EasyChair Preprint

№ 4533

Machine Learning Assisted Stochastic Unit Commitment: A Feasibility Study

Farshad Mohammadi, Mostafa Sahraei-Ardakani,
Dimitris N. Trakas and Nikos D. Hatziargyriou

November 7, 2020

Machine Learning Assisted Stochastic Unit Commitment: A Feasibility Study

Farshad Mohammadi[1], *Student Member, IEEE*, Mostafa Sahraei-Ardakani[1], *Member, IEEE*, Dimitris N. Trakas[2], *IEEE, Member,* and Nikos D. Hatziargyriou[2], *Fellow, IEEE*

Abstract— Stochastic unit commitment is an effective model for generation scheduling, in the presence of substantial uncertainty. However, effectiveness comes with substantial computational cost. Generally, stochastic unit commitment needs more time than other standard methods such as deterministic, to solve the unit commitment problem. In this paper, the results of initial feasibility studies are presented aiming to find out if using machine learning-based models can facilitate solving stochastic unit commitment problems. A real-world, large-scale test case is used to demonstrate the capabilities and shortcomings of machine learning algorithms in reducing the calculation time without sacrificing accuracy. Our feasibility study reveals that while it is unlikely to train a machine learning model to solve the problem as a standalone platform, it is possible to use a trained machine learning model to assist in accelerating the solution of the stochastic model.

***Index Terms*— Large-scale systems, Load shedding, Machine Learning, ML-assisted stochastic unit commitment, power outage, power system reliability, preventive operation, scenario creation, severe weather, stochastic optimization, transmission outage.**

NOMENCLATURE

A. Sets

g	Index of the generator, $g \in G$
n	Index of the bus, $n \in N$
k	Index of the transmission line and transformer, $k \in K$
m	Index of the monitored transmission line, $m \in M$
s	Index of the scenario, $s \in S$
o	Index of the outage, $o \in O$
frm	Set of starting bus of lines
to	Set of ending bus of lines

B. Parameters

c	Cost of generation
c^{NL}	No-load cost for generator
c^{SU}	Start-up cost for generator
c^{SD}	Shut-down cost for generator
c^{lsh}	Load shedding and Over-generation cost (penalty)
π	Scenario possibility
PG^{max}	Maximum generation power by generator
PG^{min}	Minimum generation power by generator
F^{max}	Maximum thermal capacity of the line
PTDF	Power transfer distribution factor matrix

C. Variables

F	Line flow vector
FC	Flow canceling transactions vector
P	Net nodal injected power vector
PG	Generated power of a generator
P^d	Power demand at bus
P^{lsh}	Load shedding
P^{og}	Over-generation
u	Unit commitment binary variable
v	Start-up binary variable
x	Shut-down binary variable

I. INTRODUCTION

Unit commitment (UC) is an optimization model with applications in time-ahead generation scheduling, risk analysis, forward market clearing, and planning in power systems [1], [2]. Simplified versions of the unit commitment problem, with DC power flow models, are well developed. For these models, a variety of techniques are used to facilitate solving the problem quickly [3]. However, for large networks, even with the simplification techniques, solving the stochastic unit commitment problem is still challenging. Given the needs of modern power systems, which require consideration of many uncertainties, achieving high-quality solutions within an acceptable time by commercially available hardware. It is, thus, necessary to develop novel approaches to handle the computational burden of stochastic unit commitment. Multiple sources of uncertainties must be modeled to make the results comply with the real-world standards [4], [5]. Intermittency of renewable energy resources, load uncertainties, generation and fuel availability, and status of transmission lines are some of the uncertainties that should be taken into account. [6].

To address the security concerns within the UC problem, the *N-k* secure UC, known as security-constrained UC (SCUC), has been developed. While research efforts continue to focus on improving the solvers through enhanced algorithms to make them faster and more accurate, the problem is still a computationally burdensome [5]. The problem gets even more challenging when k includes temporal outages of more than two transmission lines, because the topology of the network changes over time.

[1] Farshad Mohammadi and Mostafa Sahraei-Ardakani are with the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT, USA (e-mails: farshad.mohammadi@utah.edu, mostafa.ardakani@utah.edu.)

[2] D. N. Trakas and N. D. Hatziargyriou are with the Electrical and Computer Engineering, National Technical University of Athens, Greece (e-mails: dtrakas@power.ece.ntua.gr, nh@power.ece.ntua.gr).

This research was funded by the NSF ECCS grant # 1839833.

While deterministic methods are less likely to solve this type of problem, as the solution they offer is not efficient nor reliable when there are many sources of uncertainty, stochastic methods seem to be a good fit for this problem.

Stochastic Unit Commitment (SUC) has a primary advantage of being simple to model the uncertainty explicitly, thus offering a reliable solution [7]. However, its main disadvantage that limits its applications is the demanding calculation times. In SUC, a set of scenarios over the uncertain future are defined and used to model the probabilistic nature of uncertainties as a set of deterministic formulations. To achieve quality results, scenario generation, reduction, and aggregation must be properly performed. In general, the higher the number of scenarios the better the solution; this, however, comes at the cost of more calculations [8].

An example of a SUC application is when a severe weather event, such as a hurricane, is predicted to impact the network [8], [9]. Hurricanes can cause tens or even hundreds of transmission outages over the course of the impact, often in a few hours. These outages can be predicted in advance, but the predictions include uncertainty. While scenario selection, for preventive stochastic unit commitment during hurricanes, is difficult, solving the problem within a satisfactory time can be very challenging. An efficient method to generate scenarios, called multidimensional scenario selection (MDSS) is introduced in [8], where multiple aspects of information regarding each uncertainty are used to generate the desired number of scenarios. In [10], a new algorithm and set of equations capable of handling multiple line outages are introduced to model the problem as a preventive SUC problem. While a combination of the MDSS and the formulation introduced in [10] delivers a high-quality solution, the overall required time can still be very long for many cases.

In this article, we evaluate the feasibility of using Machine Learning (ML) algorithms, to facilitate solving the SUC problem. ML techniques have been successfully used for more than a decade ago to solve SCUC problems in order to determine dynamic security constraints, e.g. [11], or to solve dynamic security problems, such as in [12], [13]. In this paper, however, the ML application is different, since it aims to investigate how ML can improve solution times without sacrificing accuracy by predicting operating conditions or essential constraints. Thus, a perfect ML model is assumed to provide final results with the same accuracy as the solution of SUC. Using this perfect model, this paper investigates the sensitivity of a feasible solution when an imperfect ML model is trained and used.

The remainder of this paper is organized as follows: Section II reviews the original SUC problem and explains the use of machine learning algorithms, so the challenges of the original problem, and capabilities and limitations of ML are revealed. A feasibility study is presented in Section III to determine how much and how ML can help to solve the problem. Section III provides results from the application of the method on a large-scale network as a test-case. Finally, Section IV concludes the paper.

II. Background

This section explains the stochastic unit commitment problem, the challenges with solving the model, and the potentials of machine learning to facilitate solving the problem.

A. Original SUC Problem

SUC is an optimization problem defined over a set of scenarios that represent realizations of the uncertain future. The goal of SUC is to minimize the objective function, often operation cost, subject to physical and reliability constraints of the network. With high levels of uncertainties, the objective function should include not only generation costs, but also penalized load shedding (unserved load) and over-generation. Load shedding and over-generation are allowed, since multiple outages are expected to lead to such violations due to lack of sufficient transmission capacity, as well as disconnected load or generation. The objective function is defined as:

$$Minimize \sum_s \{\pi_{(s)} \sum_t [\sum_g (c_{(g)} PG_{(s,g,t)} + c^{NL}_{(g)} u_{(s,g,t)} + c^{SU}_{(g)} v_{(s,g,t)} + c^{SD}_{(g)} x_{(s,g,t)}) + \sum_n c^{lsh} (P^{lsh}_{(s,n,t)} + P^{og}_{(s,n,t)})]\}. \quad (1)$$

The objective function in (1) is subject to:

$$PG^{min}_{(g)} u_{(s,g,t)} \le PG_{(s,g,t)} \le PG^{max}_{(g)} u_{(s,g,t)} \qquad \forall\, s, g, t \quad (2)$$

$$P_{(s,n,t)} = \left[PG_{(s,n,t)} + P^{lsh}_{(s,n,t)}\right] - \left[P^{d}_{(s,n,t)} + P^{og}_{(s,n,t)}\right] \qquad \forall\, s, n, t \quad (3)$$

$$-F^{max}_{(m)} \le F_{(s,m,t)} \le F^{max}_{(m)} \qquad \forall\, s, t \text{ and } \forall\, m \in M_{(s)} \quad (4)$$

$$F_{(s,m,t)} = (\boldsymbol{PTDF}_{(m)} \times \boldsymbol{P}_{(s,t)}) + \sum_{o \in O_{(s,t)}} \left(PTDF_{(m,frm_{(o)})} - (PTDF_{(m,to_{(o)})}\right) FC_{(s,t,o)} \qquad \forall\, s, t \text{ and } \forall\, m \in M_{(s)} \quad (5)$$

$$(\boldsymbol{PTDF}_{(o)} \times \boldsymbol{P}_{(s,t)}) - FC_{(s,t,o)} + \sum_{o' \in O_{(s,t)}} \left(PTDF_{\left(o,frm_{(o')}\right)} - (PTDF_{\left(o,to_{(o')}\right)}\right) FC_{(s,t,o')} = 0 \qquad \forall\, s, t \text{ and } \forall\, o \in O_{(s,t)} \quad (6)$$

$$\sum_n [(PG_{(s,n,t)} + P^{lsh}_{(s,n,t)}) - P^{d}_{(s,n,t)} + P^{og}_{(s,n,t)}] = 0 \qquad \forall\, s, t \quad (7)$$

$$u_{(s,g,t)} = u_{(s',g,t)} \qquad \forall\, s, s' \in S \quad (8)$$

$$A_{s,t}(x_{s,t}, u_{s,t}) \le 0, \quad (9)$$

$$B_{s,t}(x_{s,t}, u_{s,t}) = 0, \quad (10)$$

Eq. (2) applies generation maximum and minimum limits, (3) calculates the nodal net injection power, while load shedding and over-generation are modeled as generator and load,

respectively. (4) keeps line flows within the acceptable range for each transmission line. Note that, while $M_{(s)}$ can include all lines, it can be any subset of lines that are selected to be monitored. Later it is shown that this selection of lines can be made through a trained ML model.

When a set of possible line outages is noted by O, (5) and (6) together calculate the power flow for monitored lines that are defined by $M_{(s)}$, by considering the effects of line outages defined by O. Note that (5) and (6) are defined based on power transfer distribution factor [14], ***PTDF***, and flow-cancelling transaction concepts [15], and model any number of line outages. (7) extends (3) to apply power balance in the network where load shedding and over-generation are modeled as generators and loads, respectively. The last equation forces the commitment status of generation units to be the same for all scenarios, which means the commitment variable is modeled as a first-stage variable in the defined multi-stage stochastic optimization problem.

It should be mentioned that (2) to (8) represent the constraints of interest in this study, while other standard constraints, such as ramping up/down, minimum up/down times for generators, constraints regarding allowed values of load shedding and over-generation, and other network constraints are considered as well. Those equality and inequality constraints are modeled through (9) and (10), respectively. Interested readers are referred to [10] for the complete formulation and algorithm description.

The power transfer distribution factor is among the most efficient methods to solve the power flows in UC problems. However, when it is used for large-scale networks with multiple outages, the number of variables and constraints becomes extremely large. Notably, (5) and (6) are the main equations responsible for growing constraints in numbers and complexity, as they combine standard power flow calculations considering the effects of outages. Moreover, load shedding and over-generation increase the size of the defined problem, as each such condition should be modeled as a load or generation unit. A promising fact in solving the SUC problem is that, while the original problem includes a large number of variables and constraints, not all variables are necessary to be calculated within the optimization process, nor all the constraints reach their limits. This suggests that if it is possible to distinguish between the essential variables and constraints to be included in the optimization problem and those that are not, the problem could be solved easier by removing unnecessary variables and constraints. It is worth mentioning that variables and constraints that are excluded from the optimization problem, can be calculated outside the optimizer so that the complete set of results is verified.

B. Machine Learning Concept

Nowadays, the capability of ML algorithms is no longer limited to only pattern recognition, when enhanced ML models make computers capable of learning to do scientific tasks [16], [17]. Increasing amounts of data analyzed by ML methods, can provide solutions of high accuracy and increased speed in power system problems, such as UC [18].

Supervised machine learning algorithms can learn through examples known as observations. Each observation consists of inputs paired with the corresponding output(s) [19]. The training algorithm searches for patterns and correlations between inputs and outputs. After training, a supervised learning algorithm for any new unseen inputs determines/predicts the outputs. At its basic form, a supervised learning algorithm can be written as:

$$Y = f(x), \tag{11}$$

where Y represents the predicted output(s) that is determined by a mapping function, f, over the value of x. The mapping function used to connect inputs/features to a predicted output(s) is created during training.

SUC, as a mixed-integer linear programming model, has two types of variables: continuous variables such as scheduled generation power, line flow, and integer variables such as commitment status. However, it is possible to assume another set of imaginary integer variables that determines whether each constraint (such as line flow) is binding. Only binding (or near-binding) constraints should be included in the SUC. Among variables, most of the required calculations concern the commitment status of generation units and line constraints. In other words, solving SUC when commitment status is known, and without line constraints is as simple as solving an economic dispatch problem and can be done very fast.

The supervised machine learning classification seems promising if we want to choose which constraints should be included in the calculations. In a perfect condition, when there is enough training data to cover the whole operating spectrum of the SUC problem, it is possible to train the supervised mapping function in (11), f, in order to obtain these essential constraints. Using the trained model, by providing the input data, such as network information, expected uncertainties, and demand data, any desired Y can be acquired rapidly with no need to run SUC again.

III. Feasibility Study

The objective of this section is to evaluate different inputs and outputs variables to discover which ones are most suitable to be considered as input features and outputs for the ML model. In order to achieve the objective, possible candidates are investigated to find the best candidates. Then as a feature engineering process, some are rejected, and the list is narrowed down to a few primary candidates. Next, a feasibility analysis is performed for each candidate to determine the best options.

A. SUC cannot be Replaced with Machine Learning

As any ML model must be trained with a set of solved SUC cases, the accuracy of the trained ML can never be better than the original SUC solution. Hence, the primary motivation for using a trained ML model is to reduce the calculation time or hardware requirements, while maintaining the same accuracy as the original SUC. A large number of variables require a large number of solved cases to train the ML model. Since the solution time of SUC is long, obtaining a large enough data set in order to train the ML becomes a real bottleneck. On the other hand, if the solution time of the SUC problem is short enough,

there is no reason to use ML for its solution.

Due to strict SUC constraints, and considering the fact that it is unlikely to train an ML model to perfectly predict every output exactly the same as solving original SUC, a potential solution is to use a machine learning as an assistant to SUC to facilitate the solution process. This way, not only is the result accurate similar to the original SUC, but also perfect ML performance is not necessary. This would translate in a reduced need to training data.

Recognizing the fact that high accuracy in UC problem solutions are required and the limited number of solved cases that can be practically used to train the ML model, an ML assisted SUC method is proposed in this paper. The objective is to use the trained ML to guess/predict the entire or a part of the final solution. Next, this possible solution is implemented into the SUC as a **warm start** (advanced start or MIP-start) to solve the case accurately and at increased speed. However, before making it possible to use trained ML to predict the output, one more question should be answered: what inputs can be used as input features, and what output should be predicted to help solve the SUC problem, if possible at all? The next subsections offer some answers to this question.

B. Candidate Inputs and Outputs

In SUC, the desired outputs are the commitment status (binary), the scheduled generated power (continuous), line flow (continuous). Candidate inputs that can be used as features for ML are the network topology, data for generators, loads (nodal with hourly profile), and data expressing uncertainties.

Each ML will especially be trained for defined network topology and generation/load units, and variations can be modeled through uncertainties. For example, assume that there are 100 lines in a network vulnerable to failure due to a hurricane. To train the ML model, thousands of SUC problems, known as observations in ML, should be defined and solved, each of which represents a possible hurricane and chance of damage to some of those 100 lines. Then the trained model can predict the SUC solution in response to any unseen hurricane. Note that, the trained ML model can be used for a considerable time as the network topology evolves slowly over time, requiring re-training when the network changes drastically.

It should be noted that continuous variables are harder to predict, meaning that more solved cases are required. Alternatively, an economic dispatch can be used to calculate generated power and line flow easily when the commitment variable is known (easy in comparison with original SUC). Hence, we use ML to predict integer variables as a supervised classification machine learning model.

C. Test Cases and Software Selection

In this feasibility study, we use as test case a synthetic grid on the footprint of South Carolina with 500 buses, 597 lines, and 90 generation units. The complete information can be found in [20], [21]. For the load profile, we used a daily load profile, as in [22].

The main code that handles the MIP-SUC problem is developed on the Java platform trough ELSIPSE IDE [23], and implements IBM CPLEX optimization studio ver. 12.10 [24] as a solver. The whole software-package runs on a system with 128 GB of DDR4 memory, and AMD 3900X as a processor unit. It should be mentioned that, while the CPU has 24 processing cores, we only utilize 4 to reduce the impact of background process on the solution time.

D. Commitment Status as Output

In this subsection, the goal is to determine how much calculation could be saved if generators' commitment status could be predicted accurately. In the feasibility test, it is assumed that a trained ML exists, which can predict the commitment status at different levels of accuracy from perfectly accurate to partly inaccurate. Then, the predicted commitment is implemented as a **warm-start** to SUC, and the accuracy of the results and solution time are compared with a **cold-start** solutions, meaning that the unit commitment status is calculated from scratch without trained ML assistance. This way, it is possible to evaluate how much time can be saved by predicting commitment status with various levels of error. Note that cold-start with all relevant constraints in effect serves as a reference for benchmarking the other solutions.

The original SUC consists of 10 scenarios, including ten lines with failure chance, and takes 206 seconds to solve. Next, by using the commitment obtained by solving the original SUC, a certain percentage of generator statuses is randomly changed. This percentage is varied from 0% to 100%, where each case represents a simulation of the accuracy of ML. Next, the trained ML with different prediction accuracies of commitment status is used to solve the SUC problem as a **warm start** model**.** While all cases result in accurate optimal value for the objective function, the solution time is highly sensitive to errors that may exist in the predicted commitment variable. The results are shown in Table 1.

Table 1. Effect of Predicting Commitment Variable on Solution Time for SUC, when Prediction Includes Different Levels of Error

Case	***Solution Time (Sec.)***
Original SUC	***206***
Assisted with commitment variable with 0.0% error	**196**
Assisted with commitment variable with 0.1% error	**197**
Assisted with commitment variable with 0.2% error	***202***
Assisted with commitment variable with 0.3% error	**216**
Assisted with commitment variable with 0.5% error	**225**
Assisted with commitment variable with 1.0% error	**231**
Assisted with commitment variable with 5.0% error	**239**
Assisted with commitment variable with 10.0% error	**240**
Assisted with commitment variable with > 10% error	**240~250**

If commitment status could be predicted with 100% accuracy, the solution time could be improved by 5%, which is not a significant improvement. On the other hand, even small errors in prediction (more than 0.3%) will increase the solution time compared to the original SUC. This can be justified considering the time CPLEX needs to implement warm-start, verify if the solution is feasible, and then calculate the other variables of the

problem. In case the provided warm-start solution is not accurate, CPLEX will try to fix the solution. The overall time for implementation, verification, doing other calculations, and fixing the solution in cases of errors, is longer than what is required to calculate a first feasible solution with cold-start.

E. Suspected Limit Violating Lines as Output

As mentioned before, if the trained ML can predict variables that do not violate their constraints in all conditions, those constraints could be removed from the model without an impact on final results. Equations (4) to (6) enforce not only many constraints to the problem, but also more complex than others. Hence, a trained ML that removes unnecessary constraints in these two equations reduces calculation times. In the following, it is assumed that a trained ML model can predict which line violates its thermal limit, represented by M in the formulation (4). The feasibility study is done in the following steps:

1- SUC is solved with only suspected lines included in (4) to (6).
2- After SUC is solved, a power flow is solved, and all lines are compared with their corresponding limitations to find, if any violation happened.
3- On case of violation, lines with flows exceeding their thermal limits are added to m.
4- SUC is solved repeatedly with new constraints until there is no new violation.

Table 2 presents results. Note that, original SUC problem is the same as is the one used in Table 1.

Table 2. Effects of Predicting Lines violating their constraints with Different Levels of Errors

Case	***Solution Time (Sec.)***
Original SUC	***206***
Assisted, suspected lines with 0.0% error	**8**
Assisted, suspected lines with 10% error	**25**
Assisted, suspected lines with 25% error	**52**
Assisted, suspected lines with 50% error	**89**
Assisted, suspected lines with 60% error	**108**
Assisted, suspected lines with 75% error	**149**
Assisted, suspected lines with 100% error	**405**

According to Table 2, predicting suspected lines with good accuracy can significantly save the computational time, while the quality of the final solution is the same as the original solution. Moreover, saving is not as sensitive to errors as it was with the commitment variable. Thus, predicting lines suspected to violate their limits is a good candidate as output of the trained ML. It should be mentioned that, while combining both commitment and suspected lines as outputs of the trained ML model can reduce the calculation time even further (when both are correctly predicted calculation time is as low as 7 seconds), this is useful only when the predictions are perfectly accurate due to the high sensitivity of the solution time on the commitment status accuracy.

F. Input Features

The input features to train the ML model should be the same as used as inputs when we use the trained ML to predict desired outputs. As explained before, the input features for ML training should not include network topology data; instead, they should include data related to uncertainties. As the prime goal is to assist solving the SUC problem, and by knowing that in SUC, uncertainties are represented by scenarios, the input features should be scenarios or data derived from them.

While the part of data that is the same among various scenarios, carries no useful information that can be used in predictions, the useful part concerns changes over scenarios and different conditions. For example, assume there are 100 lines out of a total of 999 lines in the network that are vulnerable to damage. In different conditions, some of those 100 will have a chance to fail. If any of those conditions are modeled in the SUC problem, the only difference between scenarios is related to those 100 lines and not all the 999 lines. Hence, scenarios regarding different conditions should be worked out before used as input features.

Hence, the input feature set should include those elements of the network that are vulnerable to uncertainty. The input may be defined on integer or continuous variables. For example, if the uncertainty is related to the failure of lines, it can be defined as a binary variable (classification) with its value as 0 if a line has a chance to fail, and its value as 1 if no chance for the failure of the line exists. The same uncertainties could be defined with continuous variables as temporal failure chance. While the binary definition may seem less accurate than actual temporal failure chance, it includes much fewer variables (as it is not a function of time) and needs less number of solved case for training the ML model. Moreover, ML trained for classification with binary variable (only 2 classes), can be much more accurate than a continuous variable. Choosing each of methods to define variables, ultimately depends on the design of the ML algorithm and application for the trained model.

IV. CONCLUSION

Recently, variations of stochastic unit commitment have been used for enhanced operation under complex conditions, such as changing network topology. Multiple sources of uncertainties must be considered, to ensure the obtained results are reliable and efficient. Stochastic unit commitment offers accurate solutions at the cost of long computational times and sometimes advanced hardware. This paper presented a feasibility study on using ML algorithms to assist stochastic unit commitment solvers, with the aim of reducing the computation time and hardware requirements. Although the full replacement of the original SUC by ML algorithms is debatable, we argue that a trained ML model can assist the SUC solution by providing initial predictions, i.e., through a warm-start process. Initial tests show that predicting lines that are likely to reach their flow limits can significantly reduce the computational time. The results also showed that predicting generation commitment status is likely not effective.

FUTURE WORK

Our future research plan includes using a real-world, large-

scale network with multiple uncertainty sources to train an ML model. The trained ML model will, then, be used to solve different unseen SUC problems to determine its accuracy and the savings on solution time.

REFERENCES

[1] B. Saravanan, S. Das, S. Sikri, and D. P. Kothari, "A solution to the unit commitment problem-a review," *Front. Energy*, vol. 7, no. 2, pp. 223–236, 2013, doi: 10.1007/s11708-013-0240-3.

[2] A. Bhardwaj, V. K. Kamboj, V. K. Shukla, B. Singh, and P. Khurana, "Unit commitment in electrical power system - A literature review," in *2012 IEEE International Power Engineering and Optimization Conference, PEOCO 2012 - Conference Proceedings*, 2012, pp. 275–280, doi: 10.1109/PEOCO.2012.6230874.

[3] B. Stott, J. Jardim, and O. Alsaç, "DC power flow revisited," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1290–1300, 2009, doi: 10.1109/TPWRS.2009.2021235.

[4] S. Bahrami and V. W. S. Wong, "Security-Constrained Unit Commitment for AC-DC Grids with Generation and Load Uncertainty," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 2717–2732, May 2018, doi: 10.1109/TPWRS.2017.2749303.

[5] Y. Fu and M. Shahidehpour, "Fast SCUC for large-scale power systems," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 2144–2151, Nov. 2007, doi: 10.1109/TPWRS.2007.907444.

[6] P. A. Ruiz, C. R. Philbrick, E. Zak, K. W. Cheung, and P. W. Sauer, "Uncertainty Management in the Unit Commitment Problem," *IEEE Trans. Power Syst.*, vol. 24, no. 2, pp. 642–651, May 2009, doi: 10.1109/TPWRS.2008.2012180.

[7] E. Du, N. Zhang, C. Kang, and Q. Xia, "Scenario Map Based Stochastic Unit Commitment," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 4694–4705, Sep. 2018, doi: 10.1109/TPWRS.2018.2799954.

[8] F. Mohammadi and M. Sahraei-Ardakani, "Multidimensional Scenario Selection for Power Systems With Stochastic Failures," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4528–4538, Nov. 2020, doi: 10.1109/TPWRS.2020.2990877.

[9] A. Arab, A. Khodaei, S. K. Khator, K. Ding, V. A. Emesih, and Z. Han, "Stochastic pre-hurricane restoration planning for electric power systems infrastructure," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 1046–1054, Mar. 2015, doi: 10.1109/TSG.2015.2388736.

[10] F. Mohammadi and M. Sahraei-Ardakani, "Tractable Stochastic Unit Commitment for Large Systems During Predictable Hazards," *IEEE Access*, vol. 8, pp. 115078–115088, 2020, doi: 10.1109/ACCESS.2020.3004391.

[11] K. A. Papadogiannis and N. D. Hatziargyriou, "Optimal Allocation of Primary Reserve Services in Energy Markets," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 652–659, Feb. 2004, doi: 10.1109/TPWRS.2003.820702.

[12] E. S. Karapidakis and N. D. Hatziargyriou, "Online preventive dynamic security of isolated power systems using decision trees," *IEEE Trans. Power Syst.*, vol. 17, no. 2, pp. 297–304, May 2002, doi: 10.1109/TPWRS.2002.1007896.

[13] E. M. Voumvoulakis and N. D. Hatziargyriou, "Decision trees-aided self-organized maps for corrective dynamic security," *IEEE Trans. Power Syst.*, vol. 23, no. 2, pp. 622–630, May 2008, doi: 10.1109/TPWRS.2008.920194.

[14] H. Ronellenfitsch, M. Timme, and D. Witthaut, "A Dual Method for Computing Power Transfer Distribution Factors," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1007–1015, Mar. 2017, doi: 10.1109/TPWRS.2016.2589464.

[15] P. A. Ruiz, E. Goldis, A. M. Rudkevich, M. C. Caramanis, C. R. Philbrick, and J. M. Foster, "Security-Constrained Transmission Topology Control MILP Formulation Using Sensitivity Factors," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1597–1605, 2017, doi: 10.1109/TPWRS.2016.2577689.

[16] S. Marsland, "Chapter 1: Introduction," in *Machine Learning: An Algorithmic Perspective*, Second edi., Boca Raton, FL, USA: CRC Press, Taylor and Francis Group, 2015, pp. 1–11.

[17] Y. Baştanlar and M. Özuysal, "Introduction to machine learning," *Methods Mol. Biol.*, vol. 1107, pp. 1-36o, 2014, doi: 10.1007/978-1-62703-748-8_7.

[18] N. Hatziargyriou, "Machine learning applications to power systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2049 LNAI, pp. 308–317, 2001, doi: 10.1007/3-540-44673-7_20.

[19] Y. Zhang, "Types of Machine Learning Algorithms," in *New Advances in Machine Learning*, Rijeka, Croatia: InTech, 2010, pp. 19–20.

[20] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid Structural Characteristics as Validation Criteria for Synthetic Networks," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3258–3265, Jul. 2017, doi: 10.1109/TPWRS.2016.2616385.

[21] Adam Birchfield, "ACTIVSg2000: 2000-bus synthetic grid on footprint of Texas," 2019. [Online]. Available: https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg2000/. [Accessed: 26-Feb-2019].

[22] F. Mohammadi, M. Sahraei-Ardakani, Y. M. Al-Abdullah, and G. T. Heydt, "Coordinated Scheduling of Power Generation and Water Desalination Units," *IEEE Trans. Power Syst.*, vol. 34, no. 5, pp. 3657–3666, Sep. 2019, doi: 10.1109/TPWRS.2019.2901807.

[23] ECLIPSE Foundation, "Eclipse IDE for Java EE Developers | Eclipse Packages," 2020. [Online]. Available: https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers. [Accessed: 20-Feb-2019].

[24] IBM, "CPLEX Optimizer | IBM," 2018. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer. [Accessed: 20-Feb-2019].