A Denoising Diffusion Adaptive Search for the α-Domination Problem on Social Graphs

Martin Wustinger(✉), Enrico Iurlano, and Günther R. Raidl

Algorithms and Complexity Group, TU Wien,
Favoritenstraße 9-11/192-01, 1040 Vienna, Austria
[firstName].[lastName]@tuwien.ac.at

Abstract. The α-domination problem is a generalization of the classical dominating set problem and serves as a way to model influence structures in social networks. In this work, we build on previous research and explore the use of denoising diffusion models to generate high-quality solutions for this problem. Our main contribution is the introduction of a novel variation of the Greedy Randomized Adaptive Search Procedure (GRASP), utilizing a denoising diffusion model for the parallel construction of a diverse set of initial solutions, which we refer to as the Denoising Diffusion Adaptive Search Procedure (DDASP). By focusing our efforts on real-world social network graphs, we present a superior alternative to conventional metaheuristic algorithms, as prior work has shown that such algorithms often struggle with the unique structural properties of these graphs. Furthermore, we address the challenge of generating viable training data for our models, as the graphs under consideration are typically too large to be used directly. To this end, we employ two graph generation methods to produce artificial training data derived from a set of original input graphs. Experimental results on a benchmark suite of Facebook graphs demonstrate that, particularly in the context of social networks, DDASP consistently outperforms the leading metaheuristic approach under an equivalent time budget.

Keywords: α-Domination · Denoising Diffusion · GRASP

1 Introduction

The α-domination problem asks for the selection of a minimum cardinality subset of vertices $S \subseteq V$ in an undirected simple graph $G = (V, E)$ such that each vertex $v \in V$ is either included in S or has at least a fraction $0 < \alpha \leq 1$ of its neighbors included [14]. This problem is a precursor to the Positive Influence Dominating Set problem proposed by Wang et al. [35] and the Target Set Selection problem introduced by Kempe et al. [21], both of which gained popularity and ask to meet a certain influence-demand via a static, respectively propagating, behavior of influence. These problems naturally model influence propagation in social networks, enabling applications such as the identification of key individuals for health or marketing campaigns, where influence is expected to reach

This document is a preprint accepted for EvoCOP2026.

non-targeted individuals connected to enough targeted peers. Other applications include identifying influential members of a community for educational or political initiatives [35]. As social networks continue to expand and grow in complexity, the demand for algorithms capable of efficiently handling those large-scale networks is becoming increasingly paramount.

Concerning the α-domination problem, Iurlano et al. [20] proposed a Greedy Randomized Adaptive Search Procedure (GRASP) and a configuration checking-based local search and compared them to an exact integer linear programming formulation solved by Gurobi [17]. Results indicate that while the metaheuristics and in particular the configuration checking approach perform well on randomly generated benchmark graphs, these methods struggle with real social network graphs and are often being outperformed by Gurobi within reasonable time limits. We attribute this phenomenon primarily to the unique structural properties of social networks, which are typically characterized by a few high-degree vertices connected to many low-degree ones. In this paper, we attempt to tackle this issue by employing a discrete denoising diffusion framework designed to learn how these structural characteristics shape the space of good solutions for different types of α-domination problem instances. Specifically, we investigate the feasibility of replacing the greedy construction phase of a traditional GRASP with such a framework, aiming to produce a high-quality set of initial solutions even on the inherently more complex social network graphs.

The use of denoising diffusion in the context of combinatorial optimization is relatively new, with the first major contribution in the area made in 2023 by Sun and Yang [34]. Denoising diffusion approaches are nowadays well known for their outstanding performance in image generation. Their fundamental principle is that original training images are iteratively corrupted by adding Gaussian noise, and a neural network is trained to learn the reverse process [19]. The model effectively learns to remove noise from an image, enabling it to generate new images by iteratively applying the learned denoising step to a random input. Sun and Yang [34] demonstrated that these concepts can be extended to combinatorial optimization, introducing DIFUSCO, a diffusion framework for heuristic problem solving on graphs such as the the maximum independent set problem and the traveling salesman problem. While this approach cannot compete with state-of-the-art solvers for these classical problems, the obtained results are nevertheless remarkable for a rather generic and (almost) end-to-end learning-based method. To implement the iterative denoising process for combinatorial optimization problems on graphs, the problems are formulated as the task of finding a $\{0, 1\}$-valued vector, representing the inclusion or exclusion of vertices (edges) in a candidate solution. With a graph neural network (GNN) at its core, DIFUSCO is trained in a supervised manner on a large number of random instances together with their corresponding (close to) optimal solutions and produces heatmaps, i.e., vectors of confidence scores representing the likelihood of inclusion for each vertex (edge) in a (close to) optimal solution. When combined with appropriate post-processing, this approach enables the efficient generation of a *diverse* set of high-quality candidate solutions in parallel. A main

advantage of DIFUSCO over simpler GNN based methods is the fact that DIFUSCO can effectively capture interdependencies among vertices (edges) and multiple modes (i.e., different close to optimal solutions) in the solution space.

Before training such models, a critical factor to consider is the acquisition of appropriate training data. In the original work by Sun and Yang [34], the training data was sampled from the same graph distributions as the graphs used for testing. While this approach was feasible in their setting, it becomes non-trivial when working with large-scale social network graphs, as the model requires high-quality solutions of such graphs to learn effectively. Unfortunately, generating such solutions is practically impossible, as exact solvers fail due to the sheer size of the considered instances. Moreover, training on heuristic solutions risks limiting the model's performance to the quality of those heuristics, potentially undermining the benefits of using a learning-based approach in the first place.

To address this challenge, we propose two graph generation methods, capable of generating meaningful training data based on a given set of real-world social network graphs, which are not only representative for those original graphs, but are also of a much smaller size making them solvable to near optimality with conventional exact solvers. The first method employs a random walk algorithm [18] to select subsets of vertices from the original graphs and constructs the corresponding induced subgraphs. The second method samples sets of expected vertex degrees from an estimated degree distribution of the original graphs and then employs the Chung-Lu model [2] to generate synthetic training instances.

Based on these training instances, we are able to train highly capable models, which, in combination with greedy decoding and a subsequent local search procedure, outperform the heuristic baselines from Iurlano et al. [20] on large social network graph instances. Our contributions can be summarized as follows:

- Comparison of two graph generation methods for creating synthetic training data that is not only representative of the original instances, but is also suitable for training denoising diffusion models.
- Adaptation and evaluation of the DIFUSCO framework for the α-domination problem.
- Proposal of the Denoising Diffusion Adaptive Search Procedure (DDASP), which combines GPU-based parallel generation of candidate solutions with a subsequent local search to refine the results.

The remainder of this paper is organized as follows. Sect. 2 reviews related work. Sect. 3 describes our solution approach in detail, which is empirically evaluated in Sect. 4. Finally, Sect. 5 concludes the paper and outlines directions for future research.

2 Related Work

The α-domination problem was first proposed by Dunbar et al. [14]. Formally, given a proportion $0 < \alpha \leq 1$, the goal is to find a minimum cardinality subset of vertices $S \subseteq V$ of an undirected graph $G = (V, E)$ such that, for each vertex

$v \in V$ one of the following two scenarios applies: (i) $v \in S$ is satisfied; or (ii) $v \in V \setminus S$ with $|S \cap N(v)| \geq \lceil \alpha \cdot |N(v)| \rceil$, where $N(v)$ denotes the set of neighbors of vertex v.

The constant α therefore significantly influences the nature of the problem. Values of α sufficiently close to zero turn the problem into the classical Dominating Set Problem, as we require for each closed neighborhood $(\{v\} \cup N(v))$ at least one vertex to be included. The choice $\alpha = 1$ requires the inclusion of every neighbor of a vertex, thus covering all the edges and fundamentally turning the problem into the Vertex Cover Problem. For the purposes of this paper we set $\alpha = 0.5$; this way the α-domination problem has a close analogy to the Target Set Selection Problem [21] with majority thresholds and also to the even more strongly constrained problem of Positive Influence Domination proposed by Wang et al. [35] in 2009. The main difference is that in the latter problem the aforementioned condition (ii) must be satisfied by all vertices independent of their inclusion in S. A more flexible generalization of α-domination is described by Cicalese et al. [8] in the form of Vector Domination, where for each vertex an individual number of dominators, instead of the universal α-sized share, has to be guaranteed.

The complexity of the α-domination problem has been extensively studied from theoretical perspectives in the past [11,12,16,24,25]. Most contributions that study algorithms for solving α-domination related problems have been made for Positive Influence Domination [1,26,33], like the one by Akbay et al. [3] who applied the Construct-Merge-Solve-Adapt (CMSA) metaheuristic [5]. To address the lack of literature explicitly focusing on the α-domination problem with heuristic approaches, Iurlano et al. [20] proposed two metaheuristics: an approach based on Configuration Checking [7](CC) and a GRASP-based one. The central component for both metaheuristics is their Decrease-Label-and-Compensate algorithm [20], which serves as a destroy-and-repair mechanism within the local search framework. In this work, we incorporate their CC algorithm in the final step to refine the solutions generated by the diffusion models. Additionally, their greedy construction heuristic is adapted to decode the produced heatmaps.

The paper by Sun and Yang [34] serves as the main inspiration for this work and provides the DIFUSCO framework, i.e., the specific denoising diffusion approach that we adapted in our proposed algorithm. While they do not address the α-domination problem, they examine the Maximum Independent Set Problem (MISP), which is structurally related to the α-domination problem as both involve making set-inclusion decisions for each vertex in the graph. Building on DIFUSCO and related denoising diffusion approaches for discrete optimization problems, several contributions have emerged [22,23,32,37], although these focus primarily on the Traveling Salesperson Problem and the MISP. Another notable contribution combining metaheuristics with machine learning comes from Sánchez et al. [31] for the Target Set Selection problem, who proposed a hybrid method that enhances ant colony optimization through Q-learning [36].

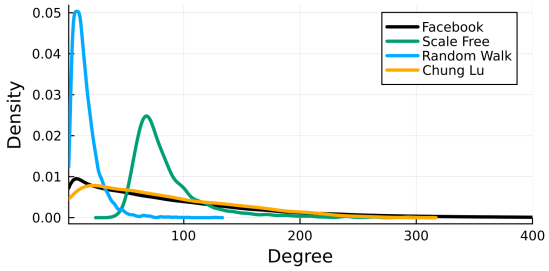


Fig. 1: Comparison of degree distributions between the original benchmark graphs (*Facebook*, up to 15126 nodes and $1.65 \cdot 10^6$ edges) and various graph generation methods. All generated graphs contain 400 vertices. The scale-free graphs were generated with an edge count proportional to that of the *Facebook* graphs. Their power-law exponent was fixed at 3.715, consistent with the estimate obtained using the procedure described by Clauset et al. [9]. The other methods do not require additional parameters.

3 Proposed Approach

In this work, we apply DIFUSCO, the denoising diffusion framework proposed by Sun and Yang [34], to the α-domination problem. Building on the algorithms developed by Iurlano et al. [20], we propose a method that uses a diffusion model at its core to generate high-quality initial solutions, which are subsequently refined by a configuration checking procedure making it structurally comparable to a GRASP.

3.1 Generation of Training Data

For any supervised machine learning model to produce meaningful predictions, the first requirement is access to suitable training data. As mentioned in the introduction, this becomes particularly challenging when working with large-scale real-world graphs, for which exact solutions are not practically attainable. Training with substantially smaller graphs is the obvious alternative, but raises the question of how to generate such graphs in a way that they remain representative of the original large-scale instances and how well the out-of-distribution generalization to larger graphs will perform. Most importantly, training graphs need to approximately share the degree distribution and key structural characteristics of their larger counterparts. Small enough training graphs then allow the use of the MILP model proposed by Raghavan and Zhang [27] in combination with the solver Gurobi [17] to solve the generated instances to (near-)optimality.

The real-world social network graphs considered in this work are the Facebook-graphs taken from the Network Repository [29] and exhibit the expected characteristic structure consisting of many low-degree vertices connected through a few high-degree vertices. At first glance, this structure is reminiscent of scale-free

graphs that follow an underlying power-law degree distribution [28]. To investigate this, we applied the methodology of Clauset et al. [9] and estimated the average power-law exponent of our training dataset to be approximately $\alpha = 3.715$. However, when we used this exponent with the `static_scale_free` generator from Julia's Graphs package [15], the resulting instances showed a degree distribution quite different to the one of, e.g., the Facebook benchmark graphs as illustrated in Figure 1. Even after adjusting the power-law exponent and the edge count, the distribution of vertex degrees remained inadequate, with many degrees being substantially under- or overrepresented in the generated scale-free graphs.

To address this limitation, we experimented with two alternative graph generation methods, described in the following. All graphs used in this study, including real-world and generated instances, are publicly available on GitHub[1].

Random Walk Subgraphs. The first method relies on a random walk procedure [18]. Starting from a randomly chosen vertex in one of the original benchmark graphs, the algorithm performs a random walk by repeatedly jumping to a randomly selected neighbor of the current vertex. This process continues until the desired number of vertices has been visited. The resulting artificial training graph is defined as the induced subgraph over these selected vertices. While this approach yields subgraphs with overall densities similar to the originals, it fails to preserve the original degree distribution accurately, since each vertex inherently lacks its connections to vertices outside the subgraph. This deviation is particularly evident in the prediction dynamics of the GNN forming the core of our denoising diffusion models, where reduced neighborhood sizes during training cause a systematic bias toward over-classification in testing. Also see Fig. 1.

Chung-Lu Random Graphs. To address the deviation from the original degree distributions observed in the previous method, our second algorithm prioritizes accurately reproducing the degree distribution at the cost of no longer preserving overall graph density. We employed standard Locally Estimated Scatterplot Smoothing (LOESS) [10], a generalization of moving averages and polynomial regression, to estimate the degree distribution across the set of original training graphs. Each graph was weighted to ensure that all instances contribute equally, regardless of their size. Using this smoothed model, we estimated a set of expected degrees for the vertices in the training data. The random graph generation method by Chung and Lu [2] was then used to generate synthetic training graphs by sampling edges based on these expected degrees, effectively mimicking the degree distribution learned via the LOESS model. As can be seen in Fig. 1, this method produces graphs with degree distributions closely matching those of the original social network (Facebook) graphs.

[1] https://github.com/mwustinger/ddasp-graph-data

3.2 Denoising Diffusion Model

Sun and Yang [34] describe diffusion models as latent variable models. The forward process progressively corrupts the data by adding noise, while the reverse (denoising) process is learned through supervised training and gradually reconstructs the original data. At test time, predictions are obtained by repeatedly applying this denoising process to a randomly initialized input, ultimately yielding probability values for the nodes indicating their membership in the solution set. The neural network at the core of our models is a GNN consisting of a stack of anisotropic graph neural network layers with edge gating, just as used in [34] for the MISP. Vertex and edge embeddings are obtained at each layer through sparse neighborhood aggregation using either sum, mean, or max pooling. Each layer incorporates timestep embeddings to capture temporal dynamics, while residual connections and normalization ensure stable training.

Since the solution space has a discrete nature, it is essential to employ diffusion methods capable of handling discrete structures. Among the options available in the DIFUSCO framework, categorical diffusion performed best in our experiments. First introduced by Austin et al. [4], categorical diffusion operates by iteratively applying transition matrices that randomly modify the categorical values of the input with a certain probability. The reverse process estimates the original data distribution by computing reverse transition probabilities using Bayes' theorem. For details on the diffusion model, including the parametrization, we refer to the original paper [34] and their experiments on the MISP.

3.3 Heatmap Generation and Decoding

Due to the probabilistic nature of DIFUSCO, it cannot guarantee that its predictions on a graph correspond to valid solutions of the α-domination problem or that a feasible solution is also minimal (i.e., no single vertex can be removed without violating the domination condition). This makes post-processing with a fast greedy decoding strategy important. By default, the models would produce discrete outputs, which discard valuable comparative information about prediction confidence. To address this, the final Bernoulli sampling step of quantization is omitted during inference. Consequently, the model outputs heatmaps that assign each vertex a confidence score in the range $[0, 1]$, rather than a (hard) binary label.

The greedy decoding procedure builds upon the greedy criterion proposed by Iurlano et al. [20]. However, the initial decisions made by the decoding are guided by the model's heatmap predictions, where vertices are provided to the heuristic in descending order of confidence. To mitigate the effect of random noise in the heatmap and to preserve some aspects of the greedy criterion, the confidence scores are rounded half up to one decimal point. Ties are then broken greedily based on the number of undominated neighbors among the top-ranked vertices. A final post-processing step removes any vertices from the solution set that are themselves dominated by the set and are not necessary to maintain the domination of their neighbors. The removal order in this step is randomized. The complete procedure is summarized in Algorithm 1.

Algorithm 1: DIFUSCO Inference and Greedy Decoding

Input: Graph $G = (V, E)$, trained DIFUSCO model $\mathcal{M}$, diffusion steps T
Output: α-dominating set S of G
`// Heatmap Generation`
Initialize latent state for all vertices normally distributed: $X \sim \mathcal{N}(0, 1)$;
for $t \leftarrow T$ **to** *1* **do**
  $X \leftarrow$ perform denoising step t on X using $\mathcal{M}$;
Round latent state to confidence/heat labels: $score \leftarrow \text{round}(X, digits = 1)$;
`// Greedy Decoding`
Initialize α-dominating set: $S \leftarrow \{\}$;
Initialize set of α-dominated vertices: $D \leftarrow \{\}$;
while $D \neq V$ **do**
  Select all vertices with highest score: $Y \leftarrow \arg\max_{v \in V \setminus S}\{score[v]\}$;
  Filter by most undominated neighbors: $Y \leftarrow \arg\max_{v \in Y}\{|N(v) \setminus D|\}$;
  Randomly pick $v' \in Y$; $S \leftarrow S \cup \{v'\}$; $score[v'] \leftarrow -\infty$;
  Update domination: $D \leftarrow D \cup \{v'\} \cup \{v \in N(v') \mid |N(v) \cap S| \geq \lceil \alpha \cdot |N(v)| \rceil\}$
`// Postprocessing Step`
for $v \in S$ *in random order* **do**
  if $S \setminus \{v\}$ *is a valid* α*-dominating set* **then**
    $S \leftarrow S \setminus \{v\}$
return S

3.4 Denoising Diffusion Adaptive Search Procedure (DDASP)

As noted in the introduction, our overall algorithm is inspired by the GRASP framework, where a randomized greedy heuristic generates initial solutions that are subsequently refined through local search. However, since the heuristics in [20] were recognized to exhibit weaknesses on social network graphs, we instead rely here on the above DIFUSCO-based solution construction.

The DIFUSCO framework [34] employs denoising diffusion to generate predictions, and its output is therefore fundamentally influenced by the randomized initialization. Much like an image generator that produces different images from the same prompt, this stochasticity allows DIFUSCO to produce a diverse set of promising candidate solutions rather than a single deterministic prediction. Consequently, DIFUSCO can serve as a natural replacement for the initial construction phase of a GRASP-like algorithm. In addition, DIFUSCO runs almost entirely on the GPU while other heuristic methods, such as the configuration checking procedure proposed by Iurlano et al. [20], are CPU-based. This may enable effective distribution of the workload across both devices.

The complete algorithm alternates between constructing solutions with DIFUSCO and refining them via local search, while maintaining the best solution found so far. In the construction phase, a trained diffusion model generates predictions for the current graph, which are then processed by the greedy decoding procedure described in Algorithm 1. In practice, multiple candidate solutions can be generated in parallel by applying the denoising step to a supergraph con-

Algorithm 2: Denoising Diffusion based Adaptive Search Procedure

Input: Graph $G = (V, E)$, trained DIFUSCO model $\mathcal{M}$
Output: α-dominating set S of G
$S_{best} \leftarrow V$;
while *termination condition not met* **do**
  Apply Algorithm 1 to generate a set of candidate solutions $\mathcal{S}$.
  Select a best performing candidate solution: $S \leftarrow \arg\min_{S \in \mathcal{S}}\{|S|\}$
  S $\leftarrow$ refine with CC(S);
  if $|S| < |S_{\text{best}}|$ **then**
    $S_{\text{best}} \leftarrow$ S;
return S_{best};

taining several copies of the original graph, allowing multiple outputs in a single pass. Among these candidates, the solution with the smallest objective value, corresponding to the α-dominating set with the smallest cardinality, is selected for refinement. This solution is then improved using the configuration checking procedure proposed by Iurlano et al. [20]. The process continues iteratively as DIFUSCO produces new solutions. For details on the configuration checking procedure, including its parameterization, we refer to the original paper [20].

4 Experiments

The DIFUSCO implementation by Sun and Yang [34] is publicly available and was originally written in Python 3.8. In contrast, the α-domination heuristics of Iurlano et al. [20] are implemented in Julia 1.10. To integrate both approaches, we ported the code to Python 3.12 and Julia 1.11.5, respectively, and developed a lightweight framework that calls DIFUSCO from Julia via the PythonCall package [30].

In the following, we evaluate our proposed approach on a selection of social network graphs from the Network Repository [29], a collection of real-world networks and benchmark datasets. We selected a total of 20 graphs from the category *Facebook Networks* and randomly split them into an equally sized training and test set, with the additional constraint that the graphs analyzed by Iurlano et al. [20] were included in our test set to ensure comparability. The graphs in the training set are very large, with an average of 10,823 vertices and 477,190 edges, making it practically infeasible to compute optimal solutions using classical methods such as an ILP approach. Therefore, we employed the two graph generation methods described in Sect. 3 to derive a large number of representative artificial training graphs from the small set of original training graphs. For each method, we generated 1,000 graphs, each containing 400 vertices and on average 2,910 edges for the Random Walk subgraphs and 15,865 edges for the Chung-Lu Random graphs. These generated graphs are large enough to capture the structural characteristics of the original graphs while remaining small enough to be solvable by exact methods. Each generated graph was then solved using the

basic MILP model proposed by Raghavan and Zhang [27] in combination with Gurobi 12.0.1 [17], imposing a time limit of five minutes and using 48 threads on an AMD EPYC 9274F 24-core processor. The resulting average optimality gaps returned by Gurobi were 7.2% for the Random Walk subgraphs and 3.7% for the Chung-Lu random graphs.

4.1 Experimental Setting

To evaluate the performance of our proposed approach, we computed several baselines for comparison. However, since the DIFUSCO framework is primarily implemented in Python and executed on the GPU, whereas the other heuristics are implemented in Julia and executed on the CPU, direct performance comparisons are inherently challenging. We limited the maximum number of threads on the CPU and possibly the GPU to one each and measured the wall-clock times taken by all our algorithms. Thus, we allowed our DIFUSCO-based approach to use both, the CPU and the GPU simultaneously, but just with one thread per device. All experiments, including the neural network training, were done on a machine with an AMD EPYC 9274F 24-core processor and an NVIDIA A40 GPU.

Our main baseline was computed using the basic MILP model proposed by Raghavan and Zhang [27] solved with Gurobi [17] (*gurobi*). Although also restricted to a single thread, the maximum time budget for this computation was set to 3,600 seconds for all experiments. To cover existing metaheuristics, we included the greedy algorithm (*greedy*) and the configuration checking procedure (CC) from Iurlano et al. [20]. The greedy algorithm had no specific time budget and completed even on the largest instances within seconds, while CC was given a budget of $|V|/20$ seconds for each graph, thus, up to 750 seconds. Note that in Table 1, the results listed for *greedy* serve as the initialization for the corresponding CC runs. As shown by Iurlano et al. [20], the relaxed neighborhood notation in their local search framework enabled the exploration of many promising candidate solutions for the α-domination problem, whereas repeatedly restarting from randomized greedy constructions, as done in GRASP, was found to be counterproductive. Consequently, we did not further pursue a GRASP baseline and instead focused on comparing to their strongest method.

In total we trained one model for each combination of GNN aggregation method (*sum, mean, max*) and training graph type, resulting in six models for the subsequent comparison. We hereafter distinguish between Random Walk models and Chung-Lu models referring to the set of synthetic graphs used during training. Each graph neural network model consists of eight hidden layers with 256 nodes per layer, employs categorical diffusion. Increasing the number of layers or their size did not yield significant improvements in performance. Training was performed for 50 epochs with a batch size of eight and 1000 diffusion steps. During inference, the number of denoising steps was reduced to 100, which, combined with rounding, yielded competitive results at a fraction of the computational cost than when performing 1000 denoising steps.

4.2 Main Results

As previously mentioned, we trained a total of six diffusion models. As will be detailed in Sect. 4.3, the models using *sum* aggregation consistently outperformed the others, so only their results are reported in Figure 2 and Table 1, while a comparison between the different aggregation methods is presented in Figure 3. To highlight the contribution of the refinement phase, we present results for both the isolated construction procedure of DIFUSCO (*difusco*) and the complete DDASP (*ddasp*). The results presented under *difusco* correspond to repeatedly applying DIFUSCO trained on our artificial training graphs followed by greedy decoding to obtain candidate solutions, whereas the *ddasp* results include the full algorithm, i.e., construction through DIFUSCO and subsequent refinement via local search. The time budget for these experiments was again set to $|V|/20$ seconds, matching the metaheuristic baseline.

In Figures 2 and 3 as well as Table 1, *rw* denotes the Random Walk training set, *cl* the training set generated via the Chung-Lu method, and *sum*, *mean*, *max* denote the aggregation method used in the GNN. For each graph in the test set, we conducted a total of 30 experiments under these settings but with different random seeds. Let S_{G} denote the solution returned by our Gurobi benchmark, S the solution returned by one of our algorithms. The %-gap, which quantifies the difference in solution quality, is given by $(|S|-|S_{\mathrm{G}}|)/|S_{\mathrm{G}}|$ and is stated in percent. Figure 2 shows the probability density functions of the %-gap achieved by the different approaches across the 30 experiments for each graph, providing a fair comparison between the heuristic baselines and the results computed using the DIFUSCO-based methods. Moreover, Table 1 reports the means and standard deviations of the objective values.

Although the MILP approach initially produced competitive results, its effectiveness declined rapidly as the graph size increased despite having a larger time budget, eventually being outperformed by all other methods. The greedy algorithm (*greedy*) served as the input to the configuration checking procedure (*CC*) and therefore provides a baseline for assessing the improvements achieved by that procedure. However, because the greedy algorithm was executed only once for this purpose, it terminated significantly faster than the other heuristics, making a direct comparison problematic. In the context of social network graphs, this improvement of the configuration checking procedure was unfortunately smaller than expected, which is consistent with the observations of Iurlano et al. [20].

To assess the statistical significance of the results, we performed Wilcoxon Signed-Rank Tests [13] on the median %-gaps for each test instance. Comparing the configuration checking procedure to the DDASP trained on the Random Walk dataset using *sum* aggregation, as well as the DDASP trained on the Chung-Lu dataset with *sum* aggregation, yielded a p-value of 0.0020. A direct comparison between the two DDASP variants resulted in a p-value of 0.0039.

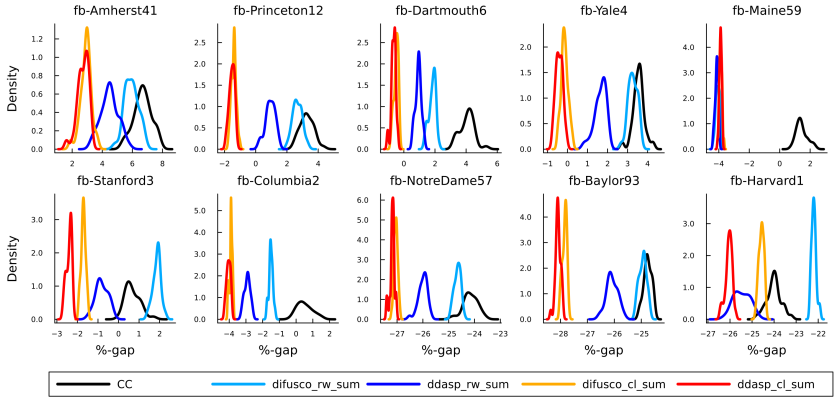


Fig. 2: Comparison between the configuration checking baseline and our algorithms involving the DIFUSCO framework.

Table 1: Objective values of the best-performing methods. For each method the mean ($\mu_{|S|}$) and standard deviation ($\sigma_{|S|}$) of the objective values of the individual experiments is given. Best mean values are printed in bold.

Instance	Graphs $\lvert V\rvert$	μ_{deg}	Baselines gurobi	greedy $\mu_{\lvert S\rvert}$	greedy $\sigma_{\lvert S\rvert}$	CC $\mu_{\lvert S\rvert}$	CC $\sigma_{\lvert S\rvert}$	Random Walk (sum) difusco $\mu_{\lvert S\rvert}$	difusco $\sigma_{\lvert S\rvert}$	ddasp $\mu_{\lvert S\rvert}$	ddasp $\sigma_{\lvert S\rvert}$	Chung-Lu (sum) difusco $\mu_{\lvert S\rvert}$	difusco $\sigma_{\lvert S\rvert}$	ddasp $\mu_{\lvert S\rvert}$	ddasp $\sigma_{\lvert S\rvert}$
fb-Amherst41	2235	81.4	**738**	806.3	6.8	787.3	4.3	781.8	3.3	771.1	4.3	759.5	2.6	758.1	2.8
fb-Princeton12	6596	88.9	2208	2313.0	7.0	2280.3	9.5	2266.9	6.6	2228.7	6.9	2176.9	3.5	**2174.3**	3.8
fb-Dartmouth6	7694	79.0	2465	2588.2	6.0	2565.1	11.3	2510.0	5.5	2488.0	4.6	2452.8	3.5	**2448.7**	3.6
fb-Yale4	8578	94.5	2928	3059.5	10.7	3032.0	8.4	3023.6	6.7	2976.0	8.7	2923.8	5.1	**2915.8**	5.0
fb-Maine59	9069	53.6	3167	3241.1	9.1	3212.6	11.7	3038.2	4.1	3035.9	3.2	3044.3	2.8	**3043.9**	2.4
fb-Stanford3	11586	98.1	3902	3965.4	12.6	3928.5	14.1	3974.8	7.5	3870.2	11.7	3835.4	4.3	**3807.9**	5.3
fb-Columbia2	11770	75.5	4005	4075.8	10.1	4025.5	18.3	3944.6	4.7	3887.9	6.9	3848.9	3.8	**3843.1**	4.7
fb-NotreDame57	12155	89.1	5804	4469.8	10.5	4402.0	16.4	4372.8	7.3	4294.9	10.1	4234.2	4.1	**4226.2**	5.3
fb-Baylor93	12803	106.2	6132	4649.8	12.1	4613.6	9.0	4602.3	8.6	4532.6	12.9	4424.4	5.2	**4409.5**	5.8
fb-Harvard1	15126	109.0	6826	5263.2	16.2	5187.0	22.8	5310.8	7.2	5087.4	25.4	5148.4	7.7	**5047.6**	10.7

4.3 Comparison of Different Aggregation Methods

One of the most important parameters influencing the performance of our denoising diffusion models was the choice of the aggregation method. The diffusion models studied by Sun and Yang [34] primarily employ *sum* to aggregate neighborhood information at each layer of the denoising step. The DIFUSCO framework, however, also supports other well-known aggregation methods, such as *mean* aggregation, which computes the arithmetic mean of the neighborhood information, and *max* aggregation, which takes the largest value among the neighbors. Our experiments confirmed that *sum* is the most effective aggregation method in the context of the α-domination problem and likely explains its extensive use in the original study. Unlike *mean* and *max* aggregation, which largely ignore vertex degree, *sum* reflects this important aspect more directly, as each neighbor directly contributes to the aggregate.

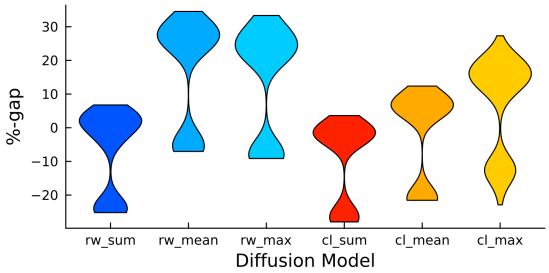


Fig. 3: Comparison of the construction procedure using different DIFUSCO models trained on different training graph types (Random Walk vs. Chung-Lu) and using different GNN aggregation methods (*sum, mean, max*).

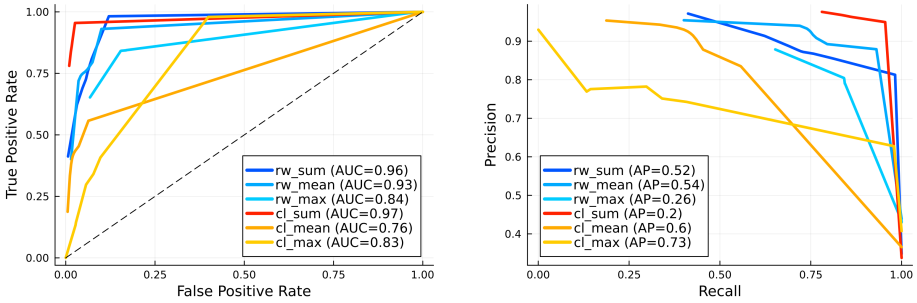
However, this prominent effectiveness comes with an important caveat. The *sum* aggregation performs well only when the degree distributions of the training and test graphs are similar. In early experiments on artificial scale-free graphs, models using *sum* aggregation quickly began to over-classify. In extreme cases, nearly all vertices were labeled as part of the α-dominating set, rendering the predictions meaningless. This observation motivated us to carefully design artificial training data that especially tries to replicate the degree distribution of the original training graphs. We argue that this is also the main reason why the Chung-Lu dataset outperforms the Random Walk dataset, as its degree distribution is more in line with the original training data.

Figure 3 shows the %-gap of the isolated construction procedure using DIFUSCO relative to the solutions computed by Gurobi. While the Random Walk training set performed worse overall than the Chung-Lu training set, the results indicate that the choice of aggregation method has far greater impact. For both training datasets, only the models using *sum* aggregation are competitive with the heuristic baselines.

4.4 From Heatmaps to Greedy-Decoded Solutions

Lastly, we examine the performance of our diffusion models to predict valid solutions, by comparing them to the solutions obtained after greedy decoding. We treat the rounded output of the denoising diffusion models as the prediction and the corresponding greedy-decoded solution as the reference. It is important to note that this does not directly assess the absolute performance of the models, which was investigated in the previous sections, as combinatorial optimization problems typically have many close to optimal solutions, and thus no unique ground truth exists. Rather, the following helps us quantify the deviation of the model's confidence scores from the final greedy solution i.e., the extent of corrective adjustments the greedy decoding performs.

Figure 4 shows on the left side Receiver Operating Characteristic (ROC) curves, which relate the False Positive Rate to the True Positive Rate and thereby



(a) ROC Curves (b) Precision–Recall Curves

Fig. 4: Comparison of ROC and Precision–Recall curves for the evaluated models.

indicate how effectively the models rank vertices that ultimately appear in the final decoded solution. On the right, Precision-Recall (PR) curves are presented illustrating the models' abilities to identify vertices that are ultimately used by the greedy decoding procedure. Although there is no direct correspondence between the overall model performance and the associated ROC and PR curves, models employing *sum* aggregation consistently exhibit clearly superior curves and larger areas under the curve (AUC). This indicates that in order to achieve better performance, it helps when the denoising diffusion models already return confidence scores that closely align with meaningful solutions and less work has to be done by the greedy decoder.

5 Conclusions

We proposed DDASP, the Denoising Diffusion Adaptive Search Procedure, to solve the α-domination problem on large-scale social network graphs. To effectively train the employed denoising diffusion models, we introduced two graph generation methods for producing suitable training data that approximate the degree distribution of the original graphs while remaining solvable by a MILP model in combination with Gurobi. By incorporating the configuration checking-based local search from [20] into our adaptation of the DIFUSCO framework, we were able to surpass both the heuristic and Gurobi baselines, achieving statistically significant improvements on a benchmark set of large, real-world social network graphs.

For future work, we plan to broaden our evaluation of denoising diffusion models, potentially implementing them natively in Julia to simplify the current Python/Julia interaction or explore alternative implementations. Preliminary trials with a GATv2 [6] graph neural network in place of the anisotropic GNN produced mixed results, motivating further investigation. Finally, we aim to extend DDASP to other combinatorial optimization problems, with the expectation that machine learning can once again capture and exploit their structural properties to enhance classical heuristic methods.

References

1. Abu-Khzam, F.N., Lamaa, K.: Efficient heuristic algorithms for positive-influence dominating set in social networks. In: INFOCOM 2018 - IEEE Conference on Computer Communications Workshops. pp. 610–615. IEEE (2018). https://doi.org/10.1109/INFCOMW.2018.8406851
2. Aiello, W., Chung, F.R.K., Lu, L.: A random graph model for massive graphs. In: Yao, F.F., Luks, E.M. (eds.) Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing. pp. 171–180. ACM (2000). https://doi.org/10.1145/335305.335326
3. Akbay, M.A., Serrano, A.L., Blum, C.: A self-adaptive variant of CMSA: application to the minimum positive influence dominating set problem. International Journal of Computational Intelligence Systems **15**(1), 44 (2022). https://doi.org/10.1007/S44196-022-00098-1
4. Austin, J., Johnson, D.D., Ho, J., Tarlow, D., van den Berg, R.: Structured denoising diffusion models in discrete state-spaces. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21 (2021). https://doi.org/10.48550/arXiv.2107.03006
5. Blum, C., Davidson, P.P., López-Ibáñez, M., Lozano, J.A.: Construct, merge, solve & adapt A new general algorithm for combinatorial optimization. Computers & Operations Research **68**, 75–88 (2016). https://doi.org/10.1016/j.cor.2015.10.014
6. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: The Tenth International Conference on Learning Representations (ICLR 2022). OpenReview.net (2022). https://doi.org/10.48550/arXiv.2105.14491
7. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence **175**(9), 1672–1696 (2011). https://doi.org/https://doi.org/10.1016/j.artint.2011.03.003
8. Cicalese, F., Milanič, M., Vaccaro, U.: On the approximability and exact algorithms for vector domination and related problems in graphs. Discrete Applied Mathematics **161**(6), 750–767 (2013). https://doi.org/10.1016/j.dam.2012.10.007
9. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-law distributions in empirical data. SIAM Review **51**(4), 661–703 (2009). https://doi.org/10.1137/070710111
10. Cleveland, W.S.: Robust locally weighted regression and smoothing scatterplots. Journal of the American Statistical Association **74**(368), 829–836 (1979). https://doi.org/10.1080/01621459.1979.10481038
11. Dahme, F., Rautenbach, D., Volkmann, L.: Some remarks on alpha-domination. Discussiones Mathematicae Graph Theory **24**(3), 423–430 (2004). https://doi.org/10.7151/DMGT.1241
12. Das, A., Laskar, R.C., Rad, N.J.: On α-domination in graphs. Graphs and Combinatorics **34**(1), 193–205 (2018). https://doi.org/10.1007/s00373-017-1869-1
13. DeniseRey, Neuhäuser, M.: Wilcoxon-Signed-Rank Test, pp. 1658–1659. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-04898-2_616
14. Dunbar, J.E., Hoffman, D.G., Laskar, R.C., Markus, L.R.: α-domination. Discrete Mathematics **211**, 11–26 (2000). https://doi.org/10.1016/S0012-365X(99)00131-4
15. Fairbanks, J., Besançon, M., Simon, S., Hoffiman, J., Eubank, N., Karpinski, S.: JuliaGraphs/Graphs.jl: an optimized graphs package for the Julia programming language (2021), https://github.com/JuliaGraphs/Graphs.jl/
16. Gagarin, A., Poghosyan, A., Zverovich, V.E.: Upper bounds for α-domination parameters. Graphs and Combinatorics **25**(4), 513–520 (2009). https://doi.org/10.1007/S00373-009-0864-6

17. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2024), https://www.gurobi.com
18. Göbel, F., Jagers, A.: Random walks on graphs. Stochastic Processes and their Applications **2**(4), 311–336 (1974). https://doi.org/10.1016/0304-4149(74)90001-5
19. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20 (2020). https://doi.org/10.48550/arXiv.2006.11239
20. Iurlano, E., Varga, J., Raidl, G.R.: Tackling the α-domination problem heuristically. In: Quesada-Arencibia, A., Affenzeller, M., Moreno-Díaz, R. (eds.) Computer Aided Systems Theory - EUROCAST 2024 - 19th International Conference. Lecture Notes in Computer Science, vol. 15172, pp. 148–156. Springer (2024). https://doi.org/10.1007/978-3-031-82949-9_14
21. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: Getoor, L., Senator, T.E., Domingos, P.M., Faloutsos, C. (eds.) Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 137–146. ACM (2003). https://doi.org/10.1145/956750.956769
22. Li, Y., Guo, J., Wang, R., Yan, J.: T2T: From distribution learning in training to gradient search in testing for combinatorial optimization. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS '23 (2023)
23. Pugacheva, D., Ermakov, A., Lyskov, I., Makarov, I., Zotov, Y.: Enhancing GNNs performance on combinatorial optimization by recurrent feature update (2024). https://doi.org/10.48550/arXiv.2407.16468
24. Rad, N.J., Volkmann, L.: Vertex-removal in α-domination. Filomat **26**(6), 1257–1262 (2012), http://www.jstor.org/stable/24895832
25. Rad, N.J., Volkmann, L.: Edge-removal and edge-addition in α-domination. Graphs and Combinatorics **32**(3), 1155–1166 (2016). https://doi.org/10.1007/S00373-015-1614-6
26. Raei, H., Yazdani, N., Asadpour, M.: A new algorithm for positive influence dominating set in social networks. In: International Conference on Advances in Social Networks Analysis and Mining. pp. 253–257. IEEE Computer Society (2012). https://doi.org/10.1109/ASONAM.2012.51
27. Raghavan, S., Zhang, R.: Rapid influence maximization on social networks: The positive influence dominating set problem. INFORMS Journal on Computing **34**(3), 1345–1365 (2022). https://doi.org/10.1287/IJOC.2021.1144
28. Reed, W.J.: The Pareto, Zipf and other power laws. Economics Letters **74**(1), 15–19 (2001). https://doi.org/10.1016/S0165-1765(01)00524-9
29. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015), http://networkrepository.com
30. Rowley, C.: PythonCall.jl: Python and Julia in harmony (2022), https://github.com/JuliaPy/PythonCall.jl
31. Sánchez, J.E.R., Sartori, C.C., Blum, C.: Q-learning ant colony optimization supported by deep learning for target set selection. In: Silva, S., Paquete, L. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2023). pp. 357–366. ACM (2023). https://doi.org/10.1145/3583131.3590396
32. Soler, J.S., Raidl, G.R.: A denoising diffusion-based evolutionary algorithm framework: Application to the maximum independent set problem (2025). https://doi.org/10.48550/arXiv.2510.08627

33. Sun, R., Wu, J., Jin, C., Wang, Y., Zhou, W., Yin, M.: An efficient local search algorithm for minimum positive influence dominating set problem. Computers & Operations Research **154**, 106197 (2023). https://doi.org/10.1016/J.COR.2023.106197
34. Sun, Z., Yang, Y.: Difusco: graph-based diffusion solvers for combinatorial optimization. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS '23 (2023). https://doi.org/10.48550/arXiv.2302.08224
35. Wang, F., Camacho, E., Xu, K.: Positive influence dominating set in online social networks. In: Du, D., Hu, X., Pardalos, P.M. (eds.) Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications. Lecture Notes in Computer Science, vol. 5573, pp. 313–321. Springer (2009). https://doi.org/10.1007/978-3-642-02026-1_29
36. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**, 279–292 (1992). https://doi.org/10.1007/BF00992698
37. Zhao, H., Yu, K., Huang, Y., Yi, R., Zhu, C., Xu, K.: Disco: Efficient diffusion solver for large-scale combinatorial optimization problems. Graphical Models **141**, 101284 (2025). https://doi.org/10.48550/arXiv.2406.19705