A Neural Combinatorial Optimization Algorithm for Unit Commitment in AC Power Systems

Shahab Bahrami, Yu Christine Chen, and Vincent W.S. Wong
Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
email: {bahramis, chen, vincentw}@ece.ubc.ca

***Abstract*—The unit commitment (UC) problem in AC power systems can be formulated as a mixed-integer nonlinear optimization program with a running time that scales exponentially with the number of generators. This paper addresses the time complexity of solving the UC problem by developing a deep learning framework that determines the generator on/off states using a transformer deep neural network (DNN), and subsequently solves an AC optimal power flow (OPF) problem to obtain the generator setpoints. To obtain a feasible binary solution, we apply a neural combinatorial optimization algorithm to train the DNN, while penalizing infeasible power flow solutions. Also, to guarantee the optimality of the generator setpoints, we transform the AC OPF problem into a semidefinite program (SDP). The proposed algorithm can obtain a near-optimal solution to the UC problem in polynomial running time. Simulations are performed for two IEEE test systems. When compared with three existing UC algorithms in the literature, our proposed algorithm can obtain a solution with at least 2.14% lower operation cost and lower running time. When compared with the MOSEK solver, our algorithm can obtain a solution with at most 1.97% greater operation cost, but with a significantly lower running time.**

I. Introduction

The unit commitment (UC) problem for AC power systems can be formulated as a mixed-integer optimization program with the objective of minimizing the cost of operating generators subject to operational constraints, including the network power balance, bus voltage limits, power flow limits of transmission lines, the generator capacity limits, ramp rate limits, and the minimum on/off time requirements. Obtaining an optimal solution to the UC problem while satisfying the network constraints is critical for economic and safe operations.

In practical settings, the UC problem is solved for large-scale transmission networks, and the network constraints are modeled with either DC or linear approximations of AC power flow equations to promote computational efficiency [1]. However, the key to guarantee a feasible solution that satisfies the network constraints is to avoid approximations of the power flow equations. While this can be achieved by the inclusion of the nonlinear AC power flow model in the UC problem, it then becomes a mixed-integer nonlinear program (MINLP) with nonconvex constraints and a large number of binary variables associated with the on/off states of all generators. These aspects render UC an NP-hard optimization problem [2], which is computationally difficult to solve. The running time required to obtain a near-optimal solution of the UC problem using nonlinear program (NLP) optimization solvers grows exponentially with the number of generators. Although the UC problem is typically solved on a day-ahead basis, an algorithm with lower running time can enable system operators to address faster and larger load variations and intermittent renewable energy resources by performing intra-day UC more often and maintain the consistency between the day-ahead and real-time electricity markets [3].

There have been many efforts to address the computational complexity of the UC problem. Amjady *et al.* in [4] formulated the UC problem with load and generation uncertainty as a three-level optimization problem. Šepetanc *et al.* in [5] applied a second-order Taylor approximation to formulate the UC problem as a mixed-integer quadratically constrained quadratic program (MIQCQP). Both [4] and [5] solved the resulting optimization problem using commercial MINLP solvers. Castillo *et al.* in [6] applied outer approximation method to decompose the UC problem into a mixed-integer linear program as the master problem and an NLP as the subproblem, and they solved the subproblem by using a successive linear programming technique. Liu *et al.* in [7] applied outer approximation method augmented with second-order cone relaxation and developed an iterative algorithm to solve the UC problem. Paredes *et al.* in [8] formulated the UC problem as an MIQCQP and solved the problem using successive semidefinite program (SDP) relaxation. Ashraphijuo *et al.* in [9] also proposed an SDP relaxation of the UC problem by including additional quadratic constraints and relaxing them to linear matrix inequalities. Quarm *et al.* in [10] proposed an SDP relaxation of the UC problem and developed a heuristic approach to obtain an suboptimal solution when the relaxation gap is not zero. Zohrizadeh *et al.* in [11] formulated the UC problem as a second-order cone program (SOCP) and developed a sequential convex relaxation algorithm to obtain a near-optimal solution. The general approach in the aforementioned works is to perform approximations on the UC problem, after which the approximate problem is solved by using either a commercial NLP solver (in, e.g., [4]–[6]) or an iterative approach comprising a sequence of convex optimization problems (in, e.g., [7]–[11]). These techniques, however, suffer from long running time to solve the approximate UC problem.

In this paper, the promising computational benefits of using deep neural networks (DNNs) to solve large-scale combinatorial optimization problems (in, e.g., [12], [13]) motivate us to address the computational complexity of the UC problem in AC power networks via a deep learning method that applies a neural combinatorial optimization algorithm to train a transformer DNN with sample UC problem instances for a

given power network. The trained DNN is augmented with an AC optimal power flow (OPF) module to obtain a near-optimal solution to a *new instance* of the UC problem. Hence, unlike solving an MINLP or a sequence of convex optimization problems (in, e.g., [4]–[11]), the trained DNN can reduce the running time to solve the UC problem by simply performing a forward propagation in the DNN and solving an OPF problem. The main contributions of this paper are as follows:

- *Deep Learning to Solve UC Problem*: We design a transformer DNN architecture that can obtain the binary variables for the set of operating generators. To train the proposed transformer DNN and promote the feasibility of the binary solution, we develop a neural combinatorial optimization algorithm [14], which penalizes infeasible solutions. By training the proposed DNN with sufficiently many sample UC problem instances using historical data for a given power network, we can obtain a feasible and near-optimal solution of a new instance of the UC problem in polynomial running time.
- *Full AC Power Flow Constraints*: To guarantee the feasibility of the generator setpoints, we consider full AC power flow equations in the UC problem. We address the nonconvexity of the full AC power flow constraints by applying convex relaxation techniques to transform the original UC problem into an SDP under a given set of generator on/off states. We can solve a single instance of SDP to obtain the setpoints of the operating generators.
- *Performance Evaluation*: Simulations are performed on the IEEE 6-bus and 300-bus test systems. For the IEEE 6-bus test system, results show that our proposed algorithm obtains the global optimal solution. When compared with existing state-of-the-art UC algorithms in the literature (i.e., [9]–[11]), our proposed algorithm obtains a solution with at least 2.14% lower objective value for the IEEE 300-bus test system. When compared with the MOSEK commercial MINLP solver with branch and bound method [15], our proposed algorithm can obtain a near-optimal solution with 1.97% higher objective value. However, in all comparison case studies, our proposed algorithm incurs a significantly lower running time to solve the UC problem.

The remainder of this paper is organized as follows. In Section II, we present the UC problem formulation and its transformation. In Section III, we develop a neural combinatorial optimization algorithm to solve the UC problem. In Section IV, we evaluate the performance of the proposed algorithm via simulations. Finally, we conclude the paper in Section V.

II. Unit Commitment Problem Formulation

Consider a power transmission network consisting of a set of buses $\mathcal{N}$ and a set of transmission lines $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$. Let $\mathcal{G} \subseteq \mathcal{N}$ denote the set of buses connected to the generators. We use the lumped-element Π model for transmission lines [1]. Let y_{nm} and $\overline{y}_{nm}$, respectively, denote the series and shunt admittances connected to bus n for line $(n,m) \in \mathcal{L}$. Further define $Y_{nm} = (\overline{y}_{nm}+y_{nm})e_n e_n^{\mathrm{T}} - y_{nm} e_n e_m^{\mathrm{T}}$ so that the entries (n,n) and (n,m) of Y_{nm} are equal to $\overline{y}_{nm}+y_{nm}$ and $-y_{nm}$, respectively, and all other entries of Y_{nm} are zero. Let Y denote the network admittance matrix. For bus $n \in \mathcal{N}$, let $e_n \in \mathbb{R}^N$ denote the n^{th} basis column vector and $Y_n = e_n e_n^{\mathrm{T}} Y$. For bus $n \in \mathcal{N}$, we define matrices

$$\mathbf{Y}_n = \frac{1}{2}\begin{bmatrix} \mathrm{Re}\{Y_n + Y_n^{\mathrm{T}}\} & \mathrm{Im}\{Y_n^{\mathrm{T}} - Y_n\} \\ \mathrm{Im}\{Y_n - Y_n^{\mathrm{T}}\} & \mathrm{Re}\{Y_n + Y_n^{\mathrm{T}}\} \end{bmatrix}, \tag{1a}$$

$$\overline{\mathbf{Y}}_n = -\frac{1}{2}\begin{bmatrix} \mathrm{Im}\{Y_n + Y_n^{\mathrm{T}}\} & \mathrm{Re}\{Y_n - Y_n^{\mathrm{T}}\} \\ \mathrm{Re}\{Y_n^{\mathrm{T}} - Y_n\} & \mathrm{Im}\{Y_n + Y_n^{\mathrm{T}}\} \end{bmatrix}, \tag{1b}$$

$$\mathbf{M}_n = \begin{bmatrix} e_n e_n^{\mathrm{T}} & 0 \\ 0 & e_n e_n^{\mathrm{T}} \end{bmatrix}. \tag{1c}$$

For each line $(n,m) \in \mathcal{L}$, we define matrices

$$\mathbf{Y}_{nm} = \frac{1}{2}\begin{bmatrix} \mathrm{Re}\{Y_{nm} + Y_{nm}^{\mathrm{T}}\} & \mathrm{Im}\{Y_{nm}^{\mathrm{T}} - Y_{nm}\} \\ \mathrm{Im}\{Y_{nm} - Y_{nm}^{\mathrm{T}}\} & \mathrm{Re}\{Y_{nm} + Y_{nm}^{\mathrm{T}}\} \end{bmatrix}, \tag{2a}$$

$$\overline{\mathbf{Y}}_{nm} = -\frac{1}{2}\begin{bmatrix} \mathrm{Im}\{Y_{nm} + Y_{nm}^{\mathrm{T}}\} & \mathrm{Re}\{Y_{nm} - Y_{nm}^{\mathrm{T}}\} \\ \mathrm{Re}\{Y_{nm}^{\mathrm{T}} - Y_{nm}\} & \mathrm{Im}\{Y_{nm} + Y_{nm}^{\mathrm{T}}\} \end{bmatrix}. \tag{2b}$$

Let $\mathcal{T} = \{1, \ldots, T\}$ denote the set of operating time horizon with T time slots of equal duration (e.g., one hour). In time slot $t \in \mathcal{T}$, let $V_n(t)$ denote the voltage phasor of bus n, and let $\boldsymbol{v}(t) = (V_n(t),\, n \in \mathcal{N})$ denote the vector of voltage phasors. We define vector $\boldsymbol{x}(t) = (\,(\mathrm{Re}\{\boldsymbol{v}(t)\})^{\mathrm{T}}\ (\mathrm{Im}\{\boldsymbol{v}(t)\})^{\mathrm{T}}\,)^{\mathrm{T}}$ consisting of the real and imaginary parts of $\boldsymbol{v}(t)$ in time slot t. We define the rank-one matrix $\mathbf{W}(t) = \boldsymbol{x}(t)\,\boldsymbol{x}(t)^{\mathrm{T}}$, i.e.,

$$\mathrm{rank}\{\mathbf{W}(t)\} = 1, \qquad t \in \mathcal{T}. \tag{3}$$

A. Operational Constraints

We denote the active- and reactive-power outputs of the generator at bus $n \in \mathcal{G}$ in time slot t by $P_{G_n}(t)$ and $Q_{G_n}(t)$, respectively. Let $P_{D_n}(t)$ and $Q_{D_n}(t)$ denote the active- and reactive- power components of the load at bus $n \in \mathcal{N}$ in time slot t, respectively. We denote the lower and upper limits of the voltage magnitude at bus n by $V_n^{\min}$ and $V_n^{\max}$, respectively. Let $S_{nm}^{\max}$ denote the upper limit for the apparent power flow in line $(n,m) \in \mathcal{L}$. We leverage the matrices defined in (1a)−(2b) to obtain the following constraints in time slot $t \in \mathcal{T}$ [16]:

$$P_{G_n}(t) - P_{D_n}(t) = \mathrm{Tr}\{\mathbf{Y}_n \mathbf{W}(t)\}, \qquad n \in \mathcal{G}, \tag{4a}$$

$$Q_{G_n}(t) - Q_{D_n}(t) = \mathrm{Tr}\{\overline{\mathbf{Y}}_n \mathbf{W}(t)\}, \qquad n \in \mathcal{G}, \tag{4b}$$

$$P_{D_n}(t) = -\mathrm{Tr}\{\mathbf{Y}_n \mathbf{W}(t)\}, \qquad n \in \mathcal{N} \setminus \mathcal{G}, \tag{4c}$$

$$Q_{D_n}(t) = -\mathrm{Tr}\{\overline{\mathbf{Y}}_n \mathbf{W}(t)\}, \qquad n \in \mathcal{N} \setminus \mathcal{G}, \tag{4d}$$

$$(V_n^{\min})^2 \le \mathrm{Tr}\{\mathbf{M}_n \mathbf{W}(t)\} \le (V_n^{\max})^2, \qquad n \in \mathcal{N}, \tag{4e}$$

$$\begin{bmatrix} (S_{nm}^{\max})^2 & \mathrm{Tr}\{\mathbf{Y}_{nm}\mathbf{W}(t)\} & \mathrm{Tr}\{\overline{\mathbf{Y}}_{nm}\mathbf{W}(t)\} \\ \mathrm{Tr}\{\mathbf{Y}_{nm}\mathbf{W}(t)\} & 1 & 0 \\ \mathrm{Tr}\{\overline{\mathbf{Y}}_{nm}\mathbf{W}(t)\} & 0 & 1 \end{bmatrix} \succeq 0, \quad (n,m) \in \mathcal{L}. \tag{4f}$$

Above constraints, (4a)−(4d) represent nodal power balance equations, constraint (4e) delineates the limits on the voltage magnitude of bus n, and constraint (4f) represents the limit on the apparent power flow in line (n,m). In time slot t, let binary variable $u_n(t) \in \{0,\, 1\}$ indicate whether the generator at bus n is on $(u_n(t) = 1)$ or off $(u_n(t) = 0)$. Also, let binary

variable $s_n(t) \in \{0, 1\}$ indicate whether the generator at bus n starts up ($s_n(t) = 1$) or not ($s_n(t) = 0$). Similarly, binary variable $d_n(t) \in \{0, 1\}$ indicates whether the generator at bus n shuts down ($d_n(t) = 1$) or not ($d_n(t) = 0$). Thus, we have $s_n(t) = d_n(t) = 0$ if generator n neither starts up nor shuts down in time slot t. Let $P_{G_n}^{\min}$ and $P_{G_n}^{\max}$, respectively, denote the lower and upper limits for the active-power output of the generator at bus n. Let $Q_{G_n}^{\min}$ and $Q_{G_n}^{\max}$, respectively, denote the lower and upper limits for the reactive-power output of the generator at bus n. For generator at bus n, we denote the maximum ramp-up, ramp-down, startup, and shutdown ramp rate r_n^{u}, r_n^{d}, r_n^{su}, and r_n^{sd}, respectively. We consider a minimum up time t_n^{u} and down time t_n^{d} for the generator at bus n. In time slot $t \in \mathcal{T}$, for the generator at bus $n \in \mathcal{G}$, we have

$$u_n(t)P_{G_n}^{\min} \le P_{G_n}(t) \le u_n(t)P_{G_n}^{\max}, \tag{5a}$$
$$u_n(t)Q_{G_n}^{\min} \le Q_{G_n}(t) \le u_n(t)Q_{G_n}^{\max}, \tag{5b}$$
$$P_{G_n}(t) - P_{G_n}(t-1) \le u_n(t-1)\, r_n^{\mathrm{u}} + s_n(t)\, r_n^{\mathrm{su}}, \tag{5c}$$
$$P_{G_n}(t-1) - P_{G_n}(t) \le u_n(t-1)\, r_n^{\mathrm{d}} + d_n(t)\, r_n^{\mathrm{sd}}, \tag{5d}$$
$$\textstyle\sum_{t'=t-t_n^{\mathrm{u}}+1}^{t} s_n(t') \le u_n(t), \tag{5e}$$
$$\textstyle\sum_{t'=t-t_n^{\mathrm{d}}+1}^{t} d_n(t') \le 1 - u_n(t). \tag{5f}$$

The variable $s_n(t)$ is equal to 1 only when the generator at bus n is off in time slot $t-1$ but is on in time slot t. Similarly, the variable $d_n(t)$ is equal to 1 only when the generator at bus n is on in time slot $t-1$ but is off in time slot t. Thus, we have

$$u_n(t) - u_n(t-1) = s_n(t) - d_n(t), \quad n \in \mathcal{G},\, t \in \mathcal{T}. \tag{6}$$

It can be shown that constraints (5e), (5f), and (6) enforce $s_n(t)$ and $d_n(t)$ to be either 0 or 1 even if we *relax* them to take any value in the interval $[0, 1]$. Then, we have

$$0 \le s_n(t),\, d_n(t) \le 1, \qquad n \in \mathcal{G},\, t \in \mathcal{T}, \tag{7}$$
$$u_n(t) \in \{0, 1\}, \qquad n \in \mathcal{G},\, t \in \mathcal{T}. \tag{8}$$

B. The UC Problem in AC Power Systems

The operation cost of a generator includes those for generation, startup, and shutdown [1]. The generation cost of a generator at bus n with output $P_{G_n}(t)$ in time slot t can be modeled by a quadratic function $c_{n2}\,(P_{G_n}(t))^2 + c_{n1}P_{G_n}(t) + c_{n0}\,u_n(t)$, where c_{n0}, c_{n1}, and c_{n2} are nonnegative coefficients. We consider a fixed startup cost c_n^{su} and a fixed shutdown cost c_n^{sd} for the generator at bus n. We define $\boldsymbol{P}_G(t) = (P_{G_n}(t),\, n \in \mathcal{G})$, $\boldsymbol{u}(t) = (u_n(t),\, n \in \mathcal{G})$, $\boldsymbol{s}(t) = (s_n(t),\, n \in \mathcal{G})$, and $\boldsymbol{d}(t) = (d_n(t),\, n \in \mathcal{G})$ in time slot t. Then, the aggregate operation cost of all generators in time slot t can be expressed as follows:

$$C^{\mathrm{op}}\big(\boldsymbol{P}_G(t),\, \boldsymbol{u}(t),\, \boldsymbol{s}(t),\, \boldsymbol{d}(t)\big) = \textstyle\sum_{n\in\mathcal{G}}\Big(c_{n2}\,(P_{G_n}(t))^2 + c_{n1}P_{G_n}(t) + c_{n0}\,u_n(t) + c_n^{\mathrm{su}}\,s_n(t) + c_n^{\mathrm{sd}}\,d_n(t)\Big). \tag{9}$$

Let $\boldsymbol{Q}_G(t) = (Q_{G_n}(t),\, n \in \mathcal{G})$ collect the reactive-power outputs of all generators in time slot t. For the UC problem over the operating horizon $\mathcal{T}$, we define the vector of decision variables $\boldsymbol{\phi} = (\mathbf{W}(t),\, \boldsymbol{P}_G(t),\, \boldsymbol{Q}_G(t),\, \boldsymbol{s}(t),\, \boldsymbol{d}(t),\, \boldsymbol{u}(t),\, t \in \mathcal{T})$. Let Φ denote the feasible space defined by constraints (3)−(8). The system operator aims to minimize the system-wide operation cost. The UC problem is formulated as

$$\boldsymbol{\mathcal{P}}_1: \quad \underset{\boldsymbol{\phi}}{\text{minimize}} \;\; C(\boldsymbol{\phi}) \quad \text{subject to } \boldsymbol{\phi} \in \Phi,$$

where $C(\boldsymbol{\phi}) = \sum_{t\in\mathcal{T}} C^{\mathrm{op}}\big(\boldsymbol{P}_G(t), \boldsymbol{u}(t), \boldsymbol{s}(t), \boldsymbol{d}(t)\big)$. Due to the quadratic objective function, the rank-one constraint (3), and the binary variables $\boldsymbol{u}(t),\, t \in \mathcal{T}$, problem $\boldsymbol{\mathcal{P}}_1$ is an MINLP, which is NP-hard [2] and difficult to solve. Thus, we solve problem $\boldsymbol{\mathcal{P}}_1$ in two steps: (i) transform it into a combinatorial optimization problem, and (ii) develop a neural combinatorial optimization algorithm that trains a transformer DNN to determine a near-optimal solution for the UC problem.

C. Transform $\boldsymbol{\mathcal{P}}_1$ into Combinatorial Optimization Problem

Let $\boldsymbol{\psi} = (\mathbf{W}(t),\, \boldsymbol{P}_G(t),\, \boldsymbol{Q}_G(t),\, \boldsymbol{s}(t),\, \boldsymbol{d}(t),\, t \in \mathcal{T})$ denote the vector of continuous decision variables. Let $\Psi_{\boldsymbol{u}}$ denote the feasible space defined by constraints (3)−(7) under a given vector $\boldsymbol{u} = (\boldsymbol{u}(t),\, t \in \mathcal{T})$. The objective function in problem $\boldsymbol{\mathcal{P}}_1$ can be decomposed as follows:

$$C(\boldsymbol{\phi}) = \widetilde{C}(\boldsymbol{\psi}) + \textstyle\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{G}}\, c_{n0}\,u_n(t), \tag{10}$$

where $\widetilde{C}(\boldsymbol{\psi}) = \sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{G}}\big(c_{n2}\,(P_{G_n}(t))^2 + c_{n1}P_{G_n}(t) + c_n^{\mathrm{su}}\,s_n(t) + c_n^{\mathrm{sd}}\,d_n(t)\big)$. Problem $\boldsymbol{\mathcal{P}}_1$ is equivalent to the following optimization problem:

$$\boldsymbol{\mathcal{P}}_2: \quad \underset{\boldsymbol{u}}{\text{minimize}} \;\; \widetilde{C}(\boldsymbol{\psi}_{\boldsymbol{u}}^*) + \textstyle\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{G}}\, c_{n0}\,u_n(t) \quad \text{subject to } \; u_n(t) \in \{0, 1\},\; n \in \mathcal{G},\, t \in \mathcal{T},$$

where $\boldsymbol{\psi}_{\boldsymbol{u}}^*$ is the solution to the following optimization problem under a given vector $\boldsymbol{u}$:

$$\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\mathrm{opf}}: \quad \underset{\boldsymbol{\psi}}{\text{minimize}} \;\; \widetilde{C}(\boldsymbol{\psi}) \quad \text{subject to } \; \boldsymbol{\psi} \in \Psi_{\boldsymbol{u}}.$$

Problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\mathrm{opf}}$ is an AC OPF for a given $\boldsymbol{u}$. The UC problem $\boldsymbol{\mathcal{P}}_1$ is equivalent to problem $\boldsymbol{\mathcal{P}}_2$. Considering vector $\boldsymbol{u}$ as the decision variable, problem $\boldsymbol{\mathcal{P}}_2$ is a combinatorial optimization problem. To solve the original UC problem $\boldsymbol{\mathcal{P}}_1$, it is sufficient to obtain the optimal solution $\boldsymbol{u}^*$ to the combinatorial optimization problem $\boldsymbol{\mathcal{P}}_2$, such that $\boldsymbol{\psi}_{\boldsymbol{u}^*}^*$ is the optimal solution to the AC OPF problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\mathrm{opf}}$ for $\boldsymbol{u} = \boldsymbol{u}^*$.

III. Algorithm Design

In this section, we leverage the promising accuracy and computational benefits of the neural combinatorial optimization algorithm with transformer DNN [12], [13] to solve combinatorial optimization problem $\boldsymbol{\mathcal{P}}_2$. The transformer DNN can determine $\boldsymbol{u}^*$, i.e., the set of operating generators. The AC OPF module then solves $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}^*}^{\mathrm{opf}}$ to obtain the optimal setpoints of the operating generators. We train the proposed DNN using sample UC problem instances for a particular power network. The trained DNN can obtain a near-optimal solution to a new instance of the UC problem.

A. Transformer DNN Architecture

The number of binary variables in problem $\boldsymbol{\mathcal{P}}_2$ is $T|\mathcal{G}|$. A standard transformer architecture (as in [12] and [13]) has an encoder-decoder neural network that requires an input sequence with $2T|\mathcal{G}|$ elements corresponding to the on and off state of the generators in all time slots. Such a DNN would have a large number of neural network parameters for the UC problem in a large-scale power system over a typical operating time horizon of $T = 24$ hours. Hence, training the DNN would require a large amount of computational and storage resources, which may not be available in practice. To address this challenge, we modify the standard transformer DNN architecture to solve problem $\boldsymbol{\mathcal{P}}_2$ in T consecutive steps. As shown in Fig. 1, in the modified DNN, we reduce the number of elements of the input sequence from $2T|\mathcal{G}|$ to $2|\mathcal{G}|$ corresponding to the on and off state of the generators in time slot t. The modified DNN determines the set of operating generators in time slot t. Thus, the output sequence contains $|\mathcal{G}|$ elements corresponding to vector $\boldsymbol{u}(t)$ in time slot t. Moreover, the embedding for the obtained set of operating generators in the previous time slot $t-1$ is included in the context embedding vector to be shared with the decoder self-attention layer as well as the encoder-decoder attention layer in time slot t. The shared information enables the decoder to account for the generators' status in time slot $t-1$ when scheduling the generators in time slot t, which guarantees that the inter-temporal constraints (5e)−(6) are satisfied. Let $\boldsymbol{\theta}$ denote the vector of neural network parameters. The total number of neural network parameters in the modified DNN architecture is independent of T. With the proposed transformer DNN, we can obtain the set of operating generators over the entire time horizon $\mathcal{T}$. We append an AC OPF module to solve optimization problem $\boldsymbol{\mathcal{P}}^{\text{opf}}_{2,\boldsymbol{u}}$ to obtain the generator setpoints.

B. Neural Combinatorial Optimization Algorithm

Algorithm 1 shows the proposed neural combinatorial optimization algorithm to solve UC problem $\boldsymbol{\mathcal{P}}_2$.

1) Sample Datasets and Algorithm Initialization: For a given power system, we use the sets $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{eval}}$ of sample UC problem instances to train and evaluate the proposed DNN, respectively. A sample problem instance comprises of historical data for the load profiles and generator specifications at different buses. Lines 1 to 3 of Algorithm 1 correspond to the algorithm initialization. Let I denote the number of training epochs. An epoch consists of B batches of E^{train} problem instances selected randomly from set $\mathcal{D}^{\text{train}}$. We denote the set of batch indices by $\mathcal{B} = \{1, \ldots, B\}$. We evaluate the updated DNN at the end of each epoch. For evaluation, we consider E^{eval} problem instances from set $\mathcal{D}^{\text{eval}}$. We initialize the number of training epochs I, number of batches per epoch B, training epoch size E^{train}, and evaluation size E^{eval}. Each update of the neural network parameter is referred to as a step. We set both the epoch index i and step index j to 1 and initialize the neural network parameter $\boldsymbol{\theta}_j$ in step $j = 1$.

2) Algorithm Training and Evaluation: The loop within Lines 4 and 20 involves the training and evaluation phases of

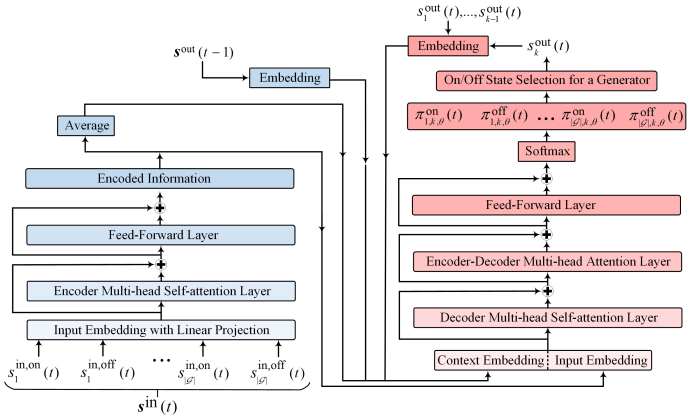

Figure 1. A modified transformer DNN for the UC problem.

Algorithm 1: Deep Learning-based UC Algorithm.

1 Initialize the number of training epochs I, number of batches per epoch B, training epoch size E^{train}, and evaluation size E^{eval}.
2 Initialize epoch index $i := 1$. Set step index to $j := 1$.
3 Initialize neural network parameter $\boldsymbol{\theta}_1$ randomly.
4 **while** $i \leq I$ **do**
5 Select set $\mathcal{D}_i^{\text{train}} \subseteq \mathcal{D}^{\text{train}}$ of E^{train} sample data randomly.
6 Select set $\mathcal{D}_i^{\text{eval}} \subseteq \mathcal{D}^{\text{eval}}$ of E^{eval} sample data randomly.
7 Divide set $\mathcal{D}_i^{\text{train}}$ into B batches $\mathcal{D}_{i,b}^{\text{train}}$, $b \in \mathcal{B}$.
8 Set batch index $b := 1$.
9 **while** $b \leq B$ **do**
10 Solve the relaxed UC problem $\boldsymbol{\mathcal{P}}_d^{\text{r}}$ to obtain input sequence $\boldsymbol{s}_d^{\text{in}}(t) := (s_{n,d}^{\text{in,on}}(t),\, s_{n,d}^{\text{in,off}}(t),\, n \in \mathcal{G})$ for sample data $d \in \mathcal{D}_{i,b}^{\text{train}}$ and time slot $t \in \mathcal{T}$.
11 Obtain output sequence $\boldsymbol{s}_d^{\text{out}}(t) := (s_{k,d}^{\text{out}}(t), k = 1, \ldots, |\mathcal{G}|)$ for sample data $d \in \mathcal{D}_{i,b}^{\text{train}}$ and time slot $t \in \mathcal{T}$.
12 Compute vector $\boldsymbol{u}_d$ as the on/off state of the generators for sample data $d \in \mathcal{D}_{i,b}^{\text{train}}$ from $\boldsymbol{s}_d^{\text{out}} := (\boldsymbol{s}_d^{\text{out}}(t),\, t \in \mathcal{T})$.
13 Obtain $\boldsymbol{\psi}_{\boldsymbol{u}_d}^{\text{p},*}$ by solving the penalized AC OPF problem $\boldsymbol{\mathcal{P}}^{\text{opf–p}}_{2,\boldsymbol{u}}$, $\boldsymbol{u} := \boldsymbol{u}_d$, for sample data $d \in \mathcal{D}_{i,b}^{\text{train}}$.
14 Update neural network parameter $\boldsymbol{\theta}_j$ in step j using (11).
15 Update batch index $b := b + 1$.
16 Update step index $j := j + 1$.
17 **end**
18 Evaluate the updated model on the evaluation set $\mathcal{D}_i^{\text{eval}}$.
19 Update epoch index $i := i + 1$.
20 **end**

Algorithm 1. In Lines 5 and 6, we randomly sample $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{eval}}$ to obtain, respectively, the training set $\mathcal{D}_i^{\text{train}}$ and evaluation set $\mathcal{D}_i^{\text{eval}}$ of sample UC problem instances in epoch i. In Line 7, we partition the training set into B batches, denoted by $\mathcal{D}_{i,b}^{\text{train}}$, $b \in \mathcal{B}$. In the loop within Lines 9 and 17, we repeatedly compute the input sequence, determine the corresponding output sequence, and update the neural network parameters. We use subscript d in parameters and variables associated with sample UC problem instance $d \in \mathcal{D}_{i,b}^{\text{train}}$.

Constructing an appropriate input sequence for a sample UC problem instance is crucial to train the proposed DNN. We relax constraints (3) and (8) in UC problem $\boldsymbol{\mathcal{P}}_1$ and replace them by $\mathbf{W}(t) \succeq 0$ and $0 \leq u_n(t) \leq 1$, $n \in \mathcal{G}$, $t \in \mathcal{T}$, respectively, to obtain the feasible space Φ^{r}. We solve the following relaxed

UC for the sample problem instance $d \in \mathcal{D}_{i,b}^{\text{train}}$:

$$\boldsymbol{\mathcal{P}}_d^{\text{r}}: \quad \underset{\boldsymbol{\phi}}{\text{minimize}} \quad C(\boldsymbol{\phi}) \\ \text{subject to} \quad \boldsymbol{\phi} \in \Phi^{\text{r}}.$$

The relaxed UC problem $\boldsymbol{\mathcal{P}}_d^{\text{r}}$ is an SDP and can be solved efficiently. The relaxed UC problem $\boldsymbol{\mathcal{P}}_d^{\text{r}}$ provides a lower bound for the original UC problem $\mathcal{P}_{2,d}$ for sample problem instance d. Also, the optimal solution $\boldsymbol{u}_d^{\text{r,opt}}(t) = (u_{n,d}^{\text{r,opt}}(t),\, n \in \mathcal{G})$ to problem $\boldsymbol{\mathcal{P}}_d^{\text{r}}$ provides implicit information about the on/off states of the generators over the entire operation horizon. Hence, $\boldsymbol{u}_d^{\text{r,opt}}(t)$ is a suitable choice to construct the input sequence $\boldsymbol{s}_d^{\text{in}}(t)$, $t \in \mathcal{T}$, for sample problem instance $d \in \mathcal{D}_{i,b}^{\text{train}}$. In Line 10, we obtain the input sequence $\boldsymbol{s}_d^{\text{in}}(t)$, $t \in \mathcal{T}$, for a sample problem instance $d \in \mathcal{D}_{i,b}^{\text{train}}$. We set $s_{n,d}^{\text{in,on}}(t) = u_{n,d}^{\text{r,opt}}(t)$ and $s_{n,d}^{\text{in,off}}(t) = 1 - u_{n,d}^{\text{r,opt}}(t)$ for generator $n \in \mathcal{G}$. Thus, the input sequence for time slot $t \in \mathcal{T}$ and problem instance $d \in \mathcal{D}_{i,b}^{\text{train}}$ is obtained as $\boldsymbol{s}_d^{\text{in}}(t) = \big(s_{n,d}^{\text{in,on}}(t),\, s_{n,d}^{\text{in,off}}(t),\, n \in \mathcal{G}\big)$.

In Fig. 1, consider input sequence $\boldsymbol{s}_d^{\text{in}}(t)$ for sample problem instance d. The input sequence is passed through a linear projection layer to construct the input embedding. Then it passes through an attention layer and a feed-forward layer to obtain the inter-dependent operation of the generators in the power network and compute the encoded information (referred to as the node embedding [12, Sec. 3.1]). The decoding process in the decoder is performed for $|\mathcal{G}|$ rounds to obtain the on/off state of all generators in a certain time slot t. In round $k = 1, \ldots, |\mathcal{G}|$, the input embedding of the decoder is the encoded information, which contains implicit information about the generators' schedules from the solution to the relaxed UC problem $\boldsymbol{\mathcal{P}}_d^{\text{r}}$. The decoder also receives the context embedding that contains the average of encoded information, the embedding for the output sequence in previous time slot $t-1$, and the embedding of the obtained set of operating generators up to round $k-1$. Such a context embedding enables the decoder to apply masking mechanism [12, Sec. 3.2] with information from the solution to the relaxed UC problem $\boldsymbol{\mathcal{P}}_d^{\text{r}}$, the inter-temporal constraints (5e), (5f), and (6), and the power flow constraints. In Lines 11 and 12 of Algorithm 1, we obtain output sequence $\boldsymbol{s}_d^{\text{out}}(t) = (s_{k,d}^{\text{out}}(t),\, k = 1, \ldots, |\mathcal{G}|)$ and vector $\boldsymbol{u}_d$ for on/off state of the generators in all time slots. Next, we solve AC OPF problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$ for $\boldsymbol{u} = \boldsymbol{u}_d$ to obtain the setpoints of the operating generators. Problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$ may not have a feasible solution for $\boldsymbol{u} = \boldsymbol{u}_d$. We penalize infeasible solutions during DNN training. We transform problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$ into a penalized AC OPF problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf-p}}$ in three steps. First, we introduce a new decision variable $\Delta u_n(t)$ for the generator at bus $n \in \mathcal{G}$. In constraints (5a)–(6), we replace $u_n(t-1)$ and $u_n(t)$ by $u_n(t-1) + \Delta u_n(t-1)$ and $u_n(t) + \Delta u_n(t)$, respectively. We define $\boldsymbol{\psi}^{\text{p}} = (\boldsymbol{\psi},\, \Delta u_n(t),\, n \in \mathcal{G},\, t \in \mathcal{T})$. Second, we relax constraint (3) and replace it by $\mathbf{W}(t) \succeq 0$, $t \in \mathcal{T}$ to obtain the feasible space $\Psi_{\boldsymbol{u}}^{\text{p}}$. Third, we define a modified objective function $\widetilde{C}^{\text{p}}(\boldsymbol{\psi}^{\text{p}})$ by including a penalty term $\rho \sum_{t\in\mathcal{T}} \sum_{n\in\mathcal{G}} \big|\Delta u_n(t)\big|$ to the objective function of problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$, where ρ is a nonnegative weight coefficient.

We have $\widetilde{C}^{\text{p}}(\boldsymbol{\psi}^{\text{p}}) = \widetilde{C}(\boldsymbol{\psi}) + \rho \sum_{t\in\mathcal{T}} \sum_{n\in\mathcal{G}} \big|\Delta u_n(t)\big|$. Problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$ is transformed into the following optimization problem:

$$\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf–p}}: \quad \underset{\boldsymbol{\psi}^{\text{p}}}{\text{minimize}} \quad \widetilde{C}^{\text{p}}(\boldsymbol{\psi}^{\text{p}}) \\ \text{subject to} \quad \boldsymbol{\psi}^{\text{p}} \in \Psi_{\boldsymbol{u}}^{\text{p}}.$$

Problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf–p}}$ always has a feasible solution. Via algebraic manipulations, problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf–p}}$ can be transformed into an SDP that can be solved in polynomial time. With quadratic cost function, practical power networks (including IEEE test systems) satisfy the sufficient conditions given in [16, Sec. IV-C] for the network topology and constraints (e.g., connected graph induced by $\text{Re}\{Y\}$ and nonnegative Lagrange multipliers for the active power balance constraints). Thus, if the original AC OPF problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$ is feasible for $\boldsymbol{u} = \boldsymbol{u}_d$, then by increasing the weight coefficient ρ, the global optimal solution $\boldsymbol{\psi}_{\boldsymbol{u}_d}^{\text{p},*}$ of problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf–p}}$ approaches the global optimal solution to problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf}}$. In Line 13 of Algorithm 1, we obtain the optimal solution $\boldsymbol{\psi}_{\boldsymbol{u}_d}^{\text{p},*}$ to problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf–p}}$. The objective value is obtained as $\widetilde{C}^{\text{p}}(\boldsymbol{\psi}_{\boldsymbol{u}_d}^{\text{p},*}) + \sum_{t\in\mathcal{T}} \sum_{n\in\mathcal{G}} c_{n0}\, u_{n,d}(t)$. We update the neural network parameter by batch gradient descent using the REINFORCE gradient estimator [14]. Define $\boldsymbol{\pi}_{\boldsymbol{\theta}_j} = (\pi_{n,k,\boldsymbol{\theta}_j,d}^{\text{on}}(t),\, \pi_{n,k,\boldsymbol{\theta}_j,d}^{\text{off}}(t),\, n \in \mathcal{G},\, k = 1, \ldots, |\mathcal{G}|,\, d \in \mathcal{D}_{i,b}^{\text{train}},\, t \in \mathcal{T})$. In Line 14 of Algorithm 1, the neural network parameter is updated as follows:

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \gamma_j \Big(\widetilde{C}^{\text{p}}(\boldsymbol{\psi}_{\boldsymbol{u}_d}^{\text{p},*}) + \sum_{t\in\mathcal{T}} \sum_{n\in\mathcal{G}} c_{n0}\, u_{n,d}(t)\Big) \nabla_{\boldsymbol{\theta}} \ln \boldsymbol{\pi}_{\boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_j}, \tag{11}$$

where γ_j is the learning rate in step j. In (11), one can include a baseline as the cost of a deterministic greedy algorithm solution obtained by the best model until epoch i. Adding the baseline function can reduce the gradient variance and increase the speed of learning. It also enables the DNN to gradually improve over itself [13]. In Lines 15 and 16, we update the batch index and step index. Epoch i is completed when all batches of sample problem instances have been processed. In Line 18, we evaluate the obtained DNN model in epoch i on the evaluation set $\mathcal{D}_i^{\text{eval}}$. In Line 19, we update the epoch index.

3) Solve UC Problem Using the Trained DNN: After I epochs, we select the model with the lowest objective value for the evaluation set as the trained DNN to solve $\mathcal{P}_2$ for a new UC problem instance. The total running time to solve a problem instance includes the time to (i) solve the relaxed UC problem, (ii) perform forward propagation in the trained DNN, and (iii) solve the AC OPF problem $\boldsymbol{\mathcal{P}}_{2,\boldsymbol{u}}^{\text{opf–p}}$. The relaxed UC and AC OPF problems are SDPs and can be solved in polynomial time. Also, the forward propagation in the trained DNN has a constant running time. Thus, by using the trained DNN, a new UC problem instance can be solved in polynomial time.

IV. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm in solving the UC problem for the IEEE 6-bus and 300-bus test systems. The data for the test systems are

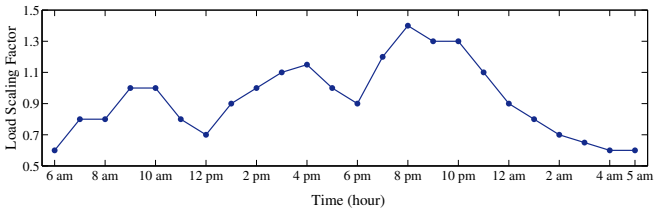

Figure 2. Load scaling factor during 24 hours.

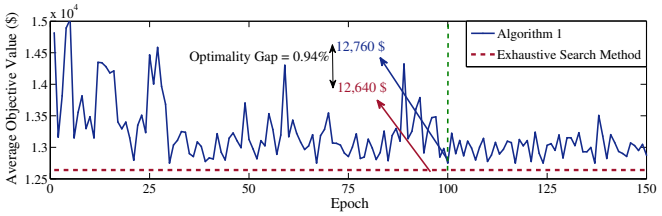

Figure 3. Average objective value versus the number of epochs for Algorithm 1 and the exhaustive search method.

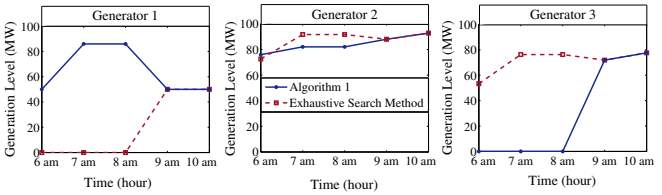

Figure 4. The output active power of the generators in an IEEE 6-bus test system obtained using Algorithm 1 and the exhaustive search method.

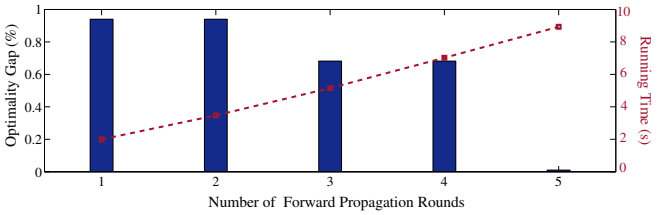

Figure 5. The optimality gap between Algorithm 1 and the exhaustive search method, as well as the running time versus the number of rounds that forward propagation is performed to select the solution with the lowest objective value.

from [17]. The data in [17] for active- and reactive-power loads are for a single time slot. Hence, for each bus, we obtain the average load profile over the operating horizon $\mathcal{T}$ by scaling the loads given in [17]. Fig. 2 depicts the load scaling factor that we use over a 24-hour period. We then use the average load profile at each bus to obtain the data samples for the active-power load during one day. That is, we scale the historical load demand data from Ontario, Canada power grid database [18] from January 1, 2020 to December 31, 2020, such that the average value in a time slot is equal to the average active-power load for a bus. We consider a fixed power factor to obtain the samples for reactive loads. We use the generation cost coefficients c_{n0}, c_{n1}, and c_{n2}, $n \in \mathcal{G}$ given in [17]. For generator $n \in \mathcal{G}$, we set the coefficients c_n^{su} and c_n^{sd} at random from a normal distribution with mean c_{n1} and variance $0.1 \times c_{n1}$. We set the parameters $P_{G_n}^{\min}$, $P_{G_n}^{\max}$, $Q_{G_n}^{\min}$, and $Q_{G_n}^{\max}$ according to [17] for generator n. We set parameters r_n^{u}, r_n^{d}, r_n^{su}, and r_n^{sd} to 50% of $P_{G_n}^{\max}$ for generator n. Unless stated otherwise, we select parameters t_n^{u} and t_n^{d} at random from set $\{0, 1, 2\}$ for generator n. For the training process in Algorithm 1, we consider $I = 150$ epochs and 4800 problem instances (i.e., $E^{\text{train}} = 32$ problem instances per epoch). The number of batches per epoch is set to 4. For validation, we consider $E^{\text{eval}} = 32$ problem instances. We perform simulations using Python/PyTorch and Python/CVXPY with MOSEK solver [15] on the Digital Research Alliance of Canada platform [19] with 24 CPUs and 4 GPUs.

First, we compare the gap between the solution obtained by Algorithm 1 and the global optimal solution of the original UC problem $\boldsymbol{\mathcal{P}}_1$ obtained by the exhaustive search method in an IEEE 6-bus test system with three generators. We set the operating time horizon $\mathcal{T}$ to 5 hours to limit the number of combinations for the on/off states of the generators to 2^{15} for the exhaustive search method. We set parameters t_n^{u} and t_n^{d} to 1 hour for all generators. Fig. 3 shows the convergence of the average objective value with Algorithm 1. The horizontal dashed line shows the average objective value with the exhaustive search method. The running time of the exhaustive search method to solve one problem instance is about one hour. The average optimality gap between Algorithm 1 and the exhaustive search method is 0.94%. As an example, Fig. 4 shows the generator setpoints for one validation sample using Algorithm 1 and the exhaustive search method. For this UC problem instance, the objective value with the trained DNN of Algorithm 1 is $\$12,780$, which is only 0.7% larger than the optimal value, $\$12,690$, obtained by the exhaustive search method. To reduce the optimality gap between Algorithm 1 and the exhaustive search method, we can run forward propagation for multiple rounds and select the solution with the smallest objective value. To justify this approach, in Fig. 5, we show the average optimality gap between Algorithm 1 and the exhaustive search method versus the number of rounds that we run forward propagation. Results demonstrate that we can obtain the global optimal solution if we run forward propagation for five rounds, with the running time being less than 10 seconds, which is significantly lower than the running time of the exhaustive search method.

Due to the high computational complexity, the exhaustive search method becomes impractical for an IEEE 300-bus test system with an operating time horizon of $T = 24$ hours. For the sake of comparison, we instead use MOSEK commercial solver [15], which applies a branch and bound method to solve the original UC problem as an MINLP. The NLP commercial solvers such as MOSEK are commonly used in the literature (in, e.g., [4]–[6]) to solve the UC problem. In MOSEK, we use the default value of 10^{-6} for the relative optimality tolerance parameter [15]. We consider both the AC and DC power flow constraints to solve the UC problem with MOSEK. Moreover, we compare the performance of Algorithm 1 with the UC algorithms proposed in [9]–[11]. We apply the multi-order weakly-strengthened SDP relaxation proposed in [9, Sec. II-D]. We apply the SDP relaxation technique in [10] to relax the binary variables. We also use

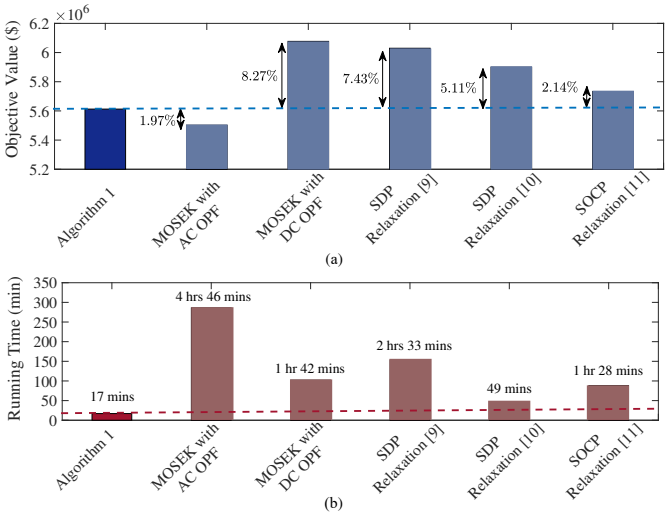

Figure 6. (a) Objective value and (b) running time for Algorithm 1, MOSEK solver, and the UC algorithms in [9]–[11] in an IEEE 300-bus test system.

the proposed heuristic approach in [10] to recover a feasible binary solution. As the proposed algorithms from [9] and [10] rely on DC power flow constraints, we solve an AC OPF problem to check the feasibility of the solution and update the setpoints for the operating generators for AC power flow constraints. The nonlinear AC power flow constraints are considered in [11]. The running time and optimality of the solution with the proposed algorithm in [11] depend on the initial point. Hence, for the sake of fair comparison, we run the algorithm in [11] for five rounds with randomly chosen initial points and report the lowest objective value and the total running time. In Algorithm 1, we use the trained DNN to solve one UC problem instance and run forward propagation for five rounds to select the solution with the smallest objective value. When compared with the proposed algorithm in [11], Fig. 6(a) shows that Algorithm 1 can obtain a near-optimal solution with 2.14% lower objective value for the IEEE 300-bus test system. Also, MOSEK with DC OPF and the proposed UC algorithms in [9] and [10] lead to a solution with higher objective value than Algorithm 1. MOSEK with AC OPF performs the best among the aforementioned algorithms and can obtain a near-optimal solution with slightly lower optimal value (i.e., 1.97%) than Algorithm 1. However, Fig. 6(b) demonstrates that Algorithm 1 benefits from a much lower running time compared with MOSEK with AC OPF and the UC algorithms developed in [9]–[11].

V. Conclusion

In this paper, we developed a deep learning algorithm to solve the UC problem with full AC power flow equations. In our proposed algorithm, we trained a transformer DNN to obtain the on/off state of the generators, and subsequently, solved an AC OPF to determine the generator setpoints. We obtained a near-optimal solution by applying a neural combinatorial optimization algorithm to penalize infeasible solutions during the training process of the proposed DNN. Via numerical simulations on two different IEEE test systems, we showed that our proposed algorithm can obtain a near-optimal solution of the UC problem with considerably lower running time. In particular, when compared with the exhaustive search method in an IEEE 6-bus test system, our proposed algorithm obtained the global optimal solution in about 10 seconds. When compared with the MOSEK solver, simulation results demonstrate a reduction in the running time from five hours to several minutes. When compared with three state-of-the-art UC algorithms in the literature, our proposed algorithm can obtain a solution with at least 2.14% lower operation cost and a lower running time in an IEEE 300-bus test system. For future work, we plan to study the UC problem with uncertainty in the load demand and renewable generation.

References

[1] R. Miller and J. Malinowski, *Power System Operation*, 3rd ed. NY: McGraw-Hill Education, 1994.

[2] H. S. Wilf, *Algorithms and Complexity*, 2nd ed. FL: CRC Press, 2002.

[3] California ISO Department of Market Monitoring, "Annual report on market issues and performance," Available: https://www.caiso.com/Documents/2021-Annual-Report-on-Market-Issues-Performance.pdf, Jul. 2022.

[4] N. Amjady, S. Dehghan, A. Attarha, and A. J. Conejo, "Adaptive robust network-constrained AC unit commitment," *IEEE Trans. Power Syst.*, vol. 32, no. 1, pp. 672–683, Jan. 2017.

[5] K. Šepetanc and H. Pandžić, "Convex polar second-order Taylor approximation of AC power flows: A unit commitment study," *IEEE Trans. Power Syst.*, vol. 36, no. 4, pp. 3585–3594, Jul. 2021.

[6] A. Castillo, C. Laird, C. A. Silva-Monroy, J.-P. Watson, and R. P. O'Neill, "The unit commitment problem with AC optimal power flow constraints," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4853–4866, Nov. 2016.

[7] J. Liu, C. D. Laird, J. K. Scott, J.-P. Watson, and A. Castillo, "Global solution strategies for the network-constrained unit commitment problem with AC transmission constraints," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1139–1150, Mar. 2019.

[8] M. Paredes, L. S. A. Martins, and S. Soares, "Using semidefinite relaxation to solve the day-ahead hydro unit commitment problem," *IEEE Trans. Power Syst.*, vol. 30, no. 5, pp. 2695–2705, Sept. 2015.

[9] M. Ashraphijuo, S. Fattahi, J. Lavaei, and A. Atamtürk, "A strong semidefinite programming relaxation of the unit commitment problem," in *Proc. of IEEE Conf. on Decision and Control (CDC)*, Las Vegas, NV, Dec. 2016.

[10] E. Quarm and R. Madani, "Scalable security-constrained unit commitment under uncertainty via cone programming relaxation," *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4733–4744, Sept. 2021.

[11] F. Zohrizadeh, M. Kheirandishfard, A. Nasir, and R. Madani, "Sequential relaxation of unit commitment with AC transmission constraints," in *Proc. of Int'l. Conf. on Decision and Control (CDC)*, Miami, FL, Dec. 2018.

[12] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. of Int'l. Conf. on Learning Representations (ICLR)*, New Orleans, LA, May 2019.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of Conf. on Neural Information Processing Systems (NIPS)*, Long Beach, CA, Dec. 2017.

[14] I. Bello, H. Pham, Q. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. of Int'l. Conf. on Learning Representations (ICLR)*, Toulon, France, Apr. 2017.

[15] MOSEK. [Online]. Available: https://www.mosek.com.

[16] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 92–107, Feb. 2012.

[17] The University of Washington, power systems test case archive. [Online]. Available: https://www.ee.washington.edu/research/pstca.

[18] Independent Electricty System Operator (IESO). [Online]. Available: https://www.ieso.ca.

[19] The Digital Research Alliance of Canada. [Online]. Available: https://www.alliancecan.ca/en.