Neural Networks 155 (2022) 168–176

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A differentiable approach to the maximum independent set problem using dataless neural networks

Ismail R. Alkhouri [a,*], George K. Atia [a], Alvaro Velasquez [b]

[a] University of Central Florida, 4000 Central Florida Blvd, Orlando, FL 32816, USA
[b] University of Colorado Boulder, 440 UCB, Boulder, CO 80309, USA

ARTICLE INFO

Article history:
Received 19 April 2022
Received in revised form 16 June 2022
Accepted 8 August 2022
Available online 19 August 2022

Keywords:
Combinatorial optimization
Maximum independent set problem
Maximum Clique problem
Dataless Neural Networks
Community detection

ABSTRACT

The success of machine learning solutions for reasoning about discrete structures has brought attention to its adoption within combinatorial optimization algorithms. Such approaches generally rely on supervised learning by leveraging datasets of the combinatorial structures of interest drawn from some distribution of problem instances. Reinforcement learning has also been employed to find such structures. In this paper, we propose a different approach in that no data is required for training the neural networks that produce the solution. In this sense, what we present is not a machine learning solution, but rather one that is dependent on neural networks and where backpropagation is applied to a loss function defined by the structure of the neural network architecture as opposed to a training dataset. In particular, we reduce the popular combinatorial optimization problem of finding a maximum independent set to a neural network and employ a dataless training scheme to refine the parameters of the network such that those parameters yield the structure of interest. Additionally, we propose a universal graph reduction procedure to handle large-scale graphs. The reduction exploits community detection for graph partitioning and is applicable to any graph type and/or density. Experimental results on both real and synthetic graphs demonstrate that our proposed method performs on par or outperforms state-of-the-art learning-based methods in terms of the size of the found set without requiring any training data.

1. Introduction

In his seminal work (Karp, 1972), Richard Karp demonstrated the reducibility among combinatorial problems that are complete for the complexity class **NP**. Combinatorial optimization problems have since been frequently associated with the **NP-hard** complexity class for which no efficient solutions are likely to exist. Despite their apparent intractability, **NP-hard** problems have found ubiquitous application across many sectors (Bengio, Lodi, & Prouvost, 2021). While there is not a known polynomial-time solver with respect to (w.r.t.) the size of the input for any of these problems, there are many approximate and efficient solvers (Lamm, Sanders, Schulz, Strash, & Werneck, 2016). In general, these solvers are broadly categorized into heuristic algorithms (Akiba & Iwata, 2016), conventional branch-and-bound methods (San Segundo, Rodríguez-Losada, & Jiménez, 2011), and approximation algorithms (Boppana & Halldórsson, 1992).

One of the most well-studied **NP-hard** problems is the Maximum Independent Set (MIS) problem which consists of finding a set of maximum cardinality that contains vertices in a graph $G = (V, E)$ such that no two vertices are connected by an edge (Tarjan & Trojanowski, 1977). There exist powerful heuristic solvers for the MIS problem. The state-of-the-art solver is the ReduMIS evolutionary algorithm, which was developed in the pioneering work of Lamm et al. (2016). Despite the notable success of ReduMIS and related heuristic solvers, developing alternative learning-based solutions to combinatorial optimization problems (He, Daume III, & Eisner, 2014; Li, Chen, & Koltun, 2018) is premised on unceasing advancement in our understanding of the theoretical and computational foundations of machine learning. However, these methods typically require extensive training of neural networks (NNs) using large graph datasets for which solutions are known. In this paper, we develop an altogether different NN-based approach to the MIS problem that rests on a dataless NN (dNN) and a novel dataless training paradigm. In our approach, for each graph G, we construct a NN with fixed and trainable parameters that is specific to G and require no training dataset, hence the appellation 'dataless'. The connectivity structure of the dNN is derived from the given graph G and its input is data-independent. The output of the dNN is minimized upon finding a desired MIS which can be constructed from the learned parameters of the NN.

The first contribution of this paper is the introduction of dNNs for which no data is required during training. The second

* Corresponding author.
E-mail address: ialkhouri@knights.ucf.edu (I.R. Alkhouri).

https://doi.org/10.1016/j.neunet.2022.08.008
0893-6080/

contribution is the representation of the MIS problem as a single differentiable function, thereby enabling the adoption of differentiable solutions. Third, we develop a graph reduction procedure based on community detection for large-scale graphs. Unlike most common reduction rules whose applicability is limited to sparse graphs, this procedure is universal in that it is applicable to any graph type. Our fourth contribution is the introduction of an iterative solution improvement procedure based on simulated annealing and dNNs. To evaluate success, we use the state-of-the-art solution size obtained by ReduMIS as a benchmark. We also compare to state-of-the-art learning-based methods, such as (Li et al., 2018), where we show that our approach yields comparable or better solutions while being dataless. As a future research avenue, we will explore extensions to other **NP-hard** problems given known reductions among problems from this complexity class.

2. Related work

Exact algorithms for **NP-hard** problems are typically based on enumeration or branch-and-bound techniques. However, these techniques are not applicable to large problem spaces (Dai, Dai, & Song, 2016). This motivated the development of efficient approximation algorithms and heuristics, such as the procedure implemented in the NetworkX library for solving MIS (Boppana & Halldórsson, 1992). These polynomial-time algorithms and heuristics typically utilize a combination of various subprocedures, including greedy algorithms, local search subroutines, and genetic procedures (Williamson & Shmoys, 2011). Such heuristics cannot generally guarantee that the resulting solution is within some small factor of optimality. In fact, for the complementary problem of finding a Maximum Clique (MC), an algorithm that provably guarantees an approximate solution to MC within a factor of $n^{1-\epsilon}$, where n is the number of nodes in the underlying graph, for any $\epsilon > 0$ is not possible unless **P** = **NP** (Hastad, 1996). Similar inapproximability results have been established for the MIS problem (Berman & Schnitger, 1992). As such, heuristics without approximation guarantees have been adopted for practical purposes for these problems.

The ReduMIS method (Lamm et al., 2016) is the state-of-the-art solver for the MIS problem. It consists of two main components: an iterative implementation of a series of graph reduction techniques, followed by the use of an evolutionary algorithm. The latter starts with a pool of independent sets, then evolves the pool over several rounds. In each round, the algorithm uses a selection procedure to select favorable nodes. This is achieved by executing graph partitioning that clusters the graph nodes into disjoint clusters and separators for solution improvement. Our method, however, does not include solution combination operation. In contrast, we leverage a community detection method prior to obtaining the initial solution(s) for the purpose of scaling up to large graphs and not for solution improvement. Moreover, we do not enforce the partitions and separators to be disjoint (i.e., sharing no edges) (See Section 4.3).

Learning-based approaches which make use of RL algorithms and ML architectures have been recently introduced to solve **NP-hard** problems. RL-based methods train a deep Q-Network (DQN) such that the obtained policy operates as a meta-algorithm that incrementally yields a solution (Bengio et al., 2021). The recent state-of-the-art work of Dai, Khalil, Zhang, Dilkina, and Song (2017) combines a DQN with graph embeddings, allowing discrimination between vertices based on their impact on the solution, and enabling scalability to larger problem instances. By contrast, our proposed method does not require training of a DQN. The authors of Schuetz, Brubaker, and Katzgraber (2021) developed a method to the MIS problem, which, similar to our approach, does not require training data. However, different from our approach, it uses a graph neural network (GNN) (its output represents the probability of each node being in the solution) whose parameters are tuned based on a loss function inspired by quadratic binary unconstrained optimization that encodes the graph of interest. In contrast, we construct a dNN with a simple architecture that only has n trainable parameters, where n is the number of nodes in the graph. This is significantly smaller than the number of tunable parameters of the aforementioned GNN, which has n parameters just in its last layer. In addition, the approach in Schuetz et al. (2021) was only tested using d-regular graphs (Steger & Wormald, 1999) and compared against the approximate solver in Boppana and Halldórsson (1992). The applicability of the former solver is limited to small size graphs and its performance was surpassed by other existing methods.

The supervised learning method in Li et al. (2018) nearly achieves the state-of-the-art performance for the MIS problem. It integrates several graph reductions (Lamm et al., 2016), Graph Convolutional Networks (GCN) (Defferrard, Bresson, & Vandergheynst, 2016), guided tree search, and a solution improvement local search algorithm (Andrade, Resende, & Werneck, 2012). The GCN is trained using benchmark graphs and their solutions as the true labels to learn probability maps for each vertex being in the optimal solution. The point of resemblance to our approach is the use of an NN to derive solutions to combinatorial optimization problems. However, a major difference is that our approach does not rely on supervised learning; it uses an entirely different dNN and obtains a solution via dataless training. More specifically, the means by which we optimize the dNN consists of applying backpropagation (Riedmiller & Braun, 1993) to a loss function defined entirely in terms of the given graph and the structure of the dNN without the need for a dataset as is standard in training deep learning models.

3. Preliminaries

An undirected graph is denoted by $G = (V, E)$, where V is its vertex set and $E \subseteq V \times V$ is its edge set. The number of nodes is $|V| = n$ and the number of edges is $|E| = m$. We also use the notation $V(H)$ and $E(H)$ to refer to the vertex and edge sets of some graph H, respectively. The degree of a node $v \in V$ is denoted by $d(v)$, and the maximum degree of the graph by $\Delta(G)$. The neighborhood of node $v \in V$ is $N(v) = \{u \in V \mid (u, v) \in E\}$. For a subset of nodes $U \subseteq V$, $G[U] = (U, E[U])$ is used to represent the subgraph induced by U, i.e., the graph on U whose edge set $E[U] = \{(u, v) \in E \mid u, v \in U\}$ consists of all edges of G with both ends in U. The complement of graph G is the graph $G' = (V, E')$ on V, where $E' = V \times V \setminus E$, i.e., E' consists of all the edges between nodes that are not adjacent in G, with $|E'| = m'$. Hence, $m+m' = n(n-1)/2$ is the number of edges in the complete graph on V. For any positive integer n, $[n] := \{1, \ldots, n\}$. We use $|\cdot|$ to denote the cardinality of a set, unless stated otherwise.

We consider the **NP-hard** problem of finding maximum independent sets (MIS). We define the MIS problem and the related maximum clique (MC) and minimum vertex cover (MVC) problems, then briefly describe how MC and MVC can be represented as instances of MIS.

Definition 3.1 (*MIS Problem*)**.** Given an undirected graph $G = (V, E)$, MIS is the problem of finding a subset of vertices $\mathcal{I} \subseteq V$ such that $E(G[\mathcal{I}]) = \emptyset$, and $|\mathcal{I}|$ is maximized.

Definition 3.2 (*MC Problem*)**.** Given an undirected graph $G = (V, E)$, MC is the problem of finding a subset of vertices $C \subseteq V$ such that $G[C]$ is a complete graph, and $|C|$ is maximized.

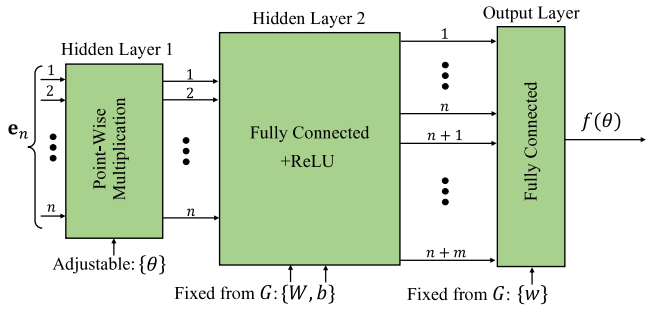

Fig. 1. Block diagram of the proposed dNN, $f(\theta)$.

Definition 3.3 (*MVC Problem*)**.** Given an undirected graph $G = (V, E)$, MVC is the problem of finding a subset of vertices $R \subseteq V$ such that, for every $(u, v) \in E$, either $u \in R$ or $v \in R$, and $|R|$ is minimized.

For the MC problem, the MIS of a graph is an MC of the complement graph (Karp, 1972). MVC and MIS are complementary, i.e, a vertex set is independent if and only if its complement is a vertex cover (Cook, Lovász, Seymour, et al., 1995). We exploit these properties in the development of our dNNs.

4. Methodology

In this section, we describe the different components of our proposed approach. To preface the discussion and distinguish our dataless solution from learning-based methods, consider first the conventional supervised learning setting. In this setting, there is generally some set of data $D = \{(x_i, y_i)\}_i$ consisting of input tensors $x_i \in \mathbb{R}^n$ and their associated value, or label, $y_i \in Y$. The goal of learning is then to train a learning model $f : \mathbb{R}^n \to Y$ parameterized by θ so that f learns to predict the input–output relationship of the underlying data distribution D'. This is given by the objective below, in which $\mathcal{L}$ denotes the loss function.

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D'}[\mathcal{L}(f(x;\theta), y)] \tag{1}$$

Since a dataset D is used in lieu of the true underlying data distribution D', the objective function becomes

$$\min_{\theta} \frac{1}{|D|} \sum_{(x,y)\in D} \mathcal{L}(f(x;\theta), y) \tag{2}$$

The loss function is chosen to be a differentiable function, such as the minimum square error, in order to leverage optimization using backpropagation. In our approach, we leverage dNNs, which we define as neural networks whose loss function $\mathcal{L}$ does not depend on data. In this sense, what we present is a NN-based technique, but not a learning method and is therefore different from supervised, unsupervised, and reinforcement learning.

4.1. Dataless neural network construction

Given a graph $G = (V, E)$, we construct a dNN f with trainable parameters $\theta \in [0, 1]^n$ whose single output is $f(\mathbf{e}_n; \theta) = f(\theta) \in \mathbb{R}$. Note that the input to f is the all-ones vector $\mathbf{e}_n$ and thus does not depend on any data. The network consists of an input layer, two hidden layers, and an output layer. The trainable parameters $\theta \in [0, 1]^n$ connect the input layer $\mathbf{e}_n$ to the first hidden layer through an elementwise product. All other parameters are fixed during training and are presented next. The connectivity structure from the first hidden layer to the second is given by the binary matrix $W \in \{0, 1\}^{n\times(n+m)}$ and will depend on G, the bias vector at the second hidden layer is given by $b \in \{-1, -1/2\}^{n+m}$, and the fully-connected weight matrix from the second hidden layer to the output layer is given by $w \in \{-1, n\}^{n+m}$. These parameters are defined as a function of G. The output of f is given by (3), where $\odot$ is the element-wise Hadamard product that represents the operation of the first hidden layer of the constructed network. The second hidden layer is a fully-connected layer with fixed matrix W and bias vector b with a ReLU activation function $\sigma(x) = \max(0, x)$, while the last layer is another fully-connected layer expressed in vector w. See Fig. 1 for a block diagram of the generalized proposed network.

$$f(\mathbf{e}_n ; \theta) = f(\theta) = w^T \sigma(W^T(\mathbf{e}_n \odot \theta) + b) \tag{3}$$

In the sequel, we prove that $f(\theta)$ is an equivalent differentiable representation of the MIS problem $G = (V, E)$ in that f achieves its minimum value when an MIS $U \subseteq V$ is found in G. Furthermore, this is a constructive representation since U can be obtained from θ as follows. Let $\theta^* = \arg\min_{\theta\in[0,1]^n} f(\theta)$ denote an optimal solution to f and let $\mathcal{I} : [0, 1]^n \to 2^V$ denote the corresponding independent set found by θ such that

$$\mathcal{I}(\theta) = \{v \in V \mid \theta^*_v \geq \alpha\}\,, \tag{4}$$

for any $\alpha > 0$. We show that $|\mathcal{I}(\theta^*)| = |U|$. Intuitively, this is tantamount to choosing the trained parameter indices in θ whose values exceed some threshold and choosing the vertices in V corresponding to those indices to be the MIS. The fixed parameters of f are constructed from the given graph $G = (V, E)$ as follows. The first $n \times n$ submatrix of W represents the nodes V in the graph and its weights are set equal to the identity matrix I_n. The following m columns of W correspond to the edges E in the graph. In particular, for the column associated with a given edge, a value of 1 is assigned to the entries corresponding to both ends of that edge and 0 otherwise. For the bias vector b, we assign a value of $-1/2$ to the entries corresponding to the first n nodes, and a value of -1 to the m entries corresponding to the edges. Finally, these nodes are input to their corresponding ReLU activation functions. For the vector w connecting the second hidden layer to the output layer, values of -1 and n are assigned to entries corresponding to nodes and edges in the second hidden layer, respectively. Hence, the parameters W, b, and w are defined as

$$\begin{aligned} W(i, i) &= 1,\ \ v_i \in V,\ \ i \in [n]\,, \\ W(i, n + l) &= W(j, n + l) = 1\,, \forall e_l = (v_i, v_j) \in E, l \in [m]\,, \end{aligned} \tag{5}$$

$$\begin{aligned} b(i) &= -1/2,\ \ w(i) = -1,\ \ v_i \in V,\ \ i \in [n]\,, \\ b(n + l) &= -1,\ \ w(n + l) = n,\ \ l \in [m]\,. \end{aligned} \tag{6}$$

Therefore, we can rewrite (3) as follows

$$f(\theta) = -\sum_{v\in V} \sigma(\theta_v - 1/2) + n \sum_{(u,v)\in E} \sigma(\theta_u + \theta_v - 1)\,. \tag{7}$$

Fig. 2 presents an example. The following theorem establishes a relation between the minimum value of (7) and the size of the MIS.

Theorem 4.1. *Given a graph $G = (V, E)$ and its corresponding dNN f, an MIS $U \subseteq V$ of G is of size $|U| = k$ if and only if the minimum value of f is $-k/2$.*

Proof ($\implies$)**.** Assume that $|U| = k$ and let $\theta_{v_i} = 1$ for each $v_i \in U$ and $\theta_{v_i} = 0$ otherwise. For an arbitrary pair of nodes $v_i, v_j \in U$, consider the output of f as visualized by Fig. 3, where edge values denote the outputs of the preceding nodes in the network and nodes η^1_i, η^2_i denote the ith neurons in the first and second hidden layers, respectively. We abuse notation to refer to both the output neuron and the output value as $f(\theta)$.

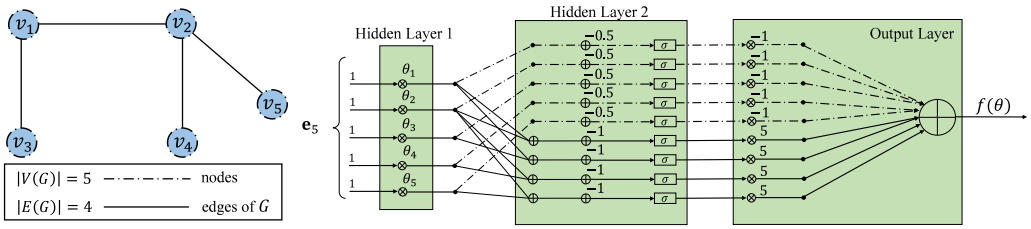

Fig. 2. An example of graph $G = (V = \{v_1, v_2, v_3, v_4, v_5\}, E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5)\})$ and its dNN construction f for the MIS problem.

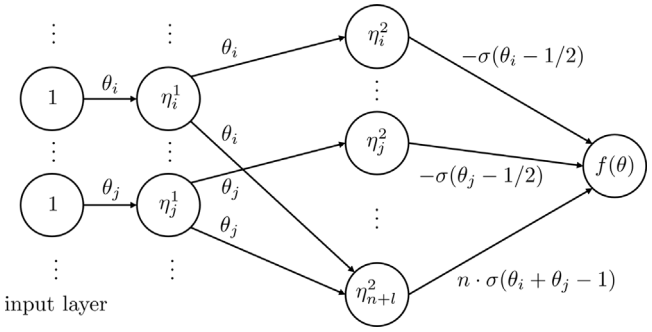

Fig. 3. Network f used in the proof of Theorem 4.1.

Note that v_i and v_j each contribute an output of $-1/2$ to $f(\theta)$. This follows from the fact that, by definition of MIS, these vertices do not share an edge and so the output of η^2_{n+l} is 0. Thus, for an MIS of size $|U| = k$, we have $f(\theta) = -k/2$. This is the minimum value attainable by f. Indeed, consider, for the sake of contradiction, that there exists θ' such that $f(\theta') < f(\theta)$. As with θ, this θ' must be defined such that $\theta_{v_i} = 1$ for each $v_i \in U$. Consider the addition of some other $v_{k'} \notin U$. Then $\eta^2_{k'}$ will contribute $-\sigma(\theta_{k'} - 1/2)$ to $f(\theta')$ and η^2_{n+l} will contribute at least $n\theta_{k'}$ for every edge $e_k = (v_{k'}, v) \in E$, where $v \in U$. By definition of MIS, some such edge must exist. Therefore, we would have $f(\theta') > f(\theta)$, yielding a contradiction.

($\Longleftarrow$) Assume that the minimum value of f is $f(\theta) = -k/2$. Consider an arbitrary edge $e_l = (v_i, v_j) \in E$. It follows from the construction of f that $\theta_{v_i} + \theta_{v_j} \leq 1$ must hold for f to achieve its minimum value. For the sake of contradiction, consider that $\theta_i + \theta_j > 1$. In this case, neuron η^2_{n+l} contributes $n(\theta_i + \theta_j - 1) > 0$ to the output $f(\theta)$. This yields a contradiction since we can simply choose θ_i and θ_j to be 0, thereby contributing a value of 0 to $f(\theta)$. Given an arbitrary vertex v and its neighbors $N(v)$, it must be the case that $\theta_i = 1$ for some $v_i \in N(v)$ as this would contribute a value of $-1/2$ to $f(\theta)$ through node η^2_i and a value of 0 through nodes η^2_{n+l} for all vertices $u \in N(v)$ connected to v through edge e_l. It follows that there must be k entries in θ with value 1, each contributing a value of $-1/2$ to the output $f(\theta)$ such that none of them share a neuron in the second hidden layer. These correspond to k vertices in V that do not share edges. Therefore, $|U| = k$. □

From Theorem 4.1, it follows that the minimum value of f is achieved when the maximum number of entries in θ have value 1 such that their corresponding nodes in G share no edges. This yields an independent set $\mathcal{I}(\theta) = \{v_i \in V \mid \theta_{v_i} = 1\}$ of maximum cardinality.

Due to the non-linearity introduced by the ReLUs in the dNN f, we obtain a minimizer for f by leveraging the backpropagation algorithm along with the well-known ADAM optimizer (Kingma & Ba, 2015). This is applied only to optimize the adjustable parameters θ. Hence, the ADAM optimizer updates θ depending on the computation of only one gradient, that is, $\nabla_\theta |f(\theta) - f_d|^2$. Given that W, b, and w represent the graph structure, they do not need to be updated, thus computing gradients w.r.t. to these parameters is not required.

As such, we iteratively minimize the loss function $\mathcal{L}(f(\theta), f_d) = |f(\theta) - f_d|^2$, where $|\cdot|$ denotes the absolute value and f_d is the minimum desired value used for parameter tuning. Per Theorem 4.1, the minimum achievable value of $f(\theta)$ is a function of the size of the MIS. Therefore, during training we select $f_d = -n/2$, a value that is only attained by $f(\theta)$ if G is a null graph.

4.2. On the duality of MIS and MC

Since the graph induced by the MIS is a null graph on G and fully-connected on its complement G', we propose to include the edges of G' in the construction of f to enhance the tuning of the parameters θ. We term the resulting enhanced dataless neural network as h with output value $h(\theta)$. In this case, we extend the definition of the fixed parameters $W \in \{0, 1\}^{n\times(n+m+m')}$, $b \in \{-1, -1/2\}^{n+m+m'}$, and $w \in \{-1, n\}^{n+m+m'}$, by defining the mapping for the augmented portion of these parameters representing the m' edges of G' as

$$W(i, n+m+l) = W(j, n+m+l) = 1, \quad \forall e_l = (v_i, v_j) \in E(G'), l \in [m'], \tag{8}$$

$$b(n+m+l) = w(n+m+l) = -1, \ l \in [m']. \tag{9}$$

The construction of the graph-specific dNN h requires a total of $n^2 + n(m+m'+3) + 2m + 2m'$ parameters of which $n^2 + n(m+m'+2) + 2m + 2m'$ parameters are fixed and n parameters are adjustable.

Given (7), (8), and (9), the output of h is

$$h(\theta) = f(\theta) - \sum_{(u,v)\in E(G')} \sigma(\theta_u + \theta_v - 1). \tag{10}$$

Fig. 4 presents an example of the proposed construction from a simple 5-node graph G to its corresponding dNN h. Our next result is analogous to Theorem 4.1 and establishes a relation between the minimum value of (10) and the size of an MIS in G.

Theorem 4.2. *Given a graph $G = (V, E)$ and its corresponding enhanced dNN h, an MIS $U \subseteq V$ of G is of size $|U| = k$ if and only if the minimum value of h is $-k^2/2$.*

Proof. Per Theorem 4.1, the minimum value of the first term of (10) is $-k/2$. Therefore, we consider the minimum value of the remaining (second) term, which corresponds to the edges of G'. Similar to Theorem 4.1, assume that $\theta_v = 1$ for each $v \in U$ and $\theta_v = 0$ otherwise. The graph induced by the MIS w.r.t. G' is a fully-connected graph, i.e., $|E(G'[U])| = k(k-1)/2$. Given the -1 bias, the outputs associated with the edges of G' will be 1. Since the subgraph induced on G' is complete, we get $-k(k-1)/2$ for the second term. The combined output is thus $-(k/2) - (k(k-1)/2) = -k^2/2$, which concludes the proof. □

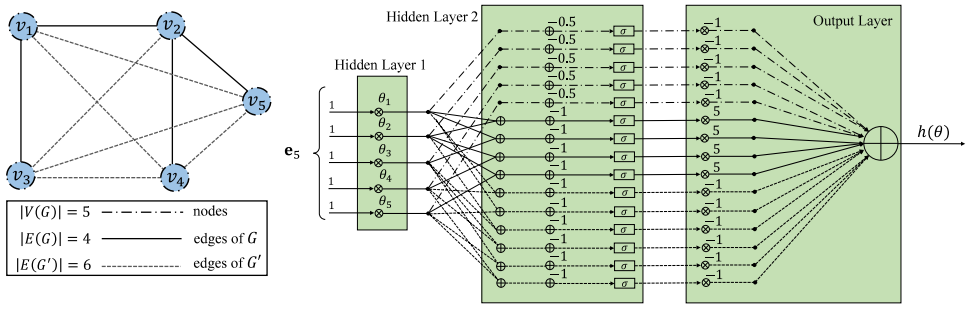

Fig. 4. An example of graph $G = (V = \{v_1, v_2, v_3, v_4, v_5\}, E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5)\})$ and its dNN construction h for the MIS problem by leveraging the duality between MIS and MC.

Algorithm 1 Finding MIS with dNN

Function: $\mathcal{I} = \texttt{dNN}(G, \alpha)$
1: **construct** h from G using (5), (6), (8), and (9)
2: **initialize** θ as in (12), $\mathcal{I}(\theta) = \emptyset$
3: **while** $\exists v \in V \setminus \mathcal{I}(\theta)$ **s.t.** $E(G[\mathcal{I}(\theta) \cup \{v\}]) \neq \emptyset$
4: **update** $\theta \leftarrow \text{argmin}_{\theta \in [0,1]^n} |h(\theta) - h_d|^2$
5: **obtain** $\mathcal{I}(\theta) = \{v \in V \mid \theta_v \geq \alpha\}$

Since the minimum value of h is $-k^2/2$, we use $h_d = -n^2/2$ for training the dNN by minimizing the loss

$$\mathcal{L}(h(\theta), h_d) = |h(\theta) - h_d|^2 . \tag{11}$$

In the case of (11), the backpropagation is used to optimize the adjustable parameters θ only. This is applied using the ADAM optimizer that updates θ depending on the computation of only $\nabla_\theta |h(\theta) - h_d|^2$.

In general, the vertices with high degrees are less likely to be part of an MIS than vertices with low degrees. Therefore, to speed up the training of the parameters of h, we initialize every element of θ with a probability that is decreasing in the node degree as

$$\theta_v = 1 - \frac{\text{d}(v)}{\Delta(G)} + s\,, \quad \theta \leftarrow \frac{\theta}{\max_{v \in V} \theta_v}\,, \tag{12}$$

where we add a small value s drawn from a uniform distribution over small positive bounds ($\ll 0.1$) as part of the ADAM stochastic algorithm to improve performance when optimizing the loss function of h (Goodfellow, Bengio, & Courville, 2017).

The complexity of optimizing over the proposed dNN depends on the ADAM algorithm that solves a minimization problem with a number of box-constrained variables equal to the number of nodes in the graph. The ADAM optimizer is a stochastic gradient-based method by which, in every iteration (update of parameters), the complexity depends on the computation of the gradient $\nabla_\theta \mathcal{L}(h(\theta), f_h)$. This is of the same computational complexity as evaluating the objective in (11) (Kingma & Ba, 2015).

4.3. Scaling up: Community detection approach

To handle large-scale graphs for the MIS problem, many techniques have been introduced in prior work, including the linear programming (LP) reduction, removal of pendant vertices, and other heuristics as presented in Akiba and Iwata (2016) and adopted in some of the most recent state-of-the-art methods. However, these techniques are only applicable to sparse graphs as pointed out in Lamm et al. (2016).

This motivates our work here on developing a reduction technique that is independent of the graph type and density. The proposed procedure described next utilizes multiple dNNs to handle large graphs, which is particularly useful when optimizing the parameters of one dNN representing the entire graph requires a large amount of storage and computation.

We perform community detection (Yang, Algesheimer, & Tessone, 2016) to partition the graph into communities, which are groups of nodes with dense connections internally and sparser connections between groups. Then, using Algorithm 1, we obtain a MIS for the subgraph induced by each community separately. Subsequently, a MIS is obtained for the full graph by processing the identified sets. More specifically, let $C_i, i \in [r]$ denote the set of nodes in community i, where r is the total number of communities found by a community detection algorithm. The inter-cluster edge set

$$R = \{(u, v) \in E \mid u \in C_i, v \in C_j, i \neq j\}\,, \tag{13}$$

is the set of edges between nodes in different communities. For every C_i, we construct a dNN h_i and obtain an MIS $\mathcal{I}_i = \texttt{dNN}(G_i, \alpha)$, where $G_i = G[C_i]$. The union of these sets is the set $B = \bigcup_{i \in [r]} \mathcal{I}_i$. Note that B is generally not an IS w.r.t. graph G since there could exist edges between nodes in the solution sets of two different communities. We call these the forbidden edges and define the set

$$F = \{(u, v) \in R \mid u \in \mathcal{I}_i, v \in \mathcal{I}_j, i \neq j\}\,. \tag{14}$$

In order to obtain a MIS w.r.t. G, we need to handle all nodes with edges in the set F. To this end, we develop the following procedure which processes every pair in F until an IS is obtained w.r.t. G. First, we select a pair $(u, v) \in F$, then for every node $q \in \{u, v\}$, we check if it can be replaced by a node from its neighborhood. A candidate replacement, $w \in N(q)$, must be 1-tight, that is $|B \cap N(w)| = 1$. If no replacements are found for either u or v, we remove the node with the larger number of repetitions in F. This is repeated until the set F is empty. The entire procedure is presented in Algorithm 2. In the case that the resulting set $\mathcal{I}$ is only an IS w.r.t. G, we obtain a MIS by executing Algorithm 1 on the subgraph induced by nodes that are neither in the solution nor in its neighborhood. In particular, $\mathcal{I}$ is updated as

$$\mathcal{I} \leftarrow \mathcal{I} \cup \texttt{dNN}(G[V \setminus (\mathcal{I} \cup N(\mathcal{I}))], \alpha)\,. \tag{15}$$

The complexity of the proposed CD approach depends on the CD algorithm utilized, the post-processing step presented in Algorithm 2, and Eq. (15). The complexity of the Algorithm depends on the size of set F, and the number of removed and replaced nodes in steps 6 and 10, respectively. If replacements and/or removals result in a MIS, then (15) is not needed. In the case where the output of the algorithm is an IS, the number of removed and replaced nodes determine the size of graph $G[V \setminus (\mathcal{I} \cup N(\mathcal{I}))]$ whose nodes and edges determine the size of the dNN needed to obtain a MIS from IS in (15).

Algorithm 2 Handling Forbidden edges.

Input: Graph $G = (V, E)$, B, F
Output: IS $\mathcal{I}$ on G
1: **initialize** $\mathcal{I} = B$
2: **while** $F \neq \emptyset$
3: **select** a pair $(u, v) \in F$, **initialize** ReplacementFlag $= 0$
4: **for all** $q \in \{u, v\}$
5: **if** $\exists w \in N(q)$ **s.t.** $|\mathcal{I} \cap N(w)| = 1$
6: **replace** q by w, that is $\mathcal{I} \leftarrow \mathcal{I} \setminus \{q\}$, $\mathcal{I} \leftarrow \mathcal{I} \cup \{w\}$
7: **update** F, ReplacementFlag $= 1$
8: **break for**
9: **if** ReplacementFlag $= 0$ (no replacement is found)
10: **remove** either u or v depending on their repetitions in F
11: **update** $\mathcal{I}$ and F

Algorithm 3 Solution improvement by dNNs

Input: Graph $G = (V, E)$, Solution $\mathcal{I}$, λ, IncreaseStep
Output: $\mathcal{I}^*$
1: **initialize** $\mathcal{I}^* = \mathcal{I}$
2: **while** stopping criteria is not satisfied
3: **obtain** $\mathcal{U} \subset \mathcal{I} : |\mathcal{U}| = \lambda, \forall u \in \mathcal{U}, v \in \mathcal{I} \setminus \mathcal{U}, \mathrm{d}(u) \leq \mathrm{d}(v)$
4: **obtain** $\mathcal{I} \leftarrow \mathcal{I} \cup \mathtt{dNN}(G[V \setminus (\mathcal{U} \cup N(\mathcal{U}))], \alpha)$
5: **if** $|\mathcal{I}| > |\mathcal{I}^*|$ (update the optimal if $\mathcal{I}$ is of higher cardinality)
6: **update** $\mathcal{I}^* = \mathcal{I}$
7: **if** $|\mathcal{I}| \leq |\mathcal{I}^*|$ (restart from the current optimal)
8: **update** $\mathcal{I} = \mathcal{I}^*$
9: **update** $\lambda \leftarrow \lambda +$ IncreaseStep

4.4. Solution improvement by dNNs

After applying the community detection algorithm and using Algorithm 1 for every resulting subgraph, Algorithm 2 along with a possible use of (15) are used to obtain MIS $\mathcal{I}$. Since high-degree nodes are less likely to be in a large IS, given graph G and solution $\mathcal{I}$, we propose a solution improvement procedure that removes a set of low-degree nodes $\mathcal{U} \subset \mathcal{I}$, such that $|\mathcal{U}| = \lambda$, along with their neighbors $N(\mathcal{U})$ from the graph. We then apply our dNN on the reduced graph $G[V \setminus (\mathcal{U} \cup N(\mathcal{U}))]$ with different initial seed for s as a form of simulated annealing (Van Laarhoven & Aarts, 1987). This procedure is iteratively applied where we increase λ at every iteration. The best solution is maintained until some stopping criteria is met. The procedure is given in Algorithm 3. While a similar criteria is used to select $\mathcal{U}$ in Lamm et al. (2016), their method recursively tries all reduction techniques on the reduced graph where λ is fixed.

The number of computations needed for Algorithm 3 depend on λ and the rate by which it increases as this determines the size of the subgraph on which the dNN procedure is applied at each iteration (step 4).

5. Experimental evaluation

In this section, we evaluate the performance of our proposed method and present comparisons to state-of-the-art methods using synthetic graphs and real-world benchmarks.

5.1. Setup, benchmarks, and baselines

We process graphs using the NetworkX library (Hagberg, Schult, & Swart, 2008) and use Tensorflow (Abadi et al., 2016) to construct the dNN h. The initial learning rate for the ADAM optimizer is set to 0.1. To tune the parameters θ of h, a set of repeated samples of the pair $(\mathbf{e}_n, -n^2/2)$ is used as the dataset. We set the probability threshold $\alpha = 0.5$ and use degree-based initialization. Experiments justifying our choice are presented in later subsections. For community detection, we use the Louvain algorithm (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008) with a resolution factor of 1.3 for large-scale low-density graphs and 0.8 for high-density instances. For Algorithm 3, we choose $\lambda = 5$ and increase it by 1 in every iteration. The algorithm stops when the number of nodes in the reduced graph is below 20. The experiments are run using Python 3 and Intel(R) Core(TM) i9-9940 CPU @ 3.30 GHz machine. The code for the experiments conducted in our paper is available online.[1]

For low-density graphs, we incorporate the inexpensive and non-recursive LP graph reduction presented in Nemhauser and Trotter (1975) prior to performing community detection and constructing the enhanced dNN h. A half-integral solution (using values 0, 1/2, and 1), $x^* = \arg\max\{\sum_{v \in V} x_v \text{ s.t. } x_v \geq 0, \forall v \in V, x_v + x_u \leq 1, \forall (u, v) \in E\}$ is obtained using bipartite matching. The vertices that are members of set $T =: \{v \in V \mid x_v^* = 1\}$ must be in the MIS and can thus be removed from G together with their neighbors in $N(T)$. The solution obtained from training h on $G[V \setminus (T \cup N(T))]$ is joined with nodes in T to obtain the MIS for G. Furthermore, we implement the 2-improvement basic local search algorithm (Andrade et al., 2012). The foregoing techniques are also used in most of the state-of-the-art solvers presented in Lamm et al. (2016), Li et al. (2018) and Ahn, Seo, and Shin (2020).

As a benchmark, we use the social network graphs from the Stanford Large Network Dataset Collection given in SNAP (Leskovec & Krevl, 2014). In these graphs, the vertices represent people and the edges reflect their interactions. We also use the citation network graphs (Sen et al., 2008) for data collected from academic search engines. In these graphs, nodes represent documents and edges reflect their citations. Using the aforementioned benchmarks, we compare the performance of our proposed framework to multiple MIS solvers, including the GCN method (Li et al., 2018), which is an ML-based approach, and the RL-based method S2V-DQN (Dai et al., 2017). We also report results from the state-of-the-art MIS solver ReduMIS (Lamm et al., 2016). We use the size of the identified independent set to measure the quality of the solution for every baseline considered. Furthermore, results from solving the MIS Integer Linear Program (ILP) in (16) using CPLEX are also reported.

$$\max_{x} \sum_{v \in V(G)} x_v \text{ subject to } x_v \in \{0, 1\}, \forall v \in V, \quad x_v + x_u \leq 1, \forall (v, u) \in E. \tag{16}$$

The aforementioned benchmarks are considered sparse graphs. Therefore, we also test our proposed method on higher-density graphs randomly generated from the Erdos–Renyi (ER) (Erdos, Rényi, et al., 1960), Barbosi–Albert (BA) (Albert & Barabási, 2002), Holme and Kim (HK) (Holme & Kim, 2002), and the Stochastic Block (SBM) (Holland, Laskey, & Leinhardt, 1983) models.

5.2. Results on SNAP and citation network benchmarks

In this subsection, we present the overall comparison results with the GCN, ReduMIS, and S2V-DQN methods along with the results from solving the MIS ILP in (16) using CPLEX. Columns 4 to 8 of Table 1 present the size of the found MIS. The results, other than the CPLEX ILP, reported for the baselines are obtained from Table 5 of Li et al. (2018).

We briefly describe the reduction techniques utilized by these baselines as they contribute significantly to their final results. All three methods remove pendent, unconfined, and twin vertices and utilize vertex folding for degree-2 nodes. Additional

[1] https://github.com/ialkhouri/MIS_dNNs.

Table 1
Comparison to state-of-the-art baselines using real-world benchmarks in terms of the size of the identified MIS.

Dataset	$\|V\|$	$\|E\|$	GCN	ReduMIS	S2V-DQN	CPLEX	dNNs
bitcoin-alpha	3783	14 124	**2718**	**2718**	2705	**2718**	**2718**
bitcoin-otc	5881	21 492	4346	4346	4334	4346	**4347**
wiki-Vote	7115	100 762	**4866**	**4866**	4779	**4866**	**4866**
soc-slashdot0811	73 399	497 274	**53 314**	**53 314**	52 719	**53 314**	**53 314**
soc-slashdot0922	82 168	582 533	**56 398**	**56 398**	55 506	**56 398**	56 395
soc-Epinions1	75 579	405 740	**53 599**	**53 599**	53 089	**53 599**	53 598
Citeseer	3327	4536	**1867**	**1867**	1705	1808	1866
Cora	2708	5429	**1451**	**1451**	1381	**1451**	**1451**
PubMed	19 717	44 327	**15 912**	**15 912**	15 709	**15 912**	**15 912**

Table 2
Comparison to state-of-the-art baselines using synthetic graphs in terms of the average size of the MIS.

Graph type	$\|V\|$	$\mathbb{E}(\|E\|)$	ReduMIS	GCN	CPLEX	dNNs
ER	100 ($p = 0.1$)	496	**30.5**	**30.5**	**30.5**	**30.5**
ER	100 ($p = 0.2$)	975	**20**	19.5	**20**	**20**
ER	200 ($p = 0.1$)	1991.5	**41**	40	**41**	**41**
ER	200 ($p = 0.2$)	3983.5	**25.5**	24	**25.5**	**25.5**
SBM	250 ($p = 0.1$)	1857.5	**61**	60.5	60.5	**61**
SBM	250 ($p = 0.2$)	2431	**51**	50.5	**51**	**51**
SBM	350 ($p = 0.1$)	3614.5	**68**	**68**	66.5	**68**
SBM	350 ($p = 0.2$)	4826	**55.5**	55	53.5	**55.5**
BA	100	2450	**45**	**45**	**45**	**45**
BA	200	9950	**95**	**95**	**95**	**95**
HK	100	2500	**30**	**30**	**30**	**30**
HK	200	9900	**60**	**60**	**60**	**60**

reductions are also considered in ReduMIS, including finding node alternatives, using packing constraints, and adopting the same LP MIS relaxation reduction we adopt in this paper. We refer the interested reader to Akiba and Iwata (2016) for a thorough discussion of these reductions.

In all datasets considered, our method outperforms S2V-DQN. Our method performs mostly on par with ReduMIS and GCN, both of which yield identical results. This is observed as exact MIS sizes are obtained for bitcoin-alpha, Wiki-Vote, soc-slashdot0811, Cora, and PubMed. While scoring higher for bitcoin-otc, our method scores lower for soc-slashdot0922, soc-Epinions1, and Citeseer. When compared to the ILP solver, our method outperforms CPLEX on bitcoin-otc and Citeseer while performing on par for all the other instances other than soc-slashdot0922 and soc-Epinions.

5.3. Results on synthetic graphs

In this subsection, we compare our proposed method to ReduMIS, GCN, and CPLEX using random graphs generated from the ER, BA, and HK models. Every size reported in Table 2 represents the average of two random graphs from the model given in the first column. For ER, p represents the probability of an edge being present. For BA, we use $\lfloor 0.5n \rfloor$ and $\lfloor 0.45n \rfloor$ edges to attach a new node to existing nodes. For HK, we add $\lfloor 0.3n \rfloor$ random edges to each new node and set the probability of adding a triangle after adding a random edge to 0.5. For SBM, we generate graphs with 5 clusters, where two nodes from the same cluster share an edge with probability p (intra-cluster density) and two nodes from different clusters share an edge with probability $q = 0.05$ (inter-cluster density). We set a 10-minute time limit for ReduMIS, GCN, and CPLEX and present the size of the largest independent set found within that time. The random graph parameters are selected to yield high-density graphs relative to the earlier benchmarks. As shown, for all the considered random high-density graphs, on average we attain the same MIS sizes of ReduMIS, while performing on par or outperforming the state-of-the-art learning-based method GCN or CPLEX. This is achieved without the use of labeled graphs for training.

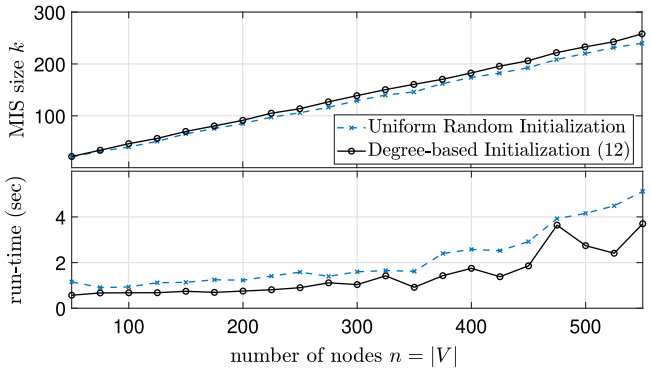

Fig. 5. Comparison between the proposed degree-based initialization and the initialization from a uniform random distribution in terms of the average size of the found MIS (*top*) and the average execution time (*bottom*).

5.4. Advantage of degree-based initialization

This subsection highlights the positive impact of utilizing the degree-based initialization, measured by the size of the MIS found and the execution time required to obtain a solution. The results, presented in Fig. 5, are compared to initializing θ from a uniform distribution over the interval $[0, 1]$. Every point in Fig. 5 (*top*) ((*bottom*)) represents the average size of the found MIS (execution time) of 5 uniformly generated random graphs with n nodes and $m = 2n$ edges. As observed, on average, our proposed method in (12) outperforms random initialization in terms of the average MIS size and also incurs shorter execution time.

5.5. Impact of the choice of α

In this experiment, we examine the impact of the probability threshold α on the number of training steps required, the execution time, and the output value of the network. Fig. 6 shows the results for graphs generated uniformly at random with $n = 100$ and $m = 500$. From top to bottom, the results show the average number of tuning steps required (*first*), average execution time (*second*), average normalized output values given by $-2h(\theta)/k^2$ (*third*), and the average obtained MIS size k (*fourth*). As observed, when α increases, the output $h(\theta)$ of the dNN also increases along with the number of tuning steps and the required run-time. However, beyond $\alpha \approx 0.55$, the average size k of the obtained MIS does not change. Therefore, to increase the size of the MIS while reducing the run-time and the number of steps, we set $\alpha = 0.5$ in step 5 of Algorithm 1.

The re-scaled zoomed curves in the inset of Fig. 6 (*third*) shows that the optimal normalized output value approaches 1 as $\alpha \to 1$. This verifies that the minimum value of $h(\theta)$ occurs when $\theta \in \{0, 1\}^n$. This result was established in the proof of Theorems 4.1 and 4.2.

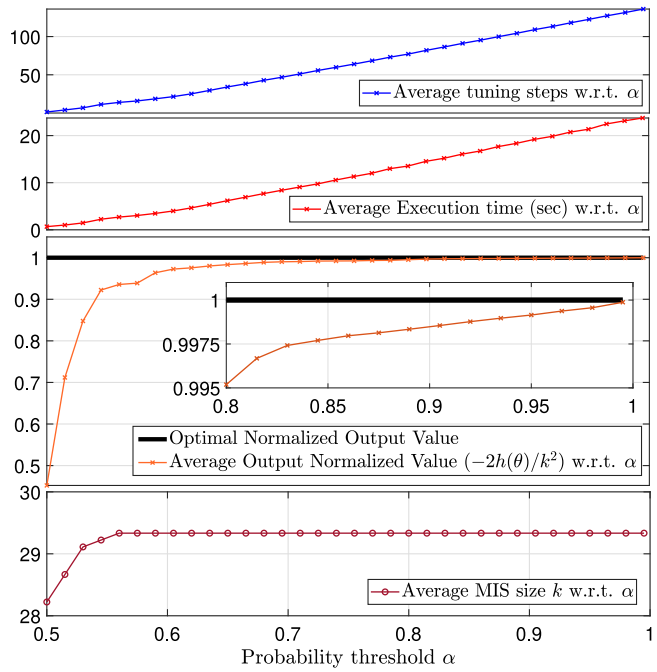

Fig. 6. The impact of the probability threshold α w.r.t. the average number of tuning steps (*first*), average execution time (*second*), average normalized output value with re-scaled zoomed values (*third*), and average size k of the MIS found (*fourth*).

6. Conclusion

We presented a dataless, differentiable methodology for solving the Maximum Independent Set (MIS) problem that is radically different from existing techniques. The underpinning of our approach is a reduction from the MIS problem to an equivalent dataless Neural Network (dNN) constructed from the given graph. The parameters of this dNN are trained without requiring data, thereby setting our approach apart from learning-based methods like supervised, unsupervised, and reinforcement learning. In particular, training is conducted by applying backpropagation to a loss function defined entirely based on the structure of the given graph. We also presented an enhanced version of the dNN by incorporating the edges from the complement graph and exploiting the duality of the Maximum Clique (MC) and MIS problems. Additionally, we developed a reduction procedure that leverages a community detection algorithm to scale our approach to larger and higher-density graphs. Unlike previous reductions, the procedure is independent of the type of the graph and its density. Experimental results on both real and synthetic benchmarks demonstrate that our proposed method performs on par with state-of-the-art learning-based methods in terms of the size of the found MIS without requiring any training data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by Air Force Research Laboratory Contract FA8750-20-3-1004, Air Force Office of Scientific Research Award 20RICOR012, National Science Foundation CAREER Award CCF-1552497, National Science Foundation Award CCF-2106339, and Department of Energy Award DE-EE0009152.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (pp. 265–283).

Ahn, S., Seo, Y., & Shin, J. (2020). Learning what to defer for maximum independent sets. In *International conference on machine learning* (pp. 134–144). PMLR.

Akiba, T., & Iwata, Y. (2016). Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theoretical Computer Science*, *609*, 211–225.

Albert, R., & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*(1), 47.

Andrade, D. V., Resende, M. G., & Werneck, R. F. (2012). Fast local search for the maximum independent set problem. *Journal of Heuristics*, *18*(4), 525–547.

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

Berman, P., & Schnitger, G. (1992). On the complexity of approximating the independent set problem. *Information and Computation*, *96*(1), 77–94.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008.

Boppana, R., & Halldórsson, M. M. (1992). Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, *32*(2), 180–196.

Cook, W., Lovász, L., Seymour, P. D., et al. (1995). *Combinatorial optimization: papers from the DIMACS Special Year, Vol. 20*. American Mathematical Soc..

Dai, H., Dai, B., & Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning* (pp. 2702–2711). PMLR.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. 2017. Learning combinatorial optimization algorithms over graphs. In *Proceedings of the 31st international conference on neural information processing systems*. (pp. 6351–6361).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, *29*, 3844–3852.

Erdos, P., Rényi, A., et al. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, *5*(1), 17–60.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep learning (adaptive computation and machine learning series). *Cambridge Massachusetts*, 321–359.

Hagberg, A. A., Schult, D. A., & Swart, P. J. 2008. Exploring network structure, dynamics, and function using networkx. In Varoquaux, G. and Vaught, T. and Millman, J. (Eds.), *Proceedings of the 7th python in science conference*. (pp. 11–15). Pasadena, CA USA.

Hastad, J. (1996). Clique is hard to approximate within n/sup 1-/spl epsiv. In *Proceedings of 37th conference on foundations of computer science* (pp. 627–636). IEEE.

He, H., Daume III, H., & Eisner, J. M. (2014). Learning to search in branch and bound algorithms. *Advances in Neural Information Processing Systems*, *27*, 3293–3301.

Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, *5*(2), 109–137.

Holme, P., & Kim, B. J. (2002). Growing scale-free networks with tunable clustering. *Physical Review E*, *65*(2), Article 026107.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Lamm, S., Sanders, P., Schulz, C., Strash, D., & Werneck, R. F. (2016). Finding near-optimal independent sets at scale. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)* (pp. 138–150). SIAM.

Leskovec, J., & Krevl, A. (2014). SNAP datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

Li, Z., Chen, Q., & Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. In *NeurIPS*.

Nemhauser, G. L., & Trotter, L. E. (1975). Vertex packings: Structural properties and algorithms. *Mathematical Programming*, *8*(1), 232–248.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE international conference on neural networks* (pp. 586–591).

San Segundo, P., Rodríguez-Losada, D., & Jiménez, A. (2011). An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, *38*(2), 571–581.

Schuetz, M. J., Brubaker, J. K., & Katzgraber, H. G. (2021). Combinatorial optimization with physics-inspired graph neural networks. arXiv preprint arXiv:2107.01188.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93.
Steger, A., & Wormald, N. C. (1999). Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, *8*(4), 377–396.
Tarjan, R. E., & Trojanowski, A. E. (1977). Finding a maximum independent set. *SIAM Journal on Computing*, *6*(3), 537–546.
Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: theory and applications* (pp. 7–15). Springer.
Williamson, D. P., & Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge University Press.
Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, *6*(1), 1–18.