A differential approach for several NP-hard optimization problems

Sangram K. Jena[*] and **K. Subramani**[†] and **Alvaro Velasquez** [‡]

Abstract

In this paper, we study a variety of **NP-hard** optimization problems, such as MAXCUT, MAXkSAT, and MAXNAE2SAT from the perspective of obtaining exact solutions. We derive differentiable functions for each of these problems using the dataless neural networks framework. Recently, it was shown that a single differentiable function on a dataless neural network can capture the Maximum Independent Set problem. Inspired by this design, we design dataless neural networks for a host of combinatorial optimization problems. We also establish the correctness of our derivations in a rigorous fashion.

1 Introduction

NP-hard optimization problems have applications in almost every domain, such as scheduling, routing, telecommunications, planning, transportation, and decision-making processes (Bengio, Lodi, and Prouvost 2021; Festa 2014). These problems do not admit a polynomial-time efficient solution unless some well-established complexity-theoretic conjectures fail. While it is challenging to solve such problems optimally, several techniques can approximate (Vazirani 2002) optimal solutions or even solve the problems efficiently in non-trivial exponential time (Fomin and Kratsch 2010). One method to handle these problems is by using neural network (NN) frameworks. However, this method requires extensive training of neural networks to generate the required solution. To overcome these problems, recently, an efficient method was developed using dataless neural network (dNN) frameworks (Alkhouri, Atia, and Velasquez 2022). Unlike NN frameworks, dNN frameworks do not require any data other than input instances.

[*]LDCSEE, West Virginia University, Morgantown, WV, USA sangramkishor.jena@mail.wvu.edu

[†]LDCSEE, West Virginia University, Morgantown, WV, USA k.subramani@mail.wvu.edu

[‡]Department of Computer Science, University of Colorado Boulder, Boulder, CO, USA alvaro.velasquez@colorado.edu

Let f be a conventional neural network parameterized by trainable parameters θ. Furthermore, assume that θ can be trained on some dataset $\{(x_i, y_i)\}$. Let x_i be an instance of a differentiable **NP-hard** optimization problem χ, and y_i be the values of the optimal solution of χ. To make the output $f(x_i;\theta)$ of f as close to y_i as possible, the parameters θ are typically updated using backpropagation by minimizing a differentiable loss function $L(x_i, f(x_i;\theta))$. The parameters are updated by the backpropagation in the direction of $\theta := \theta - \alpha \cdot \partial L(x_i, f(x_i;\theta))/\partial\theta$. Here, α controls the learning rate. The dataless neural networks are based on the concept that there is no data for training the neural network. So, what we have as an output of the neural network is simply $f(e_n;\theta) = f(\theta)$, where e_n is the all-ones vector representing a trivial input to the neural network. Thus, instead of finding patterns in some data set, dataless neural networks attempt to find the optimal solution to a given discrete optimization problem by enforcing a certain structure on f and θ.

The rest of this paper is organized as follows: In Section 2, we formally define the problems and the notations used in the paper. Section 3 describes related work in the literature. In Section 4, we design a dNN for the MAXCUT problem. Section 5 discusses a dNN for the MAXkSAT problem. Finally, we design a dNN in Section 6 for the MAXNAE2SAT problem.

2 Statement of Problems

In this section, we define the problems and some of the notations used in this paper.

Definition 1 **MAXCUT:** *Given a graph $G = (V, E)$, find a set $C \subseteq E$ that partitions the set V into two subsets S and T such that the number of edges between S and T is maximized.*

Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a collection of n Boolean variables. A literal is either a variable or the negation of a variable. A clause C is a disjunction of literals over the variable set X, for example, $C = (x_1 \vee x_2 \vee x_3 \vee x_4)$. A formula is said to be in Conjunctive

Normal Form (CNF) if it is the conjunction of clauses. For instance, $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$ is a CNF formula with three clauses over the variable set $X = \{x_1, x_2, x_3, x_4\}$. A CNF formula is said to be in kCNF form if each clause consists of exactly k literals.

Definition 2 *SAT: Given a CNF formula Φ, does there exist an assignment that satisfies each clause, i.e., sets at least one literal to* **true** *in each clause?*

Definition 3 *MAXSAT: Let Φ denote a formula in CNF with m clauses over n variables, find an assignment for the variables, which maximizes the number of satisfied clauses in Φ.*

Definition 4 *MAXkSAT: Let Φ denote a formula in kCNF with m clauses over n variables, find an assignment for the variables, which maximizes the number of satisfied clauses in Φ.*

Definition 5 *NAESAT: Given a CNF formula Φ, does there exist an assignment that nae-satisfies each clause, i.e., each clause is satisfied and at least one literal from each clause is set to* **false***?*

Note that unit clauses cannot be nae-satisfied. Only clauses of size at least two can be nae-satisfied.

Definition 6 *MAXNAEkSAT: Given a kCNF formula Φ with m clauses over n variables, find an assignment for the variables of Φ that maximizes the number of nae-satisfied clauses in Φ.*

In our design of dataless neural networks, we use a rectified linear (ReLU) activation function. It is a piecewise linear function that outputs the input directly if it is positive; otherwise, it outputs zero, i.e., $\sigma(x) = max(0, x)$. For any positive integer n, $[n] := \{1, 2, \ldots, n\}$. Unless mentioned otherwise, $|\cdot|$ represents the absolute value or modulus.

3 Related work

This section discusses some of the related work available in the literature for the neural network (NN) and dataless neural network (dNN). We discuss the available results for NNs and dNNs with respect to many combinatorial optimization problems (COPs). The most interesting COPs are **NP-hard**. It is well-known that such problems do not have polynomial time-efficient algorithms unless some established complexity-theoretic conjectures fail. Despite being inefficient, these problems have real-time applications in almost every domain, such as routing, scheduling, planning, telecommunications, transportation, and decision-making processes (Bengio, Lodi, and Prouvost 2021; Festa 2014). Due to the importance of these problems, a lot of research is going on to address them with different efficient, approximate solvers (Lamm et al. 2016). Broadly, these solvers are categorized into heuristic algorithms (Akiba and Iwata 2016), approximation algorithms (Boppana and Halldórsson 1992), and conventional branch-and-bound methods (San Segundo, Rodríguez-Losada, and Jiménez 2011). Such approaches may produce suboptimal solutions. Some of the other well-studied approaches to dealing with **NP-hard** problems use parameterized (Cygan et al. 2015; Flum and Grohe 2006; Niedermeier 2006) and exact exponential algorithmic techniques (Fomin and Kratsch 2010; Gaspers 2010).

However, another approach to address the COPs is to use the concept of machine learning (Bengio, Lodi, and Prouvost 2021; Wilder, Dilkina, and Tambe 2019). The use of reinforcement learning to automate the search of the heuristics for COPs is discussed in (Drori et al. 2020; Mazyavkina et al. 2021). These models require training based on the problems. More specifically, they rely on supervised learning using datasets of the combinatorial structures of interest drawn from some distribution of problem instances. In (Alkhouri, Atia, and Velasquez 2022), the authors introduced dNNs for which no data is required for training. By designing a single differentiable function, they captured the well-known combinatorial optimization problem, the maximum independent set (MIS) problem. They also designed a similar dNN structure for the maximum clique (MC) and minimum vertex cover (MVC) problems related to the MIS problem. In (Jena, Subramani, and Velasquez 2023), we developed dNNs tailored for solving the maximum dissociation set, k-coloring, and maximum cardinality distance matching problems.

The literature discusses several powerful heuristic solvers for the MIS problem. One of the heuristic solvers is ReduMIS (Lamm et al. 2016). It consists of two components. The first component is an iterative implementation of a series of graph reduction techniques. The second component is the use of an evolutionary algorithm. These methods usually involve extensive training of neural networks (NNs) using large graph datasets for which solutions are known. Another method for the MIS problem, similar to the method of dNN for the MIS problem discussed in (Alkhouri, Atia, and Velasquez 2022), was developed in (Schuetz, Brubaker, and Katzgraber 2022). The method discussed in (Schuetz, Brubaker, and Katzgraber 2022) uses a graph neural network and does not require training data. More specifically, its output is represented by the probability of each node being in the solution. In contrast to the method discussed in (Alkhouri, Atia, and Velasquez 2022), it uses a loss function to adjust its parameter that encodes the graph of interest. Furthermore, the approach discussed by Alkhouri et al. (Alkhouri, Atia, and Velasquez 2022) uses n trainable parameters where n is the number of vertices in the input graph. However, the number of tunable parameters used by the approach discussed in (Schuetz, Brubaker, and Katzgraber 2022) is large in size. It uses n parameters in its last layer only. In

(Alkhouri, Atia, and Velasquez 2022), the authors also showed some experimental results by comparing them with the best heuristics available in the literature. They evaluated success by taking the solution size obtained by ReduMIS as a benchmark. They also showed that their experimental results perform as well or outperform the state-of-the-art learning-based methods discussed in (Li, Chen, and Koltun 2018).

4 MAXCUT

In this section, we design a dataless neural network (dNN) for the MAXCUT problem. Note that the MAXCUT problem is a special case of the MAXNAE2SAT problem, where all the literals in the CNF formula are positive. Thus, the following dNN for the MAXCUT problem works for the MAXNAE2SAT problem with positive literals.

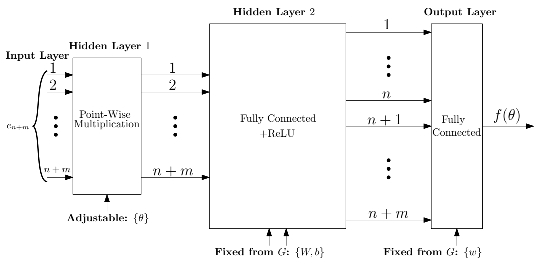


Figure 1: Block diagram of dNN.

Let $G = (V, E)$ be a graph with n vertices over m edges. We construct a dNN f with trainable parameters $\theta \in [0, 1]^{n+m}$ with respect to G. That means for each vertex $v \in V$, there is a corresponding trainable parameter θ_v in f. For each edge $e_i \in E$, there is a corresponding trainable parameter θ_{e_i} in f. The input to the dNN is an all-ones vector e_{n+m} which does not depend upon any data. The output of the dNN is $f(e_{n+m}; \theta) = f(\theta) \in \mathbb{R}$. There are four layers in the dNN for the MAXCUT problem. The four layers are categorized as one input layer, two hidden layers, and one output layer (see the block diagram in Figure 1 for the proposed network). The input layer e_{n+m} is connected with the first hidden layer through an elementwise product of the trainable parameters θ. The first hidden layer is connected to the second hidden layer by the binary matrix $W \in \{0, 1\}^{n \times (n+m)}$. The binary matrix is only dependent on G. At the second hidden layer, there exists a bias vector $b \in \{0, -\frac{1}{2}\}^{2 \cdot m}$. There is a fully connected weight matrix $w \in \{-1, -n\}^{2 \cdot m}$ in the second hidden layer to the output layer. Note that all the parameters are defined as a function of G. The output of f is given as follows:

$$f(e_{n+m}; \theta) = f(\theta) = w^T \cdot \sigma((W^T \cdot (e_{n+m} \odot \theta)) + b). \tag{1}$$

Here $\odot$ is the element-wise Hadamard product that represents the operation of the first hidden layer of the constructed network. The fully-connected second hidden layer consists of the fixed matrix W and a bias vector b with a ReLU activation function $\sigma(x) = max(0, x)$. The last layer is another fully-connected layer and is expressed in vector w.

On the other hand, we prove that when a max-cut $C \subseteq E$ in G is found, $f(\theta)$ attains its minimum value. Therefore, $f(\theta)$ is an equivalent differentiable function of the max-cut generated in G. Moreover, C can be constructed from θ as follows. Let $\theta^* = argmin_{\theta \in [0,1]^{n+m}} f(\theta)$ be an optimal solution to f. Let $I : [0, 1]^m \to 2^E$ be a max-cut corresponding to θ such that $I(\theta) = \{e \in E \mid \theta^*_e \geq \alpha\}$, for $\alpha > 0$. We show that $|I(\theta^*)| = |C|$. We choose the edges in the max-cut C in G corresponding to the indices of θ whose value exceeds a threshold (say α). From an input graph $G = (V, E)$, the fixed parameters of f can be constructed as follows: In the binary matrix W, the first $n \times n$ submatrix represents the vertices V of G. Its weights are set equal to the identity matrix I_n (see the 5×5 submatrix in Figure 2 (b) corresponding to the 5 vertices of G in Figure 2 (a)). Furthermore, the remaining m columns of W represent the edges of G and for each edge $e_i = uv \in E$, the value of $u = v = 1$ in the column (see the 6 columns e_1 to e_6 in Figure 2 (b) corresponding to the 6 edges of G in Figure 2 (a)). The bias vector consists of two parts, each with m entries resulting in $2 \cdot m$ entries. For each edge of G, the corresponding value of the first m entries is $-\frac{1}{2}$ in the biased vector b. For each edge, the corresponding value of another m entries in the bias vector is set to 0. Finally, the value of -1 is assigned in the entries corresponding to the edges of G in vector w. For another m entries, the value is set to $-n$ in w. Hence, the parameters W, b, and w are defined as follows:

$$\begin{gathered} W(i, i) = 1, v_i \in V, i \in [n], \\ W(i, n+k) = W(j, n+k) = 1, \\ \forall e_k = v_i v_j \in E, k \in [m], \end{gathered} \tag{2}$$

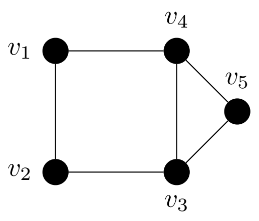
$$\begin{gathered} b(i) = -\frac{1}{2}, w(i) = -1, e_i \in E, i \in [m], \\ b(m+k) = 0, w(m+k) = -n, k \in [m]. \end{gathered} \tag{3}$$

The function in (1) can be rewritten as follows:

$$f(\theta) = -\sum_{e_i = uv \in E} (\sigma(\theta_{e_i} - \frac{1}{2}) - n \cdot \sigma(\theta_{e_i} - |\theta_u - \theta_v|)). \tag{4}$$

An example of the above discussed dNN construction is presented in Figure 3.

With the above dNN construction, we prove the following theorem to establish a relation between the solution of the MAXCUT problem and the minimum value of f.



(a) A graph G

$$W = \begin{array}{c|ccccccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline v_1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ v_2 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ v_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ v_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

(b) A binary matrix W

Figure 2: Representation of a binary matrix W corresponding to G.

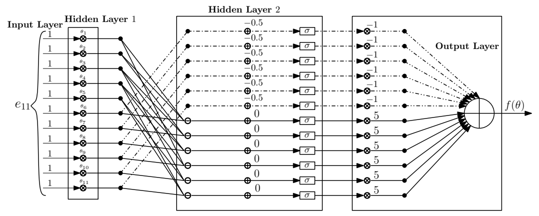


Figure 3: Construction of dNN f corresponding to the graph in Figure 2 (a) for the MAXCUT problem.

Theorem 1 *Let $G = (V, E)$ be a graph with m edges over n vertices. Let f be the corresponding dNN of G. G has a max-cut of size k, if and only if the minimum value of f is $-\frac{k}{2}$.*

Proof: Let $C \subseteq E$ be a max-cut of size k in G that partitions V into two sets S and T. For each $v \in V$, if $v \in S$, then set $\theta_v = 1$. Otherwise, set $\theta_v = 0$. For each edge $e_i \in E$, set $\theta_{e_i} = 1$, if $e_i \in C$; otherwise set $\theta_{e_i} = 0$. Consider the output f represented in Figure 4 for an arbitrary edge $e_i = uv \in E$.

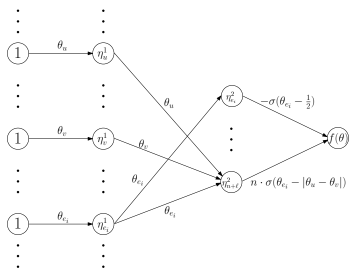


Figure 4: Output with respect to an arbitrary edge.

As per the construction of the dNN, the edge values denote the outputs of the preceding nodes in the network. Furthermore, the i^{th} neurons in the first hidden layer are denoted by η^1_i and the second hidden layer is denoted by η^2_i. If $e_i \in C$, then the values of both θ_u and θ_v are not equal. Moreover, $\theta_{e_i} = 1$. The above θ values of the corresponding edge e_i will contribute $-\frac{1}{2}$ to f. Furthermore, if $e_i = uv \notin C$, then the values of both θ_u and θ_v are equal and $\theta_{e_i} = 0$. The above θ values of the corresponding edge e_i will contribute 0 to f. There are k edges present in C. Therefore, it will contribute $-\frac{k}{2}$ to f.

Conversely, assume that the minimum value of the output function f is $f(\theta) = -\frac{k}{2}$. We construct a max-cut C for G of size k from f as follows: From the construction of the dNN, it is clear that, for each edge $e_i = uv \in E$, if $\theta_{e_i} = 1$, then the values of θ_u and θ_v are not equal. Otherwise, f does not achieve its minimum value. To prove this, assume that the values of θ_u and θ_v are equal and $\theta_{e_i} = 1$. It follows that the neuron $\eta^2_{n+\ell}$ contributes $n \cdot (\theta_{e_i} - |\theta_u - \theta_v|) > 0$ to the output $f(\theta)$. This is a contradiction to the fact that f achieves its minimum value. We can simply assign the value of $\theta_{e_i} = 0$ and reduce the value of f further. So, it is clear that for any edge $e_i = uv \in E$, if $\theta_{e_i} = 1$, then the values of θ_u and θ_v are not equal. Each such edge $e_i \in E$, which θ values of both the end vertices are not equal and $\theta_{e_i} = 1$ contributes $-\frac{1}{2}$ to $f(\theta)$ through η^2_i. Furthermore, it contributes a value of 0 to $f(\theta)$ through $\eta^2_{n+\ell}$. That means there are k entries of value 1 in θ corresponding to edges. For each $e_i = uv \in E$, consider the θ_u and θ_v values. If $\theta_u = 0$ ($\theta_v = 0$), then assign the vertex u (v) in set S. Otherwise, assign the vertex u (v) in set T. For each edge $e_i \in E$, if $\theta_{e_i} = 1$, then assign e_i in the set C. Observe that C is a max-cut in G of size k that divides V into two sets S and T. $\square$

5 MAXkSAT

In this section, we design a dNN for the MAXkSAT problem. First, we discuss a dNN for the MAX2SAT problem and then we generalize it for the MAXkSAT problem. Let Φ be an instance of the MAX2SAT problem. Furthermore, assume that Φ has m clauses over n variables. We construct a dNN f with trainable parameters $\theta \in [0, 1]^{n+m}$ with respect to Φ as follows: For each variable x_i, create a corresponding trainable parameters θ_{x_i}. For each clause c_i, there is a corresponding trainable parameter θ_{c_i}. The input to the dNN is an all-ones vector e_{n+m} which does not depend upon any data. The output of the dNN is $f(e_{n+m}; \theta) = f(\theta) \in \mathbb{R}$. Similar to the dNN structure of MAXCUT, the dNN for the MAX2SAT consists of four layers (see Figure 1 for the block diagram of the dNN structure). The dNN consists of one input layer, two hidden layers, and one output layer. The input layer e_{n+m} is connected with the first hidden layer through an elementwise product of the trainable parameters θ. The first hidden layer is connected to the second hidden layer by the binary matrix $W \in \{0,1\}^{2\cdot n\times(2\cdot n+m)}$. Here, $2 \cdot n$ is the total number of literals size for n variables. That means for a variable x_i, the binary matrix has two entries: one for literal x_i and another for literal $\bar{x}_i$. Note that the binary matrix is only dependent on Φ. At the second hidden layer, there exists a bias vector $b \in \{0, -\frac{1}{2}\}^{2\cdot m}$. There is a fully connected weight matrix $w \in \{-1, n\}^{2\cdot m}$ in the second hidden layer to the output layer. Note that all the parameters are defined as a function of Φ as follows:

$$f(e_{n+m};\theta) = f(\theta) = w^T\sigma(W^T(e_{n+m}\odot\theta)+b). \quad (5)$$

Note that we represent the value of the variable $\bar{x}_i$ as the complement of the neuron $\bar{\theta}_{x_i}$. That means, if θ_{x_i} is assigned as one then $\bar{\theta}_{x_i}$ represents zero. So, the function in (5) can be rewritten as follows:

$$\begin{aligned} f(\theta) = -\sum_{c_i\in\Phi}(\sigma(\theta_{c_i} - \frac{1}{2})) + n\cdot \\ \sum_{c_i=(\bar{x}_i\vee x_j)\in\Phi}(\sigma(\theta_{c_i} - (\bar{\theta}_{x_i} + \theta_{x_j}))). \end{aligned} \quad (6)$$

Here the clause $c_i = (\bar{x}_i \vee x_j)$ has taken arbitrarily to highlight the utilization of the negative literals. We prove the following theorem using the above dNN:

Theorem 2 *Let Φ be an instance of MAX2SAT with m clauses over n variables. Let f be the corresponding dNN of Φ. There exists an assignment for Φ that satisfies k clauses, if and only if the minimum value of f is $-\frac{k}{2}$.*

Proof: Let $\mathcal{A}$ be an assignment for Φ that satisfies k clauses. For each variable x_i that is assigned to **true**, set the corresponding $\theta_{x_i} = 1$. If it is assigned to **false**, then set $\theta_{x_i} = 0$. For each clause c_i that is satisfied, set the corresponding $\theta_{c_i} = 1$. If it is not satisfied, then set $\theta_{c_i} = 0$. Consider the output f represented in Figure 5 for an arbitrary clause $c_i = (x_i \vee x_j) \in \Phi$.

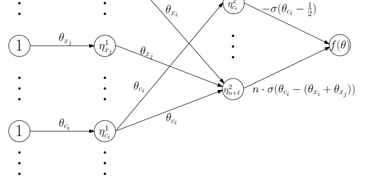


Figure 5: Output with respect to an arbitrary clause.

As per the construction of the dNN, the clause values denote the outputs of the preceding nodes in the network. Furthermore, the i^{th} neurons in the first hidden layer are denoted by η^1_i and the second hidden layer is denoted by η^2_i. If clause c_i is satisfied, then the value of both θ_{x_i} and θ_{x_j} cannot be 0. Moreover, $\theta_{c_i} = 1$. The above θ values of the corresponding clause c_i will contribute $-\frac{1}{2}$ to f. Furthermore, if c_i is not satisfied, then the values of θ_{x_i}, θ_{x_j}, and θ_{c_i} are 0. The above θ values of the corresponding clause c_i will contribute 0 to f. There are k clauses that are satisfied by assignment $\mathcal{A}$. Therefore, it will contribute $-\frac{k}{2}$ to f.

Conversely, assume that the minimum value of the output function f is $f(\theta) = -\frac{k}{2}$. We construct an assignment $\mathcal{A}$ for Φ that satisfies k clauses from f as follows: From the construction of the dNN, it is clear that, for each clause $c_i = (x_i \vee x_j)$, if $\theta_{c_i} = 1$, then $\theta_{c_i} - (\theta_{x_i} + \theta_{x_j}) \leq 0$. Otherwise, f does not achieve its minimum value. To prove this, assume that $\theta_{c_i} - (\theta_{x_i} + \theta_{x_j}) > 0$. It follows that the neuron $\eta^2_{n+\ell}$ contributes $n \cdot \sigma(\theta_{c_i} - (\theta_{x_i} + \theta_{x_j})) > 0$ to the output $f(\theta)$. This is a contradiction to the fact that f achieves its minimum value. We can simply assign the value of $\theta_{c_i} = 0$ and reduce the value of f further. So, it is clear that for any clause $c_i = (x_i \vee x_j)$, if $\theta_{c_i} = 1$, then $\theta_{c_i} - (\theta_{x_i} + \theta_{x_j}) \leq 0$. Each such clause, which θ value is one, contributes $-\frac{1}{2}$ to $f(\theta)$ through η^2_i. Furthermore, it contributes a value of 0 to $f(\theta)$ through $\eta^2_{n+\ell}$. That means there are k entries of value 1 in θ corresponding to clauses. For each $\theta_{c_i} = 1$, consider the θ_{x_i} and θ_{x_j} values. If $\theta_{x_i} = 0$ ($\theta_{x_j} = 0$), then assign the truth value of variable x_i (x_j) as **false**. Otherwise, assign the truth value of variable x_i (x_j) as **true**. Similarly, assign the truth values of the variables when $\theta_{c_i} = 0$. Observe that it will lead to an assignment $\mathcal{A}$ for Φ that satisfies k clauses. □

Corollary 1 *The above-discussed dNN for the MAX2SAT problem can be generalized for the MAXkSAT problem.*

Proof: Observe that the above-discussed dNN for MAX2SAT can capture MAXkSAT by changing the entries of the binary matrix W. For each clause in Φ, the corresponding column in W matrix has k entries as one instead of two in case of MAX2SAT. The values of the parameters θ, b, and w are the same in the dNN. In this case, $f(\theta)$ can be represented as follows:

$$\begin{aligned} f(\theta) = -\sum_{c_i\in\Phi}(\sigma(\theta_{c_i} - \frac{1}{2})) + n\cdot \\ \sum_{c_i=(x_1\vee\cdots\vee x_k)\in\Phi}(\sigma(\theta_{c_i} - (\theta_{x_1} + \cdots + \theta_{x_k}))). \end{aligned} \quad (7)$$

The construction and the proof for the MAX2SAT problem will follow for the MAXkSAT problem. □

6 MAXNAE2SAT

In this section, we design a dNN for the MAXNAE2SAT problem. Observe that the dNN for the MAXCUT problem captures the MAXNAE2SAT problem when all the literals of the formula Φ are positive. We design a dNN for the MAXNAE2SAT problem, where the problem instance Φ consists of both positive and negative liters.

Let Φ be an instance of the MAXNAE2SAT problem with m clauses over n variables. We construct a dNN f with trainable parameters $\theta \in [0,1]^{n+m}$ with respect to Φ as follows: For each variable x_i, there is a corresponding trainable parameter θ_{x_i}. For each clause c_i, there is a corresponding trainable parameter θ_{c_i}. The input to the dNN is an all-ones vector e_{n+m} which does not depend upon any data. The output of the dNN is $f(e_{n+m};\theta) = f(\theta) \in \mathbb{R}$. Similar to the above-discussed dNN structures, the dNN for the MAXNAE2SAT consists of one input layer, two hidden layers, and one output layer. The input layer e_{n+m} is connected with the first hidden layer through an elementwise product of the trainable parameters θ. The first hidden layer is connected to the second hidden layer by the binary matrix $W \in \{0,1\}^{2\cdot n\times(2\cdot n+m)}$. At the second hidden layer, there exists a bias vector $b \in \{0, -\frac{1}{2}\}^{2\cdot m}$. There is a fully connected weight matrix $w \in \{-1, n\}^{2\cdot m}$ in the second hidden layer to the output layer. Observe that all the parameters are defined as a function of Φ. The output of the dNN f corresponding to the MAXNAE2SAT problem is given by (5). The second hidden layer consists of the fixed matrix W and a bias vector b with a ReLU activation function $\sigma(x) = max(0, x)$. The last layer consists of the vector w.

Therefore, we can rewrite the function in (5) as follows:

$$f(\theta) = -\sum_{c_i\in\Phi}(\sigma(\theta_{c_i} - \frac{1}{2})) + n\cdot \sum_{c_i=(\bar{x}_i\vee x_j)\in\Phi} (\sigma(\theta_{c_i} - |\bar{\theta}_{x_i} - \theta_{x_j}|)). \tag{8}$$

We have used the clause $c_i = (\bar{x}_i \vee x_j)$ for showing the utilization of negative literals by the function and it is taken arbitrarily. With the above dNN construction, we prove the following theorem.

Theorem 3 *Let Φ be an instance of MAXNAE2SAT with m clauses over n variables. Let f be the corresponding dNN of Φ. There exists an assignment for Φ that nae-satisfies k clauses, if and only if the minimum value of f is $-\frac{k}{2}$.*

Proof: Let $\mathcal{A}$ be an assignment for Φ that nae-satisfies k clauses. For each variable x_i that is assigned to **true**, set the corresponding $\theta_{x_i} = 1$. If it is assigned to **false**, then set $\theta_{x_i} = 0$. For each clause c_i that is nae-satisfied, set the corresponding $\theta_{c_i} = 1$. If it is not nae-satisfied, then set $\theta_{c_i} = 0$. Consider the output f represented in Figure 6 for an arbitrary clause $c_i = (x_i \vee x_j) \in \Phi$.

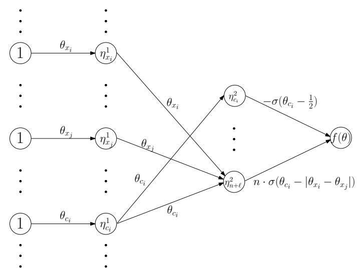


Figure 6: Output with respect to an arbitrary clause.

As per the construction of the dNN, the clause values denote the outputs of the preceding nodes in the network. Furthermore, the i^{th} neurons in the first hidden layer are denoted by η^1_i and the second hidden layer is denoted by η^2_i. If clause c_i is nae-satisfied, then the values of both θ_{x_i} and θ_{x_j} cannot be same. Moreover, $\theta_{c_i} = 1$. The above θ values of the corresponding clause c_i will contribute $-\frac{1}{2}$ to f. Furthermore, if c_i is not nae-satisfied, then the values of both θ_{x_i}, θ_{x_j} is either 0 or 1. However, the value of θ_{c_i} is 0. The above θ values of the corresponding clause c_i will contribute 0 to f. There are k clauses that are nae-satisfied by assignment $\mathcal{A}$. Therefore, it will contribute $-\frac{k}{2}$ to f.

Conversely, assume that the minimum value of the output function f is $f(\theta) = -\frac{k}{2}$. We construct an assignment $\mathcal{A}$ for Φ that nae-satisfies k clauses from f as follows: From the construction of the dNN, it is clear that, for each clause $c_i = (x_i \vee x_j)$, if $\theta_{c_i} = 1$, then $\theta_{c_i} - |\theta_{x_i} - \theta_{x_j}| \leq 0$. Otherwise, f does not achieve its minimum value. To prove this, assume that $\theta_{c_i} - |\theta_{x_i} - \theta_{x_j}| > 0$. It follows that the neuron $\eta^2_{n+\ell}$ contributes $n \cdot \sigma(\theta_{c_i} - |\theta_{x_i} - \theta_{x_j}|) > 0$ to the output $f(\theta)$. This is a contradiction to the fact that f achieves its minimum value. We can simply assign the value of $\theta_{c_i} = 0$ and reduce the value of f further. So, it is clear that for any clause $c_i = (x_i \vee x_j)$, if $\theta_{c_i} = 1$, then $\theta_{c_i} - |\theta_{x_i} - \theta_{x_j}| \leq 0$. Each such clause, which θ value is one, contributes $-\frac{1}{2}$ to $f(\theta)$ through η^2_i. Furthermore, it contributes a value of 0 to $f(\theta)$ through $\eta^2_{n+\ell}$. That means there are k entries of value 1 in θ corresponding to clauses. For each $\theta_{c_i} = 1$, consider the θ_{x_i} and θ_{x_j} values. If $\theta_{x_i} = 0$ ($\theta_{x_j} = 0$), then assign the truth value of variable x_i (x_j) as **false**. Otherwise, assign the truth value of variable x_i (x_j) as **true**. Similarly, assign the truth values of the variables when $\theta_{c_i} = 0$. Observe that it will lead to an assignment $\mathcal{A}$ for Φ that nae-satisfies k clauses. $\square$

References

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

Alkhouri, I. R.; Atia, G. K.; and Velasquez, A. 2022. A differentiable approach to the maximum independent set problem using dataless neural networks. *Neural Networks* 155:168–176.

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

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

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Drori, I.; Kharkar, A.; Sickinger, W. R.; Kates, B.; Ma, Q.; Ge, S.; Dolev, E.; Dietrich, B.; Williamson, D. P.; and Udell, M. 2020. Learning to solve combinatorial optimization problems on real-world graphs in linear time. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 19–24.

Festa, P. 2014. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In *2014 16th International Conference on Transparent Optical Networks (ICTON)*, 1–20.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Fomin, F. V., and Kratsch, D. 2010. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Gaspers, S. 2010. *Exponential Time Algorithms - Structures, Measures, and Bounds*. VDM.

Jena, S. K.; Subramani, K.; and Velasquez, A. 2023. Differentiable discrete optimization using dataless neural networks. The 16^{th} Annual International Conference on Combinatorial Optimization and Applications (COCOA), 2023 (accepted).

Lamm, S.; Sanders, P.; Schulz, C.; Strash, D.; and Werneck, R. F. 2016. Finding near-optimal independent sets at scale. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, 138–150.

Li, Z.; Chen, Q.; and Koltun, V. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems* 31.

Mazyavkina, N.; Sviridov, S.; Ivanov, S.; and Burnaev, E. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research* 134:105400.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

San Segundo, P.; Rodríguez-Losada, D.; and 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.; and Katzgraber, H. G. 2022. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence* 4(4):367–377.

Vazirani, V. 2002. *Approximation Algorithms*. Springer Science Publishers, 1^{st} edition.

Wilder, B.; Dilkina, B.; and Tambe, M. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1658–1665.

7 Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency through grant HR001123S0001-FP-004.