the relation between the flow network and the sampling policy. This could happen by, for example, modifying the loss function to include a penalisation term for the policy divergence from the heuristic algorithms. Instead, I introduced a positive reinforcement mechanism into the training process. The GFN is trained on-policy, generating new trajectories by sampling. The regularisation was added by including the trajectories sampled from the heuristics with a boosted reward. The goal of this method was to shift the policy distribution from the GFN's native policy to the heuristic policy, while preserving GFN model properties. The process is described in the regularisation step Algorithm 4.

The training methodology follows an on-policy approach where new trajectories are continuously sampled from the currently trained policy version. The trajectory sampling step of Algorithm 4 is based on the GFN model M generating new trajectories from its current forward policy P_F, while the heuristic model H provides reference trajectories from the implemented greedy algorithm. Next, the rewards for the trajectories are computed. The heuristic trajectories are boosted by a factor b, while the GFN trajectories are left unchanged. To ensure that the boosted heuristic trajectories are used as the training data,

a batch is created by randomly sampling the heuristic trajectories, which is then blended with the GFN trajectories in a ratio f. The buffer operation relates to the transition-based training process, see Section 2.1.2, where the buffer of transitions is used as a training batch supplier. In the algorithm, it is updated by the blended training data. The training step is performed by the model M on the updated buffer. The canonical GFN training process is presented in Algorithm 5 for reference.

Notation : Heuristic trajectories T_H, regularised GFN trajectories T_{GFN}, heuristic trajectories rewards R_H, GFN trajectories rewards R_{GFN}
Input: GFN model M, heuristic algorithm H, reward function R, transitions buffer B, training step over batch *train*
Output: Regularised GFN model M_{reg}
Parameters : Heuristic sample fraction f, reward boost factor b
foreach *epoch* **do**
  foreach *batch* **do**
    `// Sample trajectories`
    $T_H \leftarrow H(\text{batch})$;
    $T_{GFN} \leftarrow M(\text{batch})$;
    `// Compute rewards`
    $R_H \leftarrow R(T_H) \cdot b$;
    $R_{GFN} \leftarrow R(T_{GFN})$;
    `// Create training data (trajectory, reward)`
    $D_H \leftarrow \{(T_H[i], R_H[i]) \mid i \in [0, |T_H| - 1]\}$;
    $D_{GFN} \leftarrow \{(T_{GFN}[i], R_{GFN}[i]) \mid i \in [0, |T_{GFN}| - 1]\}$;
    `// Blend heuristic and GFN trajectories`
    $Indices \leftarrow \text{sample}(\{0, 1\}, \text{size} = |D_H|, \text{p} = [f, 1 - f])$;
    $D \leftarrow \{D_H[i] \text{ if } Indices[i] = 0 \text{ else } D_{GFN}[i] \mid i \in [0, |D_H|)\}$;
    `// Update buffer`
    $B \leftarrow B \cup D$;
    `// Train step`
    $M \leftarrow \text{train}(M, B)$;
  end
end
$M_{reg} \leftarrow M$;
return M_{reg};

Algorithm 4: Regularisation of GFN

During the development of the algorithm, I initially experimented only with the ratio parameter f. But the early results indicated that the GFN might need an additional signal to learn from the heuristic trajectories. The reward boost factor b was introduced to encourage the GFN to incorporate the information from the heuristic solutions. More details on the evaluation of the parameters are presented in Chapter 4.

The proposed algorithm was selected for further implementation after considering other

Notation : GFN trajectories T_{GFN}, GFN trajectories rewards R_{GFN}
Input: GFN model M, reward function R, transitions buffer B, training step over batch *train*
Output: trained GFN model $M_{trained}$
foreach *epoch* **do**
  foreach *batch* **do**
    `// Sample trajectories`
    $T_{GFN} \leftarrow M(\text{batch})$;
    `// Compute rewards`
    $R_{GFN} \leftarrow R(T_{GFN})$;
    `// Create training data (trajectory, reward)`
    $D_{GFN} \leftarrow \{(T_{GFN}[i], R_{GFN}[i]) \mid i \in [0, |T_{GFN}| - 1]\}$;
    `// Update buffer`
    $B \leftarrow B \cup D_{GFN}$;
    `// Train step`
    $M \leftarrow \text{train}(M, B)$;
  end
end
$M_{trained} \leftarrow M$;
return $M_{trained}$;

Algorithm 5: Canonical GFN

designs for the regularisation. One of the alternatives was introducing a penalty term to the GFN loss function. However, the GFN objective functions are related to the flow network, therefore connecting the penalty with the heuristic solutions trajectories or actions would not have been straightforward. Early tests of a KL divergence penalty term did not yield satisfactory results. Other methods could involve distillation or adversarial training; those more complex methods were not considered as the taken approach was successful.

3.4. Comparative Study of Solutions

The comparative study of the solutions was designed to derive insights into similarities between the GFN and the heuristics, the effectiveness of the regularisation method and the impact of the regularisation on the GFN's generative process.

Optimality of the Solutions For each of the graph CO problems, there exists an objective function, which is the ultimate benchmark of the solution. Each of the solutions was compared under the objective function value, based on the terminal states of their trajectories. The objective functions were defined in Equation 1, Equation 2 and Equation 3.

Similarity to Canonical GFNs To quantify the similarity of the reference solution to the GFN, access to the GFN's forward sampling policy P_F will be used. Because the

GFN model satisfies the Markovian assumption, the likelihood of a particular trajectory $T = (s_0, s_1, ..., s_n)$ can be computed as a product of the conditional probabilities:

$$P(T) = \prod_{i=0}^{n-1} P_F(s_{i+1}|s_i)$$

Now a likelihood of a reference solution (regularised or heuristic) $P(T_r)$ can be compared to a canonical GFN likelihood $P(T_{GFN})$, which naturally is the highest for the GFN itself. The relative ratio of the likelihoods can be used as a measure of similarity between the solutions.

Regularisation Effectiveness The regularised GFNs can be compared to the canonical version of the model using the likelihood defined in the previous section. In order to further evaluate the regularisation influence on the generative process by comparing it to heuristic algorithms, the following methods were proposed and considered:

1. **Regularisation Ratio (RR):** In a given batch of one-step transitions (related to the transition-based training process, see Section 2.1.2), the ratio of the GFN's actions, which are the same as the heuristic actions.

$$RR = \frac{1}{|B|} \sum_{i=1}^{|B|} \mathbb{1}(a_{\text{GFN}}^{(i)} = a_{\text{H}}^{(i)})$$

   where $a_{\text{GFN}}^{(i)}$ and $a_{\text{H}}^{(i)}$ are the GFN and heuristic actions for the i-th transition in the batch.

2. **Solution Intersection:** For the given final solution, the intersection of the graph nodes belonging to the regularised GFN and heuristic solutions.

$$\text{SI} = \frac{|S_{\text{GFN}} \cap S_{\text{H}}|}{|S_{\text{H}}|}$$

   where S_{GFN} and S_{H} are the sets of nodes in the GFN and heuristic solutions.

3.5. Empirical Analysis of Regularisation Impact

The quantitative comparative study is complemented by an empirical analysis of the regularisation impact on the interpretability of the GFN. The goal is to showcase emerging patterns in the GFN's generative process, which could lead to better interpretability. The empirical analysis will be conducted on selected graph instances from the test set, where the regularised GFN outperformed the canonical version, either in terms of solution quality or in enhanced human-understandable decision patterns. The analysis is conducted through manual comparison of the GFN and heuristic trajectories. The trajectories are compared side by side in tabular form, with additional columns for important properties,

such as the degree of the sampled node. The canonical GFN solution trajectories will be presented for reference. The expected outcome is to find a set of graphs on which the regularised GFN achieves high-reward solutions and preserves decision patterns of the heuristic algorithms. A decision pattern in this context is a sequence of actions taken by the model in a particular order. The discovered patterns will be judged for their potential to enhance the interpretability. This empirical analysis is presented in the Chapter 6.

4. Model Architecture and Training

This chapter details the model training procedure, including parameters selection, evaluation metrics, and model architecture. It also includes some intermediate results used to tune the models.

4.1. Model Architecture

The GFN model architecture is based on the one proposed by [ZDM+23]. For the purpose of this research, it is built upon the Graph Isomorphism Network (GIN), which is a specific type of GNN. The GFN model consists of two separate GIN networks: one modelling the flow function and the other modelling the forward policy. The training process follows an on-policy exploration method with intermediate rewards, generating new trajectories for each batch. The training batches consist of randomly sampled single transitions from these trajectories. The transition-based training was described in the context of the GFN model in Section 2.1.2. The key parameters of the GFN model are presented in Table 4.1. The complete configuration can be found in the code repository [Fic25]. The GFN and GIN model hyperparameters were set based on [ZDM+23].

Table 4.1.: GFN model parameters

Parameter	**Value**
GIN Hidden Layers	5
GIN Hidden Units	256
Dropout	0
Graph Batch Size	64
Transitions Batch Size	30
Transition Buffer Size	10^6
Number of Epochs	10
Optimizer	Adam
Learning Rate	10^{-3}

4.2. Training Procedure

The main goals of the model training were to achieve a similar level of performance measured by the solutions' rewards compared to the original results from [ZDM+23], and to

tune the regularisation mechanism. The latter depends on two introduced hyperparameters from Section 3.3:

1. **Heuristic Sample Fraction** f, which determines the fraction of the trajectories in the batch that are generated by the heuristic policy.

2. **Reward Boost** b, which scales the rewards of the heuristic samples.

Training was conducted on a single T4 GPU with 16GB of memory. The implementation was developed in PyTorch [PGM+19], with significant use of DGL [WZY+19] for the graph operations. The code is available in the code repository [Fic25]. Computation time was a significant limiting factor. The training of a single model on a given task varied from ∼3h on MIS, through ∼7h on MC, to ∼12h on MDS. The MIS task was chosen as a starting point to approximate optimal hyperparameter selection.

4.3. Hyperparameters Selection

The grid search was conducted over the following values: $f \in \{0.05, 0.1, 0.25, 0.5\}$ and $b \in \{5, 10, 50, 100, 1000\}$. Including the canonical GFN ($f = 0$ and $b = 1$), a total of 21 models were trained. The training process for each model candidate from the grid search was monitored by evaluating it on a validation set. The reward was used as the evaluation metric. The training converged after the initial 1-2 epochs depending on the task and fluctuated around the optimal value in the following epochs. The convergence plots from the validation dataset are presented together for all tasks in Figure 4.1. It gives an initial insight into performance with respect to hyperparameter selection. We observe that some of the hyperparameter combinations lead to performance on a similar level to the canonical GFN. The grid search results on the MIS task are presented in Table 4.2 for the Regularisation Ratio, see metric 1, and in Table 4.3 for the Intersection metric, see metric 2. Both metrics were summarised with a set of basic statistics: mean, median, standard deviation, minimum and maximum, computed on the validation dataset after the last epoch of training.

From the experiments on the MIS task, I concluded that:

- The regularisation hyperparameter choice has a significant impact on the regularisation outcome. Depending on the parameter values, the regularisation effect can be either weak (e.g. $f = 0.05$ or $f = 0.10$) or strong (e.g. $f = 0.5$ and $b = 50$). This observation is supported by the values of the regularisation metrics.

- The heuristic sample fraction f has a crucial impact on the regularisation effectiveness. By examining the metrics values for a given f, one can notice that the regularisation strength is not significantly different depending on the boost factor b value, which has less impact.

- The higher values of both f and b lead to stronger regularisation, as expected.

Table 4.2.: Regularization ratio statistics on the MIS validation set.

f	b	mean	median	std	min	max
0.50	**50**	**0.68**	**0.69**	**0.09**	**0.17**	**0.92**
0.25	**1000**	**0.68**	**0.69**	**0.09**	**0.16**	**0.94**
0.25	50	0.67	0.67	0.09	0.11	0.91
0.50	1000	0.67	0.67	0.09	0.17	0.91
0.25	100	0.66	0.66	0.09	0.12	0.91
0.50	10	0.65	0.66	0.09	0.12	0.89
0.50	100	0.65	0.66	0.09	0.14	0.91
0.25	10	0.65	0.66	0.09	0.08	0.89
0.25	5	0.64	0.66	0.09	0.16	0.91
0.50	5	0.64	0.64	0.09	0.16	0.91
0.00	1	0.60	0.60	0.08	0.13	0.86
0.10	10	0.26	0.27	0.07	0.03	0.56
0.10	50	0.26	0.25	0.07	0.02	0.50
0.05	10	0.26	0.25	0.07	0.05	0.53
0.05	50	0.25	0.25	0.07	0.05	0.59
0.05	5	0.25	0.25	0.07	0.03	0.53
0.05	100	0.24	0.25	0.06	0.03	0.47
0.10	100	0.24	0.25	0.07	0.05	0.56
0.05	1000	0.24	0.23	0.07	0.03	0.48
0.10	1000	0.23	0.23	0.06	0.03	0.47
0.10	5	0.23	0.22	0.07	0.03	0.53
0.00	1	0.21	0.21	0.05	0.05	0.44

Table 4.3.: Intersection metric statistics on the MIS validation set.

f	b	mean	median	std	min	max
0.50	**10**	**0.71**	**0.74**	**0.21**	**0.00**	**1.00**
0.50	**50**	**0.71**	**0.74**	**0.21**	**0.00**	**1.00**
0.25	**1000**	**0.71**	**0.73**	**0.21**	**0.00**	**1.00**
0.25	**10**	**0.71**	**0.74**	**0.22**	**0.00**	**1.00**
0.25	**50**	**0.71**	**0.73**	**0.22**	**0.00**	**1.00**
0.50	100	0.70	0.73	0.21	0.00	1.00
0.25	100	0.70	0.72	0.22	0.00	1.00
0.50	1000	0.69	0.72	0.21	0.00	1.00
0.25	5	0.69	0.71	0.21	0.00	1.00
0.50	5	0.68	0.71	0.21	0.00	1.00
0.10	50	0.44	0.44	0.18	0.00	1.00
0.05	50	0.43	0.44	0.18	0.00	1.00
0.05	100	0.43	0.44	0.18	0.00	1.00
0.00	1	0.43	0.44	0.17	0.00	1.00
0.05	5	0.42	0.41	0.17	0.00	1.00
0.05	1000	0.42	0.41	0.18	0.00	1.00
0.10	1000	0.41	0.41	0.18	0.00	1.00
0.10	100	0.41	0.41	0.18	0.00	1.00
0.10	5	0.41	0.41	0.19	0.00	1.00
0.05	10	0.40	0.40	0.18	0.00	1.00
0.10	10	0.40	0.39	0.18	0.00	1.00

The models for the MDS and MC tasks were trained with the following hyperparameters, because of the discovered sensitivity of the regularisation mechanism and computational limitations:

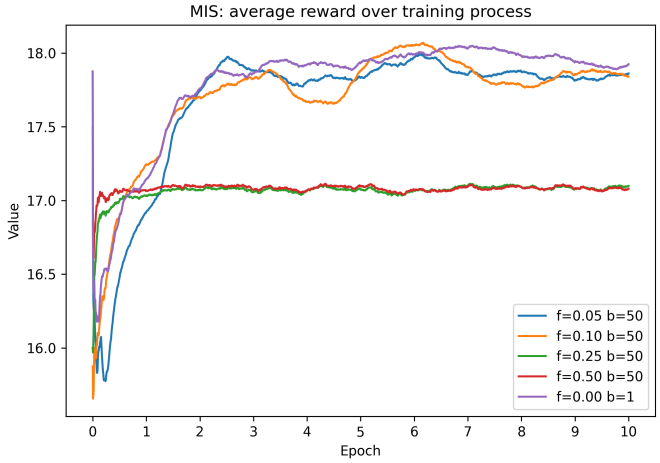
- $f \in \{0.05, 0.1, 0.25, 0.5\}$
- $b \in \{50\}$

The results of these experiments are presented in Appendix A and are consistent with the observations for the MIS task.

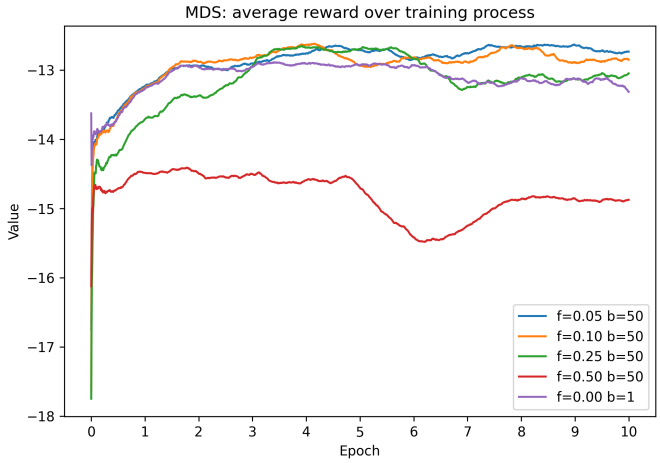
Based on the conclusions from the intermediate results, the final experiments aiming to further interpret the regularisation mechanism and its impact on the GFN model were conducted on the following models for each task:

1. Canonical GFN ($f = 0$ and $b = 1$)
2. Regularised GFN ($f = 0.5$ and $b = 50$)

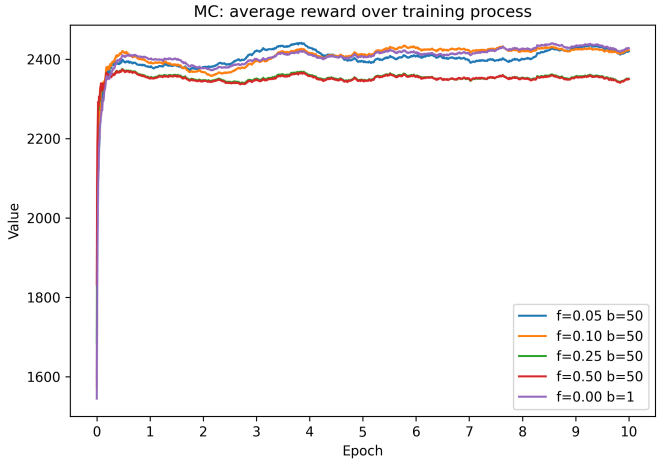
The restriction to the above models was made to simplify the analysis and to focus on the most promising regularisation settings. The final results are presented in the next chapter.



(a) MIS problem



(b) MDS problem



(c) MC problem

Figure 4.1.: Average rewards during the training process on the validation dataset.

5. Results

This chapter presents the final results of the conducted experiments. It is divided into sections dedicated to the quality of the solutions, the trajectory likelihoods and the regularisation effectiveness. Finally, a cross task comparison is presented.

5.1. Quality of the Solutions

The graph datasets used for training, validation and testing consisted of randomly generated examples. Therefore, the exact optimal solutions are unknown. Research by [ZDM+23] suggests that the canonical GFN performs comparably to the SOTA baselines after comparing the performance to a broad range of models including operations research solvers, reinforcement and supervised learning models or heuristics. A key focus point of this section is analysing the distribution of rewards (scores) achieved by the canonical GFN, regularised GFN and greedy heuristics.

Maximal Independent Set Table 5.1 shows that the performance of all three competing methods on the MIS problem was comparable, considering the mean and median scores. While all three models achieved the same maximum score, the regularised GFN and heuristic are distinguished by lower minimum scores. The histograms and boxplots (Figure 5.1) for the MIS problem indicate that the canonical GFN algorithm achieves the highest reward values on average. The mode of the reward distribution for the regularised GFN shifts towards the mode of the heuristic reward distribution and shows comparable performance.

Table 5.1.: Scores summary on the MIS problem.

Algorithm	median	mean	std	min	max
Canonical GFN	18.00	18.00	1.63	14.00	23.00
Regularized GFN	17.00	17.02	1.72	12.00	23.00
MIS Greedy	17.00	17.01	1.72	12.00	23.00

Minimal Dominating Set In this task, the goal, from an optimisation perspective, is to minimise the objective function. Therefore, the scores are the negative of the number of nodes in the Minimal Dominating Set. The results are summarised in Table 5.2. The charts (Figure 5.2) show more significant differences between the models than in the MIS problem. The ranking of model performance remains the same under both mean and median score criteria.

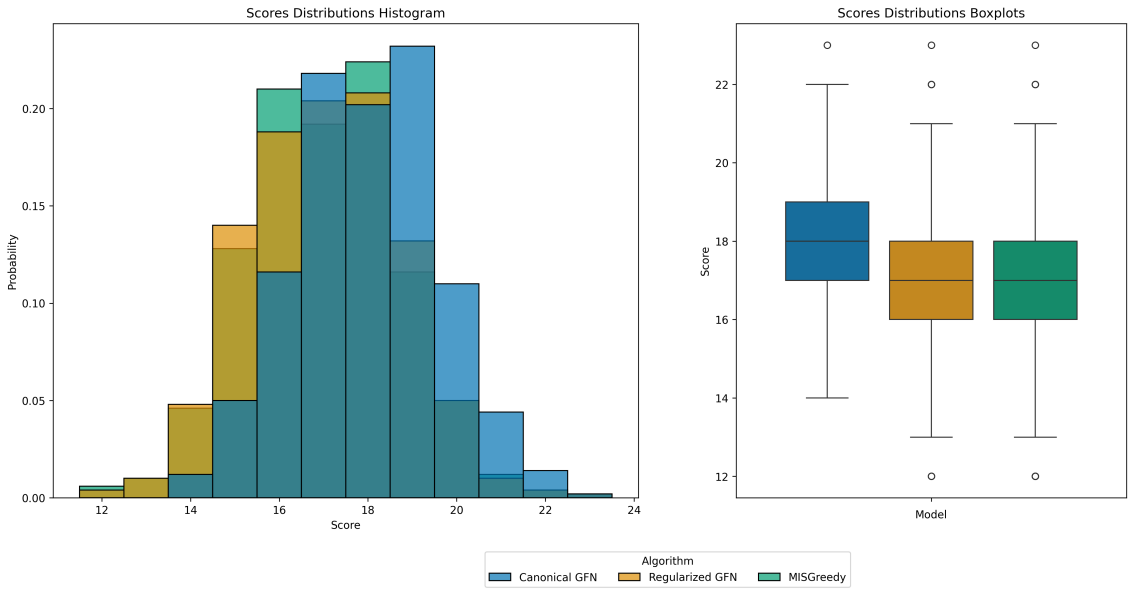


Figure 5.1.: Models scores on the MIS problem.

Table 5.2.: Scores summary on the MDS problem.

Algorithm	median	mean	std	min	max
Canonical GFN	-13.00	-12.80	1.64	-18.00	-8.00
Regularized GFN	-14.00	-14.18	1.87	-20.00	-9.00
MDS Greedy	-16.00	-15.85	1.92	-23.00	-11.00

Maximum Cut Table 5.3 presents the results of the Maximum Cut problem. The observations remain the same as in the previous tasks. The scale of the differences between the models is smaller, but can be clearly seen on the charts (Figure 5.3).

Table 5.3.: Scores summary on the Max Cut problem.

Algorithm	median	mean	std	min	max
Canonical GFN	2429.50	2449.79	662.49	825.00	3912.00
Regularized GFN	2355.00	2366.75	652.52	791.00	3798.00
MC Greedy	2342.50	2347.71	654.12	791.00	3808.00

Summary Across all three graph problems the canonical GFN consistently achieved the highest reward values on average, proving superior problem-solving capabilities compared to both the regularised GFN and greedy heuristic approaches. The regularised GFN showed intermediate performance, with its reward distribution shifting towards the heuristic one.

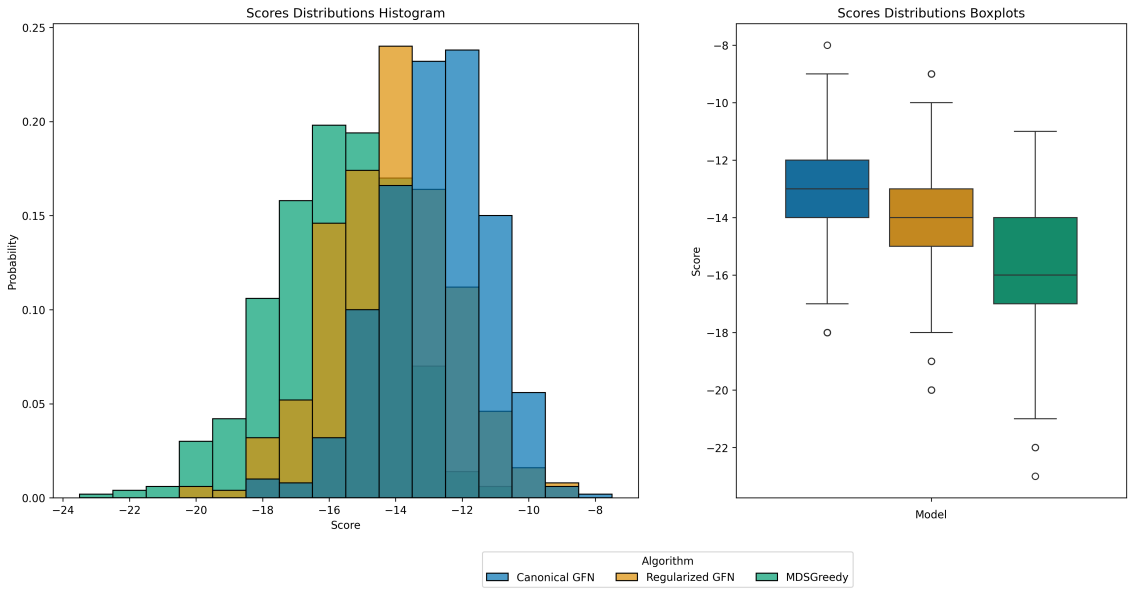


Figure 5.2.: Models scores on the MDS problem.

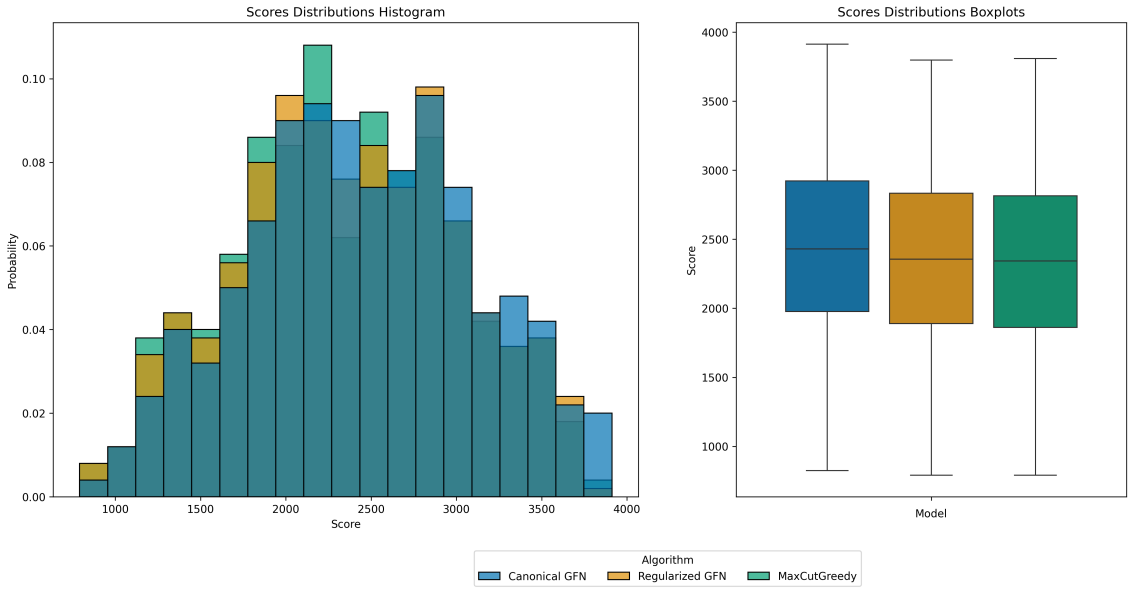


Figure 5.3.: Models scores on the Max Cut problem.

The greedy heuristic consistently achieved the lowest scores on average, but the scale of the differences varied between tasks, with Minimal Dominating Set showing the largest gaps between models. Consistently the maximum scores were the highest for the canonical GFN.

5.2. Trajectory Likelihoods

For this experiment, the three models were used to generate their solution trajectories. Next, the trajectory likelihoods were computed under the canonical GFN policy, see Section 3.4. The goal of this analysis is to investigate the similarity of the regularised GFN and heuristic trajectories to the canonical GFN trajectories and what is the impact of the regularisation in that context. The canonical GFN trajectory likelihoods serve as a reference for the other models, as they are expected to achieve the highest likelihood values under their own sampling policy.

Maximal Independent Set The likelihoods of the trajectories on the MIS problem are summarised in Table 5.4. They confirm the intuitive expectation that the regularised GFN trajectories have higher likelihood values than greedy heuristics under the canonical GFN policy. The details are presented in histograms and boxplots in Figure 5.4.

Table 5.4.: Likelihoods summary on MIS task.

Algorithm	median	mean	std	min	max
Canonical GFN	1647.02	1636.56	639.20	253.37	3214.87
Regularized GFN	1597.04	1568.23	639.06	41.29	3222.52
MIS Greedy	1585.88	1565.22	638.55	-203.06	3222.52

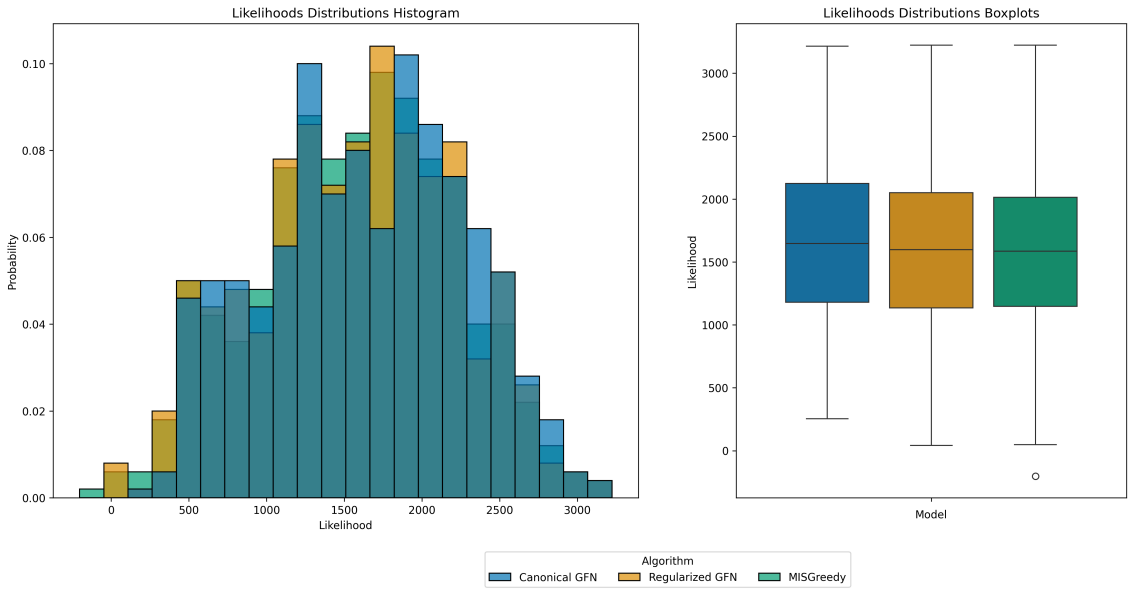


Figure 5.4.: Models likelihoods on the MIS problem.

Minimal Dominating Set The MDS problem is more complex in terms of number of required steps to reach the optimal solution than the first presented MIS problem. This can be a reason for a larger variance in the likelihood distributions between the models. The pattern is similar and we can observe that the regularised GFN has lower likelihoods under the canonical GFN policy, but higher than the greedy heuristic. The results are summarised in Table 5.5. The charts are presented in Figure 5.5.

Table 5.5.: Likelihoods summary on MDS task.

Algorithm	median	mean	std	min	max
Canonical GFN	40681.16	41341.17	3765.43	33112.72	49591.42
Regularized GFN	40239.59	40827.58	3821.28	32976.93	49341.59
MDS Greedy	39471.77	39903.76	3852.78	32034.88	48174.67

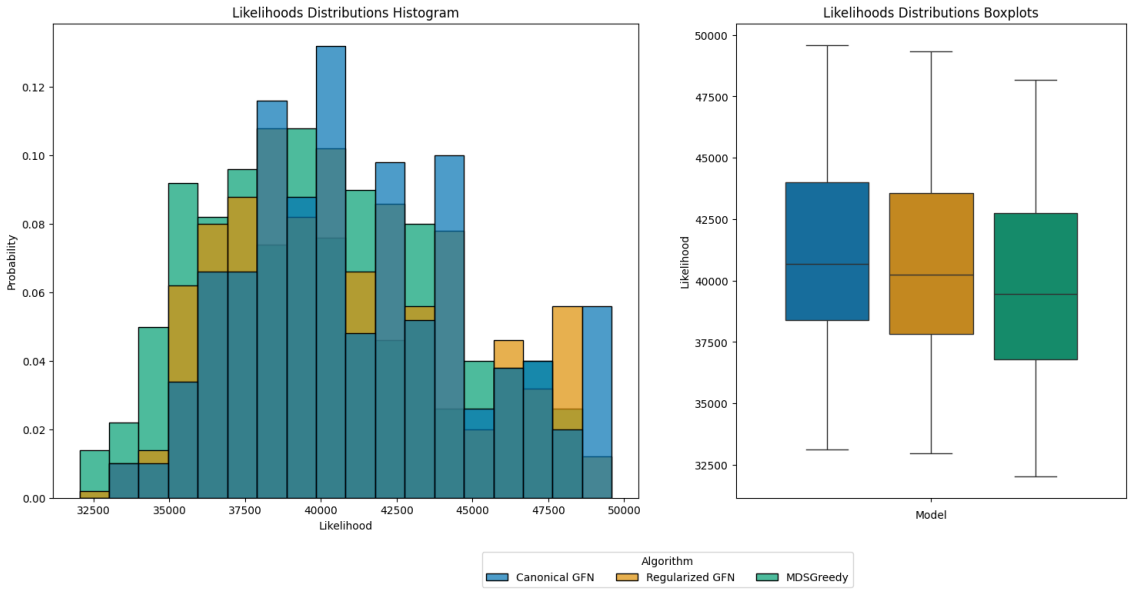


Figure 5.5.: Likelihoods summary on the MDS problem.

Maximum Cut For the Maximum Cut problem, the difference in likelihoods is significant, as presented in Table 5.6. Both the regularised GFN and the greedy algorithm have much lower likelihoods than the canonical GFN for itself. The charts are presented in Figure 5.6.

Summary The results show a consistent pattern across the problems. The regularised GFN reveals an interesting and desired behaviour, as its mode of the likelihood distribution lies between the canonical GFN and the greedy heuristic. The greedy heuristic achieves the lowest likelihoods, reflecting its fundamentally different approach to solution construction.

Table 5.6.: Likelihoods summary on Max Cut task.

Algorithm	median	mean	std	min	max
Canonical GFN	-10283.86	-102299.22	185463.38	-1156001.40	12887.45
Regularized GFN	-44042.24	-141491.65	227248.75	-1597107.84	6160.01
MC Greedy	-49289.95	-145877.80	227919.57	-1597090.13	4766.86

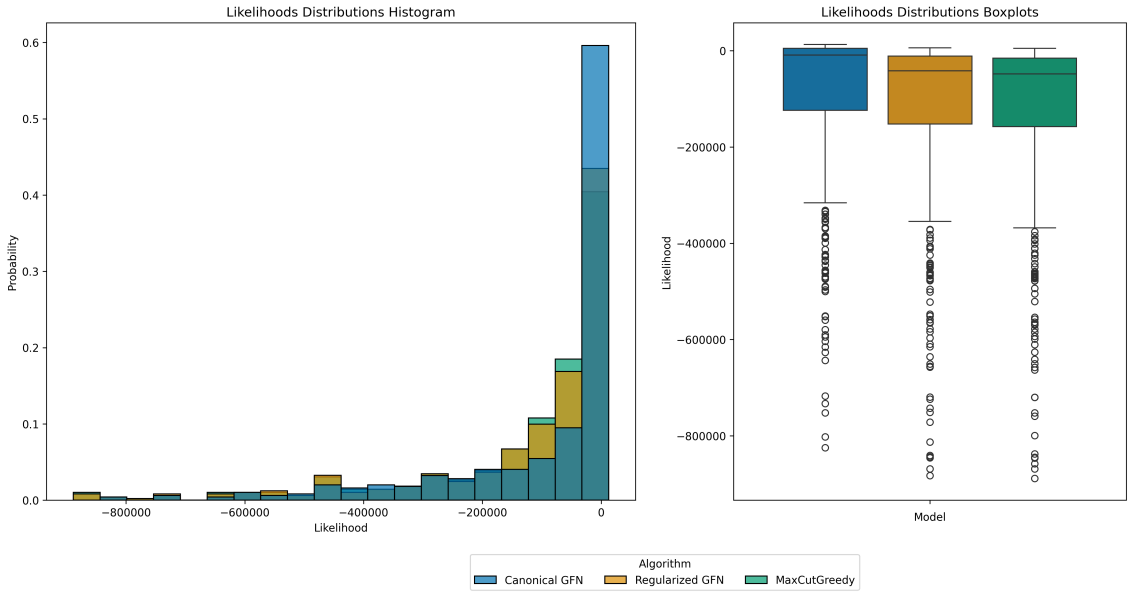


Figure 5.6.: Likelihoods summary on the Max Cut problem.

5.3. Regularisation Effectiveness

This section focuses on analysing the performance of the regularisation mechanism. The regularisation effectiveness is different across the three graph problems. The results are based on the Intersection metric 2, which focuses on the similarity of the final solutions, and Regularisation Ratio metric 1, which focuses on the similarity of the trajectories.

Maximal Independent Set On the MIS problem, a significant influence of the regularisation mechanism can be observed. In terms of the Intersection metric 2, the regularised GFN algorithm yielded much more similar solutions to the greedy heuristic than the canonical GFN. The detailed results are shown in Figure 5.7.

The new behaviour can also be observed in the Regularisation Ratio metric 1. The regularised GFN, for a higher number of actions, follows the same path as the greedy heuristic in the early and middle stages of the trajectory. The detailed results are shown in Figure 5.8.

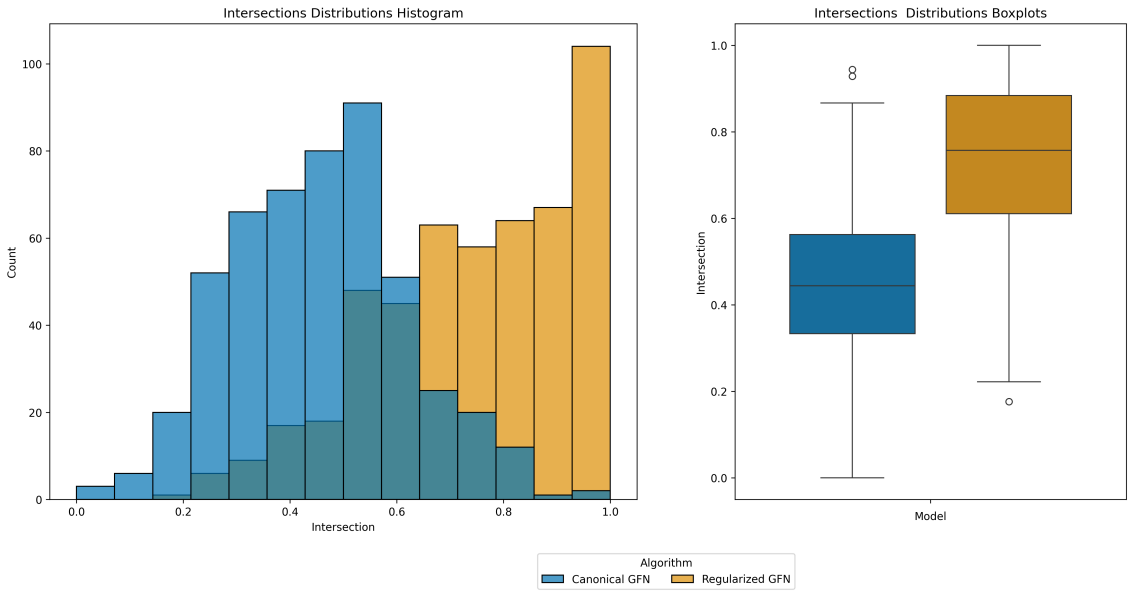


Figure 5.7.: Intersection metrics of the regularised and canonical GFNs solutions on the MIS problem.

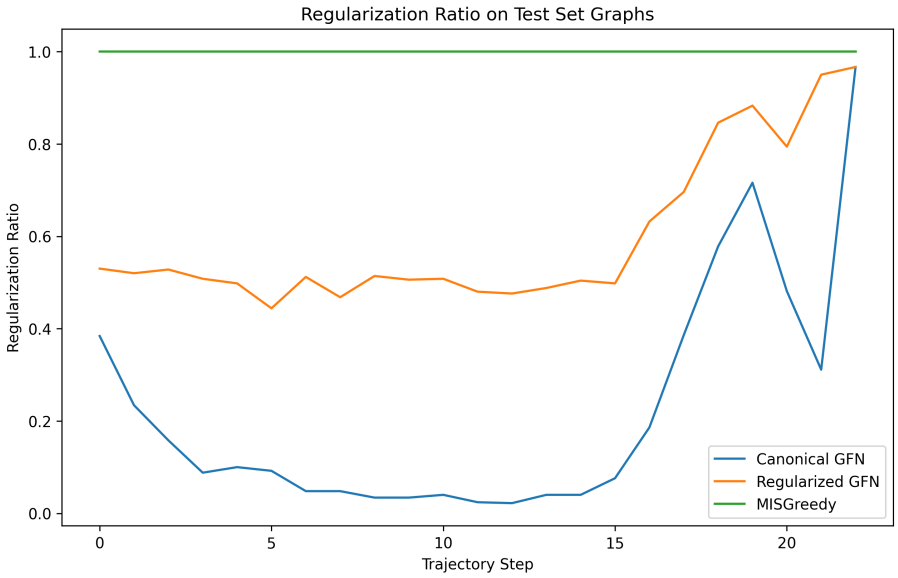


Figure 5.8.: Regularisation Ratio metric on the MIS problem.

Minimal Dominating Set The MDS task is characterised by having the longest trajectories among the three problems and the largest number of nodes in the average solutions. This complexity can be a reason for the regularised GFN to follow the heuristic less strictly and achieve low scores of the considered metrics. The possibility of various trajectories

leading to similar solutions might have a negative impact on accommodating the regularisation mechanism by the GFN model. In Figure 5.9 the Intersection metric scores are slightly higher for the regularised than for the canonical GFN. However, in comparison to the other two tasks, the difference is not as significant.

Moreover, the Regularisation Ratio metric indicates that the regularised GFN algorithm may follow the greedy heuristic even less than the canonical GFN. The details can be seen in Figure 5.10. In the case of trajectories longer than those of the greedy heuristic, the metric value is set to 0, as there are no reference actions to compare to. This can be seen as flat parts of the metric curves on the charts.

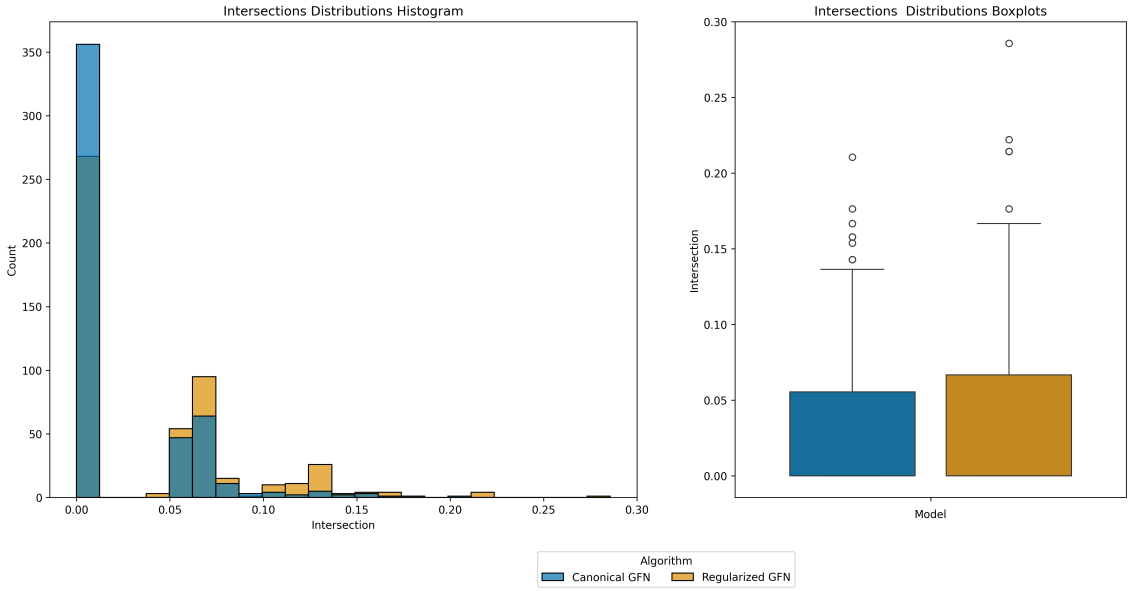


Figure 5.9.: Intersection metrics of the regularised and canonical GFNs solutions on the MDS problem.

Maximum Cut The differences in scores and likelihoods for the Maximum Cut problem did not seem as large as in the other two tasks. On the other hand, the regularisation made the GFN model follow the greedy heuristic very closely. The gain in the Intersection metric is presented in Figure 5.11. This suggests that the regularisation mechanism is particularly effective when the problem has multiple competitive solutions and the GFN can prioritise the consistency with the heuristic.

The Regularisation Ratio metric for the Maximum Cut problem resembles the one from the MIS task. The regularised GFN algorithm follows the greedy heuristic in the early steps, decreases through the middle stages, and then rapidly bounces back and increases at the end. The detailed results are presented in Figure 5.12.

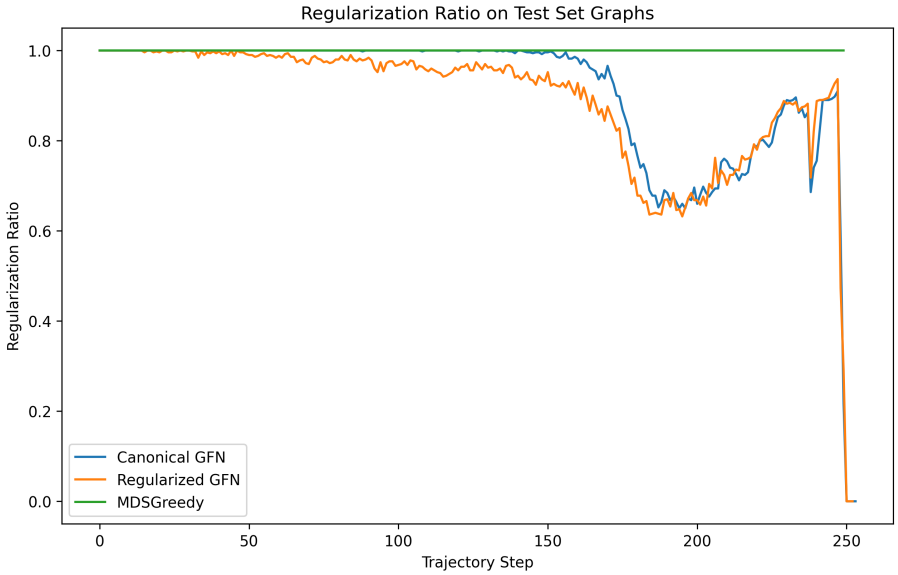


Figure 5.10.: Regularisation Ratio metric on the MDS problem.

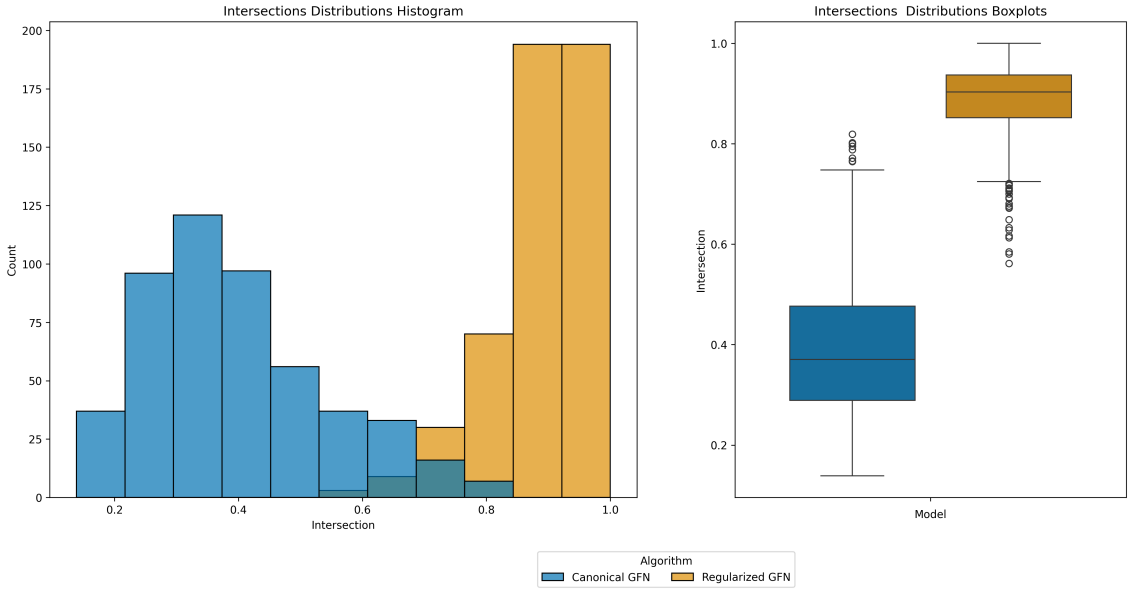


Figure 5.11.: Intersection metrics of the regularised and canonical GFNs solutions on the Max Cut problem.

Summary The Intersection metric reveals that the regularised GFN solutions are systematically more similar to the greedy heuristic than the canonical GFN. The Regularisation Ratio metric analysis indicates that the regularised GFN follows the greedy heuristic more than the canonical GFN as expected. Interestingly, the similarity is higher in the early

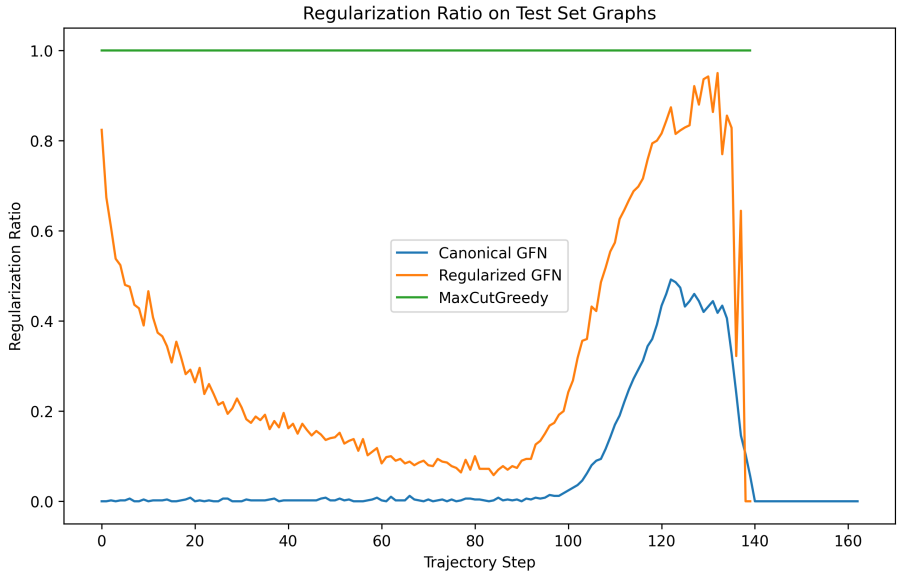


Figure 5.12.: Regularisation Ratio metric on the Max Cut problem.

stages of trajectory construction than in the middle and last phases. The MDS problem shows limited regularisation effectiveness, likely due to the aforementioned task characteristics.

5.4. Cross Task Comparison

This section focuses on relative model comparisons across tasks. To ensure meaningful conclusions, previously defined metrics were normalised with respect to the baseline. The normalised difference is defined as follows:

$$\text{Normalised Metric} = \frac{\text{Metric} - \text{Baseline Metric}}{|\text{Baseline Metric}|}$$

In the case of Score and Likelihood, the baseline was the canonical GFN. For the Regularisation metrics, the baseline was the greedy heuristic. Table 5.7 shows that the discrepancies between the tasks are large. This might be due to the variety of graph problems characteristics and the choice of greedy heuristics. For example, in the MDS task, longer trajectories might have reduced the impact of the regularisation. The relative effectiveness of the regularisation is presented in Table 5.8 and supports similar conclusions. Understanding these differences and their reasons could help in further refinement and tuning of the regularisation mechanism.

Table 5.7.: Relative difference of Likelihood and Scores compared to the canonical GFN.

Metric	Task	Algorithm	median	mean	std	min	max
Likelihood	**MDS**	MDSGreedy	-0.03	-0.04	0.02	-0.08	0.00
	MDS	Regularized GFN	-0.01	-0.01	0.01	-0.06	0.04
	MIS	MISGreedy	-0.05	-0.05	0.17	-1.48	0.59
	MIS	Regularized GFN	-0.04	-0.04	0.16	-0.90	0.69
	Max Cut	MaxCutGreedy	-1.57	-5.72	26.62	-446.62	0.60
	Max Cut	Regularized GFN	-1.36	-4.34	17.15	-311.09	0.84
Score	**MDS**	MDSGreedy	-0.25	-0.25	0.17	-0.78	0.22
	MDS	Regularized GFN	-0.08	-0.12	0.17	-1.00	0.36
	MIS	MISGreedy	-0.06	-0.05	0.06	-0.24	0.12
	MIS	Regularized GFN	-0.06	-0.05	0.06	-0.24	0.12
	Max Cut	MaxCutGreedy	-0.11	-0.11	0.07	-0.34	0.08
	Max Cut	Regularized GFN	-0.11	-0.12	0.07	-0.35	0.07

Table 5.8.: Relative difference of Intersection metric values compared to the greedy heuristic.

Metric	Task	Algorithm	median	mean	std	min	max
Intersection	**MDS**	Canonical GFN	-1.00	-0.98	0.04	-1.00	-0.79
	MDS	Regularized GFN	-1.00	-0.96	0.05	-1.00	-0.71
	MIS	Canonical GFN	-0.56	-0.55	0.16	-1.00	-0.06
	MIS	Regularized GFN	-0.24	-0.26	0.19	-0.82	0.00
	Max Cut	Canonical GFN	-0.63	-0.60	0.14	-0.86	-0.18
	Max Cut	Regularized GFN	-0.10	-0.12	0.08	-0.44	0.00

6. Discussion

In this chapter, I discuss the proposed regularisation method using selected examples. The goal is to illustrate the regularised GFN model behaviour to provide insights into the method's impact on the GFN's interpretability. I also present the limitations and observations important for future improvements and practical applications of the method. For the discussion, I chose to focus on the MIS problem, because of the relatively short trajectories and the simple greedy heuristics, which helps to examine the sampling process empirically.

6.1. Trajectories Analysis

Regularisation Effectiveness in Examples The previous chapter presented the regularisation effectiveness with various metrics on the MIS task. The observations showed that also within the particular task, the regularisation can have different effectiveness. To better understand why this is the case, I focused on examining such examples by analysing the sampling process of the regularised GFN model. First, I want to present a graph for which the Intersection and Regularisation Ratio metrics are in the low range for the MIS task.

Table 6.1 presents a sampled trajectory of the regularised GFN model on the graph number 129 from the test set. This graph was chosen because of a low value of the Intersection metric of approximately 0.18, although the GFN regularisation was configured with the highest considered regularisation strength. The table includes the node sampled by the regularised GFN in each step, the corresponding degree of the sampled node and additionally the reference information on what the greedy heuristic would select. As a reminder, the heuristic greedily selects the node with the lowest degree. The immediate observation is the presence of ties with regard to the heuristic criterion. The regularised GFN model selects different nodes than the greedy heuristic, but with the same degree value. From the GFN perspective, the nodes are equally good, but the greedy heuristic has a deterministic tie-breaking strategy, which was probably not learnt by the GFN model. For comparison, the same data is shown for the canonical GFN in Table 6.2. The scores of both model versions are the same but the trajectories differ. The canonical GFN takes different actions especially in the second half of the trajectory, where it does not sample low-degree nodes. To sum up, the regularisation in this case was actually active, but because of the revealed common ties it could not get high scores of considered metrics. The metrics are defined to distinguish each individual graph node, corresponding to the action of the model, hence selecting a node with the same degree but different node identifier is considered misaligned with the heuristic.

On the contrary, Table 6.3 presents a trajectory with nearly perfect regularisation (Inter-

Table 6.1.: Regularized GFN (f=0.5, b=50) on graph 129 (aligned actions in bold).

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	93	18	0	18
2	154	18	0	18
3	149	18	0	18
4	174	18	0	18
5	132	18	0	18
6	22	18	0	18
7	163	18	0	18
8	7	18	0	18
9	15	18	10	18
10	**33**	**36**	**33**	**36**
11	**60**	**36**	**60**	**36**
12	185	36	74	36
13	75	36	74	36
14	**120**	**36**	**120**	**36**
15	48	56	40	56
16	196	56	112	56
17	**112**	**56**	**112**	**56**

Table 6.2.: canonical GFN on graph 129.

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	168	18	0	18
2	150	18	0	18
3	146	18	0	18
4	174	18	0	18
5	18	18	0	18
6	94	18	0	18
7	20	18	0	18
8	131	18	0	18
9	2	18	0	18
10	**33**	**36**	**33**	**36**
11	51	56	60	36
12	120	36	60	36
13	88	56	60	36
14	75	36	60	36
15	112	56	60	36
16	69	38	60	36
17	**183**	**36**	**183**	**36**
18	**197**	**56**	**197**	**56**

Table 6.3.: Regularized GFN (f=0.5, b=50) on graph 271.

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	**236**	**30**	**236**	**30**
2	**24**	**32**	**24**	**32**
3	92	42	0	42
4	**0**	**42**	**0**	**42**
5	**41**	**44**	**41**	**44**
6	**84**	**46**	**84**	**46**
7	**207**	**52**	**207**	**52**
8	**17**	**56**	**17**	**56**
9	**159**	**58**	**159**	**58**
10	127	60	39	60
11	**39**	**60**	**39**	**60**
12	**74**	**74**	**74**	**74**
13	227	78	187	78
14	**187**	**78**	**187**	**78**
15	**56**	**80**	**56**	**80**
16	**67**	**84**	**67**	**84**
17	218	90	165	90
18	**165**	**90**	**165**	**90**
19	**139**	**102**	**139**	**102**

section metric of 1.0) that follows the greedy algorithm almost in every step. This graph structure is very different from the previous one, because the ties rarely occur. For the canonical GFN, the sampling process deviates from the heuristic and follows its unique path shown in Table 6.4. These selected examples indicate one of the major factors that influenced the regularisation results. In the chosen domain of graph CO problems, the graph structure could cause ties with respect to the heuristic criteria. On the practical side, this will not be a serious limitation and could be resolved by, for example, randomised tie-breaking. One could also modify the regularisation metrics to consider any tie-breaking choice as aligned with the heuristic.

Table 6.4.: canonical GFN on graph 271.

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	24	32	236	30
2	**236**	**30**	**236**	**30**
3	**0**	**42**	**0**	**42**
4	84	46	92	42
5	124	58	92	42
6	95	44	92	42
7	16	62	41	44
8	39	60	41	44
9	**41**	**44**	**41**	**44**
10	159	58	207	52
11	**207**	**52**	**207**	**52**
12	**77**	**66**	**77**	**66**
13	227	78	163	76
14	60	78	163	76
15	179	96	163	76
16	187	78	163	76
17	57	84	163	76
18	**163**	**76**	**163**	**76**
19	**105**	**94**	**105**	**94**
20	**199**	**94**	**199**	**94**
21	**136**	**106**	**136**	**106**

Patterns Enhancing Interpretability The introduction of the regularisation to the GFN model was motivated by taking a step towards enhancing the interpretability of the model. By regularising it, we gain control over the sampling process. The previous examples in Section 6.1 and quantitative results in Section 5.3 showed that the proposed method indeed works and makes the GFN's generation process more similar to the greedy heuristic. The ultimate goal would be to obtain a model which is moderately regularised, hence preserving the performance of the canonical GFN, but with a more interpretable sampling process. In this section, I will present a set of examples where the regularised GFN solutions yield higher task objective values than the canonical GFN and reveal a common pattern from the regularisation. Such a scenario occurred for the regularisation parameter setting, which implied less strict regularisation. The examples come from evaluation of the model with parameters $f = 0.1$ and $b = 50$.

The first example is presented in Table 6.5 from the graph number 392. The score of the regularised GFN solution is 19, while the canonical GFN achieves a score of 17. This example illustrates well a pattern of the initial steps of the generation process following closely the greedy heuristic. From the middle section of the trajectory, the model starts to deviate and take different actions. This relates to the observations from Regularisation Ratio metric on aggregated results in Section 5.3.

The next example is presented in Table 6.6 from the graph number 274. The objective score of the regularised GFN is 20, which is 2 higher than the canonical GFN. This trajectory also reveals a pattern of similar first steps, but additionally the model follows the heuristic even tighter at the end of the trajectory.

The last example is presented in Table 6.7 from the graph number 184. The regularised GFN solution score is 21, compared to 19 of the canonical GFN. This trajectory example lies between the previous two and confirms the observations of regularisation impact on the early and late stages of the trajectory.

The observed regularisation patterns contribute to GFNs interpretability by establishing a systematic view on analysing GFNs decision-making processes. The consistent temporal patterns of alignment with greedy heuristics and exploration phases could be formalised as follows:

- Phase 1: Greedy Initialisation - the model strongly follows the greedy heuristic.
- Phase 2: Exploration - the model deviates from the greedy heuristic.
- Phase 3: Heuristic Re-alignment - the model aligns with the greedy heuristic for the last steps.

This structured behaviour enables decomposition of the model's decision sequence into interpretable phases, where Phases 1 and 2 are fully interpretable, due to their deterministic definition. The Exploration phase stays in the black-box category, but is restricted to a certain part of the construction process, and one can expect and trace how the model gets there. The examples provide empirical evidence that regularised models can achieve superior performance while maintaining this interpretable structure, for certain graphs. A possible explanation is that the heuristic algorithm is well suited for these graphs' structure,

Table 6.5.: Regularized GFN (f=0.1, b=50) on graph 392.

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	**143**	**32**	**143**	**32**
2	**113**	**34**	**113**	**34**
3	**4**	**46**	**4**	**46**
4	175	52	173	52
5	**136**	**54**	**136**	**54**
6	**211**	**60**	**211**	**60**
7	234	68	68	64
8	**68**	**64**	**68**	**64**
9	**38**	**70**	**38**	**70**
10	14	98	223	80
11	63	94	223	80
12	109	88	223	80
13	80	92	223	80
14	162	94	223	80
15	258	82	223	80
16	97	102	223	80
17	193	138	52	82
18	46	90	52	82
19	**181**	**134**	**181**	**134**

Table 6.6.: Regularized GFN (f=0.1, b=50) on graph 274.

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	**28**	**30**	**28**	**30**
2	188	46	68	42
3	152	58	68	42
4	**68**	**42**	**68**	**42**
5	**36**	**44**	**36**	**44**
6	234	62	110	44
7	178	58	110	44
8	**110**	**44**	**110**	**44**
9	161	64	47	48
10	56	76	47	48
11	50	48	47	48
12	221	98	207	56
13	16	80	207	56
14	96	76	207	56
15	**244**	**60**	**244**	**60**
16	134	80	202	62
17	**202**	**62**	**202**	**62**
18	**102**	**84**	**102**	**84**
19	**213**	**98**	**213**	**98**
20	**166**	**100**	**166**	**100**

Table 6.7.: Regularized GFN (f=0.1, b=50) on graph 184.

Step	Sampled Node Id	Node Degree	Greedy Node Id	Minimal Node Degree
1	**181**	**32**	**181**	**32**
2	**120**	**42**	**120**	**42**
3	**115**	**44**	**115**	**44**
4	18	50	10	48
5	203	74	25	60
6	228	74	25	60
7	76	64	25	60
8	231	76	25	60
9	29	60	25	60
10	156	70	151	70
11	218	84	172	70
12	32	78	172	70
13	56	74	172	70
14	**172**	**70**	**172**	**70**
15	**6**	**84**	**6**	**84**
16	61	92	162	84
17	169	90	162	84
18	43	108	107	90
19	191	98	107	90
20	**107**	**90**	**107**	**90**
21	**139**	**116**	**139**	**116**

but not all cases from the test set. A certain level of performance degradation might be acceptable for the gain of interpretability. The control of the GFN in the first phase enables a predictable start of the generation process. The three-phase pattern is supported by the Regularisation Ratio metric plots for the whole test set, see Section 5.3. The balance between exploration and heuristic-driven steps can be controlled by the hyperparameters (f and b). The regularisation strength can be fine-tuned for the particular example, to achieve the desired consistency in following the heuristic. The bounded Exploration phase is a meaningful step towards the interpretability of the GFN model. In the future, this part could be further analysed to bridge the gap to full interpretability. Overall, the established interpretability structure follows a common pattern for Explainable AI, where the model is interpretable via heuristic-proxy in Phases 1 and 2, and incorporates canonical GFN variance in the Exploration phase.

6.2. Related Work on Partially Explainable Models

The goal of the master thesis is to enhance the interpretability of the GFN model. The results and empirical analysis indicate that the introduced regularisation mechanism leads to a partially interpretable GFN, in which segments of the generation process exhibit interpretable behaviour. To situate this result in the broader context of research on partially explainable models, I will present several related works. [Fit25] describes a similar architecture called the Symbolic Wrapper in the domain of Symbolic AI. The architecture integrates a symbolic model with a neural network, where the symbolic model is used in the input and output phases, while the neural network performs data-driven learning in the middle phase. The design is motivated by combining the neural network's problem-solving capabilities with the symbolic model's interpretability. The architecture closely resembles the three-phase pattern of the regularised GFN model. Another approach, called the Hybrid Predictive Model, is presented in [WL21]. The framework is based on an objective function that combines the model's predictive performance with transparency, defined as the percentage of data processed by the interpretable model. The goal of the method is to substitute the black-box model for those data points where using it is not justified by the gained performance and the simpler model is sufficient. The evaluated hybrid models combined association rules or linear models with black-box models. The paper similarly mentions the trade-off between performance and interpretability and shows that the hybrid models can be an effective solution. Another approach combining human knowledge and predictive modelling is presented in [MS20]. The framework introduces a model that can decide to defer a decision to an expert. The objective function uses cost-sensitive learning with an additional class for the deferred action. The authors similarly mention the trade-off between deferred decisions and prediction accuracy. The notion of deferred decisions reflects the heuristic-driven phases of the regularised GFN model. A key observation is that the approaches cited above explicitly try to combine interpretable or human-based predictions with an autonomous model's decisions. On the contrary, the regularised GFN exhibited such behaviour by itself, while allowing for trade-off control through its hyperparameters.

6.3. Future Work and Practical Applications

Application of the introduced regularisation method can create a partially interpretable and significantly more controllable GFN model. The benefits come with a trade-off of weaker performance in most cases. The regularisation approach is particularly useful when the task demands control of the sampling process, but can allow some parts of the process not to be fully explainable, due to the exploration phase. This direction could be further explored in practical applications, where certain patterns are highly desired or required, given the current state of the environment. Potentially, the heuristic-driven initialisation could be used to speed up the optimisation process for highly complex tasks, where human experts can sketch plausible initial steps.

Future work could focus on reducing the trade-off and finding ways to obtain inter-

pretable GFN models directly from the canonical model to avoid sampling policy change by heuristic-driven regularisation. Possibly, the black-box GFN policy could be projected onto an interpretable policy space, which would allow us to obtain a simpler model with similar performance. The regularisation method could support it by evaluating the receptivity of the GFN to the projection policy design.

The proposed regularisation mechanism can be used in various scenarios. In particular, when trust in the black-box model is not sufficient, or allowed by regulatory requirements. The first example that illustrates such a case could be an application of the GFN in engineering design. Assuming that the task is to allocate wireless network retransmitters, a plausible heuristic could be to place them in the nodes of the network with a low degree to obtain placement close to the Minimal Dominating Set. The model regularised according to the introduced methodology will be more explainable according to the three-phase pattern from Section 6.1. A dedicated study could verify if such behaviour would increase trust and satisfaction of the engineers. The human experts, who are end users of the model predictions, are often hesitant to trust the black-box models. The model that follows the domain knowledge up to a desired level can be more acceptable.

Another example could lie in the field of drug discovery. The GFN model was already used to generate new molecules. With the regularisation method, the model could be used to follow certain patterns, sequences of actions, or chemical properties set by human experts. The laboratory experiments are often expensive and time-consuming, therefore a controlled model might be preferred to reduce risk of failure for the generated molecules.

7. Conclusions

Summarising the study, the main goals of this work were to enhance the interpretability of the Generative Flow Networks, while proposing a regularisation method and approaches to compare and analyse trajectories of step-wise generative algorithms.

The introduced regularisation method was based on the heuristic algorithms guided augmentation of the GFN model training process. The method aimed to make the GFN's sampling policy more similar to the actions of the heuristics. The conducted set of experiments showed that the regularisation can be implemented successfully with the introduced approach. The regularisation mechanism made the sampling trajectories of the regularised GFN more aligned with the trajectories of heuristic algorithms. The proposed metrics of the regularisation impact, namely Regularisation Ratio and Intersections, provided necessary insights to judge the effectiveness of the regularisation on the selected tasks and graphs. The regularisation mechanism, on a higher level, allowed us to gain a better understanding of the GFN model's actions. The empirical analysis of the regularised GFN model showed a systematic three-phase pattern of the sampling process, which combines interpretable and black-box parts. The method, after further tuning and improvements, could be used to control the GFN's behaviour, with the adjustable trade-off between performance and interpretability, thanks to the hyperparameters of the regularisation method. The regularisation mechanism showed its limitations, especially on the MDS task. The regularised GFN did not follow the heuristic trajectories. This indicated that the method might not be suitable for all types of problems, or algorithms used as a reference trajectory supplier. Further study is required to comprehensively evaluate the method with various problems and heuristics.

The methods and conclusions drawn from the study represent valuable steps towards the enhancement of the Generative Flow Networks interpretability. The regularised model's trajectories could be further analysed to bridge the interpretability gap between heuristic-driven and exploratory actions. Some possible approaches could be using a decision tree or linear ML models to project the GFN sampling policy onto simpler classifiers. The model features could correspond to the problem description relevant for the heuristic used as the baseline. Moreover, the hidden knowledge from GFN could be extracted from the neural networks, if they used, e.g., attention mechanisms.

In conclusion, Generative Flow Networks are a promising approach to combinatorial optimisation problems. The regularisation method proposed in this work is a step towards interpretability and user control over the model. The method could be further developed and evaluated to provide better results and robustness.

Bibliography

[BJK+21] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[BLD+23] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations, 2023.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[CPC19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186, 2019.

[DGE+22] Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR, 2022.

[Fic25] Piotr Fic. Code repository for generative flow networks regularisation study. `https://github.com/okcze/GFlowNet-CombOpt`, 2025. Source code and implementation for the regularisation mechanism in Generative Flow Networks applied to combinatorial optimisation problems.

[Fit25] Ricardo Fitas. Neuro-symbolic ai for advanced signal and image processing: A review of recent trends and future directions. *IEEE Access*, 2025.

[GBY+18] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[GPAM+20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2020.

[GS23] Jeff Guo and Philippe Schwaller. Beam enumeration: Probabilistic explainability for sample efficient self-conditioned molecular design. 2023.

[HAR+16] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 3–19. Springer, 2016.

[HCDR21] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685, 2021.

[HMJ+23] Edward J Hu, Nikolay Malkin, Moksh Jain, Katie E Everett, Alexandros Graikos, and Yoshua Bengio. Gflownet-em for learning compositional latent variable models. In *International Conference on Machine Learning*, pages 13528–13549. PMLR, 2023.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[JBHG+22] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[JRHG+23] Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International conference on machine learning*, pages 14631–14653. PMLR, 2023.

[KKZ+23] Minsu Kim, Joohwan Ko, Dinghuai Zhang, Ling Pan, Taeyoung Yun, Woochang Kim, Jinkyoo Park, and Yoshua Bengio. Learning to scale logits for temperature-conditional gflownets. 2023.

[KW+19] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[LJD+23] Dianbo Liu, Moksh Jain, Bonaventure FP Dossou, Qianli Shen, Salem Lahlou, Anirudh Goyal, Nikolay Malkin, Chris Chinenye Emezue, Dinghuai Zhang, Nadhir Hassen, et al. Gflowout: Dropout with generative flow networks. In

International Conference on Machine Learning, pages 21715–21729. PMLR, 2023.

[MJB+22] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

[MRBK+23] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability, 2023.

[MS20] Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR, 13–18 Jul 2020.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[PMZB23] Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, pages 26878–26890. PMLR, 2023.

[PZC+22] Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative augmented flow networks. 2022.

[RBPB23] Julien Roy, Pierre-Luc Bacon, Christopher Pal, and Emmanuel Bengio. Goal-conditioned gflownets for controllable multi-objective molecular design. 2023.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[SPE23] Tony Shen, Mohit Pandey, and Martin Ester. Tacogfn: Target conditioned gflownet for structure-based drug design. 2023.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[WL21] Tong Wang and Qihang Lin. Hybrid predictive models: When an interpretable model collaborates with a black-box model. *Journal of Machine Learning Research*, 22(137):1–38, 2021.

[WZY+19] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. 2019.

[XBHL07] Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial intelligence*, 171(8-9):514–534, 2007.

[ZCY+23] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 2023.

[ZDM+23] Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023.

A. Appendix

A.1. Training Results for MDS and MC tasks

Table A.1.: Scores statistics on MDS validation set.

f	b	mean	median	std	min	max
0.05	**50**	**-12.87**	**-12.78**	**0.31**	**-16.75**	**-12.63**
0.10	50	-12.92	-12.86	0.32	-17.62	-12.62
0.00	1	-13.10	-13.04	0.22	-14.38	-12.89
0.25	50	-13.15	-13.11	0.46	-17.75	-12.65
0.50	50	-14.81	-14.77	0.30	-16.12	-14.41

Table A.2.: Intersection metric statistics on MDS validation set.

f	b	mean	median	std	min	max
0.50	**50**	**0.05**	**0.05**	**0.06**	**0.00**	**0.40**
0.00	1	0.04	0.00	0.05	0.00	0.36
0.05	50	0.04	0.00	0.05	0.00	0.36
0.10	50	0.04	0.00	0.05	0.00	0.43
0.25	50	0.02	0.00	0.04	0.00	0.36

Table A.3.: Regularization ratio statistics on MDS validation set.

f	b	mean	median	std	min	max
0.50	**50**	**0.05**	**0.05**	**0.04**	**0.00**	**0.23**
0.25	50	0.02	0.02	0.02	0.00	0.16
0.00	1	0.02	0.02	0.02	0.00	0.34
0.05	50	0.02	0.02	0.02	0.00	0.25
0.10	50	0.02	0.02	0.02	0.00	0.28

Table A.4.: Scores statistics on Max Cut validation set.

f	b	mean	median	std	min	max
0.00	**1**	**2407.09**	**2412.43**	**41.40**	**1544.62**	**2440.64**
0.10	50	2407.00	2418.59	38.63	1571.38	2434.77
0.05	50	2401.88	2402.46	38.24	1591.88	2441.73
0.25	50	2353.17	2353.58	20.76	1683.50	2375.93
0.50	50	2351.83	2352.54	16.84	1832.38	2374.59

Table A.5.: Intersection metric statistics on Max Cut validation set.

f	b	mean	median	std	min	max
0.25	**50**	**0.88**	**0.90**	**0.08**	**0.48**	**1.00**
0.50	**50**	**0.88**	**0.90**	**0.08**	**0.50**	**1.00**
0.05	50	0.54	0.55	0.17	0.11	0.95
0.10	50	0.51	0.50	0.16	0.11	0.93
0.00	1	0.47	0.45	0.17	0.10	0.94

Table A.6.: Regularization ratio statistics on Max Cut validation set.

f	b	mean	median	std	min	max
0.25	**50**	**0.47**	**0.47**	**0.08**	**0.02**	**0.72**
0.50	50	0.45	0.45	0.08	0.08	0.72
0.05	50	0.04	0.04	0.02	0.00	0.20
0.10	50	0.04	0.03	0.03	0.00	0.19
0.00	1	0.03	0.03	0.03	0.00	0.17