Machine Learning for Expediting Security Constraint Unit Commitment Solution

Álinson S. Xavier [1] **Feng Qiu** [1] Shabbir Ahmed [2]

[1] Argonne National Laboratory, Lemont, IL

[2] Georgia Institute of Technology, Atlanta, GA

FERC Technical Conference
June 26, 2019, Washington DC

Security Constrained Unit Commitment and Power Systems

- Unit commitment (UC)
  - Generator scheduling and power output levels
- Security constrained UC (SCUC)
  - Physical laws of power flows in transmission network (Kirchhoff's Laws and Ohm's Laws)
  - Overhead transmission line thermal limits
  - Security requirement: N-1 contingency
- SCUC and power systems
  - Electricity market clearning (day-ahead market)
  - Reliability assessment
  - System expansion

Demand for Computational Performance Improvement

- Day-ahead electricity market
  - $400 billion electricity market
  - Industry standard: MIP gap 0.1% within 30 minutes; ISO often terminates solve with a large gap due to time limits
  - Potential $200 million cost savings if MIP gap is reduced to 0.05%
- Computational performance improvement are needed for
  - More accurate modeling of energy components, e.g., combined heat&power generators
  - More accurate modeling of operations, e.g., smaller time periods
  - Operations under uncertainty, e.g., stochastic UC
- Literature review
  - Stronger formulations [Rajan et al. 05‘; Carrino & Arroyo 06‘; Ostrowsk et al. 12‘; Atakan et al. 17‘; Gentile et al. 17‘; Morales-Espana et al. 15‘]
  - Cutting planes [Lee et al. 04‘; Damcı-Kurt et al. 15‘;Ostrowski et al. 12‘]
  - Decomposition [Ma & Shahidehpour 98‘; Feizollahi et al. 15‘; Kim et al. 18‘]

SCUC Application Setting: Routinely Solved

- SCUC is solved routinely
  - Multiple times per day and all year around
    - Day-ahead SCUC: once per day;
    - Resource reliability and commitment: twice per day;
    - Reliability assessment commitment: 24 times per day
  - Data vary slightly iteration to iteration
    - Same topology and same set of generators
    - Slightly different net loads, cost curves
  - "Patterns" of characteristics of optimal scheduling can be observed
  - Volume SCUC data saved, e.g., 2-3 terabyte SCUC data at MISO each year
- Current practice: one-shot optimization
  - Start each solve from scratch, dump all information when finished

Machine Learning for Solving Optimization Problems

- Early work using machine learning to solve SCUC
  - Artificial neural networks to predict generator status [Sasaki et al, 92';Z Ouyang & Shahidehpour 92'; Park 93'; Huang 97']
  - Not successful even on small instances with simplifications
- Machine learning for optimization
  - Machine learning for branch-and-bound algorithms [Khalil et al. 16',16',17';Alvarez 16'; Lodi & Zarpellon 17';]
  - Machine learning for combinatorial optimization [Baluja&Davies 97'; Bojnordi & Ipek 16'; Dai et al. 17']
- Our perspective
  - Focus on a specific class of problems - Day-Ahead SCUC
  - Routinely solved problems share great similarity
  - Work within, instead of replace, existing optimization paradigm

Understanding Security Constrained Unit Commitment

$$\text{minimize} \quad \sum_{g \in G} c_g(x_{g\bullet}, y_{g\bullet}) \tag{1}$$

$$\text{subject to} \quad (x_{g\bullet}, y_{g\bullet}) \in \mathcal{G}_g \qquad \forall g \in G \tag{2}$$

$$-F_l^c \leq \sum_{b \in B} \delta_{lb}^c \left(\sum_{g \in G_b} y_{gt} - d_{bt} \right) \leq F_l^c \qquad \forall c \in L \cup \{0\}, l \in L, t \in T. \tag{3}$$

$$x_{gt} \in \{0, 1\} \qquad \forall g \in G, t \in T \tag{4}$$

$$y_{gt} \geq 0 \qquad \forall g \in G, t \in T \tag{5}$$

- Once x_{gt} (gen. commit. var.) determined, y_{gt} (dispatch) can be calculated efficiently
- Constraints (3) are necessary only when there are congestions on line l in scenario c in time t

Day-Ahead SCUC

- Application setting
  - Bids and offers submitted at the end of the day
  - Usually 30 minutes time window is given to solve SCUC
- Data that do not change often
  - Generator characteristics: ramping rates, minimum on/off time, capacity,
  - Topology and transmission line thermal limits
- Data that change
  - Generator cost curves, start-up costs
  - Load
    - Geographic distribution
    - Peak load and load profile
- Learning schemes
  - Study the past (or simulated) SCUC input data (i.e., only data that change) and solutions
  - Generate oracles that can give hints to MIP solvers to expedite next solve, based on input data

Learning Violated Transmission Constraints

- Transmission constraints significantly impact computational performance; few of them are actually needed
- Transmission oracle
  - Predict which transmission constraints should be added to the relaxation and which constraints can be safely omitted based on statistical data
- Transmission(security) constraints

$$-F_l^c \leq \sum_{b \in B} \delta_{lb}^c \left(\sum_{g \in G_b} y_{gt} - d_{bt} \right) \leq F_l^c.$$

- A vector of hints $h_{l,c,t} \in \{\texttt{ENFORCE}, \texttt{RELAX}\}$, for $l \in L, c \in L \cup \{0\}, t \in T$, indicating whether the thermal limits of transmission line l, under contingency scenario c at time t, should be enforced
- A simple prediction algorithm: $h_{l,c,t} = \texttt{ENFORCE}$ if the transmission constraint (l, c, t) was necessary during the solution of at least k (e.g., 1%) percent of the training samples.

Learning Initial Feasible Solutions

- High-quality initial feasible solutions are critical in closing the gap in SCUC
- Feasible solution oracle
  - Quickly produce high-quality valid solutions for SCUC, based on historical data
- Instance-based learning
  - Given instances $I(p^i)$ and their optimal solutions (x^i, y^i), and the test instance $I(\tilde{p})$
  - Output the integer part of solutions (i.e., x^i) of the first k closest instances to $\tilde{p}$ with regard to a predefined norm
- Feed k integer solutions to MIP solver as warm starts
  - $k = 10$
- Suitable for online learning

Learning Affine Subspaces

- System operators learn patterns of optimal generator scheduling after years of work
- Affine subspace oracle
  - Predict a list of hyperplanes $(h^1, h_0^1), \ldots, (h^k, h_0^k)$ such that, with very high likelyhood, the optimal solution (x, y) of $I(\tilde{p})$ satisfies $\langle h^i, x \rangle = h_0^i$, for $i = 1, \ldots, k$
  - Currently focus on the following hyperplanes:

$$x_{gt} = 0,$$
$$x_{gt} = 1,$$
$$x_{gt} = x_{g,t+1}$$

  - More complicated hyperplanes
    - $x_{g,t}$ generators of same buses
    - $x_{g,t}$ generators of same type
    - other patterns discovered in optimal scheduling

Learning Affine Subspaces

- Construct affine subspace oracle $\phi_{(h,h_0)} : \mathbb{R}^n \to \{\texttt{ADD}, \texttt{SKIP}\}$
  - Let $(h, h_0) \in \mathcal{H}$, and $I(p^1), \ldots, I(p^s)$ be training samples, and $(x^1, y^1), \ldots, (x^s, y^s)$ be their respective optimal solutions
  - $z^i = 1$ if $\langle h, x^i \rangle = h_0$, and $z^i = 0$ otherwise
  - $\bar{z} = \sum z^i / s$
  - If $\bar{z} \leq 0.05$ always return `SKIP`
  - If $\bar{z} \geq 0.95$ always return `ADD`
  - Otherwise, train binary classifier $\theta_{(h,h_0)}$ and evaluate its accuracy using k-fold cross validation.
  - If accuracy ≥ 0.95, use the classifier. Otherwise, always return `SKIP`.
  - Reduce feature dimension by only using (1) the peak system load, (2) the hourly system loads, (3) the average production cost of generator g and (4) the average production costs of the remaining generators

Experiment Settings and Instances

- Experiment settings
  - Java, Python 3, using pandas and scikit-learn
  - IBM ILOG CPLEX 12.8.0 as MIP solver
  - Traning: Intel Xeon E5-2695v4, 36 cores, 128GB DDR4; testing: AMD Ryzen 7 1700, 8 cores, 16GB DDR4
- Test instances
  - Adapted from MatPower [Zimmerman at el. 11‘]

Instance	Buses	Units	Lines
case1888rte	1,888	297	2,531
case1951rte	1,951	391	2,596
case2848rte	2,848	547	3,776
case3012wp	3,012	502	3,572
case3375wp	3,374	596	4,161
case6468rte	6,468	1,295	9,000
case6470rte	6,470	1,330	9,005
case6495rte	6,495	1,372	9,019
case6515rte	6,515	1,388	9,037

Figure: Test Power System Instances

Training Data

- 300 instances for training, solved to optimality; 30 instances for testing
- Data generation
  - Production and startup costs: uniform distribution
  - Geographical load distribution: uniform distribution
  - Peak system load and temporal load profile: Gaussian distribution from PJM hourly data; Uniform distribution for peak load

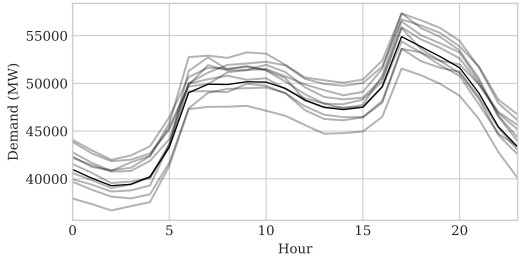


Figure: Sample of artificially generated load profiles.

A Benchmark Algorithm - Contingency Screening

- A contingency screening algorithm
  - Add security constraints only when they are violated
  - Add iteratively until no violation of security constraints

Algorithm 1 Security-Constrained Unit Commitment

Let L^M be the set of monitored transmission lines
Let L^V be the set of transmission lines susceptible to disruption
Create a relaxation of SCUC without any transmission constraints
Solve the current relaxation
Compute pre-contingency flow f^0 using ISF
Compute post-contingency flow f^v using LODF, $\forall v \in L^V$
Let $\gamma_m^v = \max\{F_m - f_m^v, 0, f_m^v - F_m\}, \forall v \in L^V \cup \{0\}, m \in L^M$
Let $\Gamma = \{(v,m) \in (L^V \cup \{0\}) \times L^M : \gamma_m^v > 0\}$ be the violations
if Γ is empty **then return**
else
    For $m \in L^M$, keep in Γ only the pair (v,m) with highest γ_m^v
    Keep in Γ only the k pairs (v,m) having the highest γ_m^v
    For every violation in Γ, add the corresponding cut to the relaxation
    goto step 4

Figure: Contingency Screening Algorithm

- Compare our benchmark algorithm with best literature results Tejada-Arango, Sánchez-Martin, Ramos (2017)

Instance	Parameters	Load	Iterations	Time per Iteration (s)	Total Time (s)	Obj Value
472 buses 752 lines 216 units	gap: 0.10% cold start	0.977	3	4.2	12.6	5,350,329.35
		1.050	4	6.4	25.7	5,874,778.27
		1.029	5	6.5	32.3	5,721,391.51
		1.008	5	4.7	23.4	5,570,239.20
		0.987	2	4.0	8.0	5,420,692.62
		0.809	4	1.9	7.8	4,273,054.93
		0.788	4	2.0	7.8	4,148,955.87
				Average	**16.8**	
				Speedup	**13.7x**	

Instance	Parameters	Load	Iterations	Time per Iteration (s)	Total Time (s)	Obj Value
472 buses 752 lines 216 units	gap: 0.10% Tejada-Arango	0.977	5	38.7	193.4	5,350,163.06
		1.050	7	91.7	642.1	5,874,833.76
		1.029	5	46.0	229.8	5,721,214.78
		1.008	5	45.2	226.0	5,570,332.17
		0.987	2	42.0	84.0	5,421,390.02
		0.809	2	32.2	64.4	4,273,150.99
		0.788	6	28.1	168.5	4,148,600.21
				Average	**229.7**	

Figure: Comparison with Other Screening Algorithm

Performance Evaluation

- Computational performance

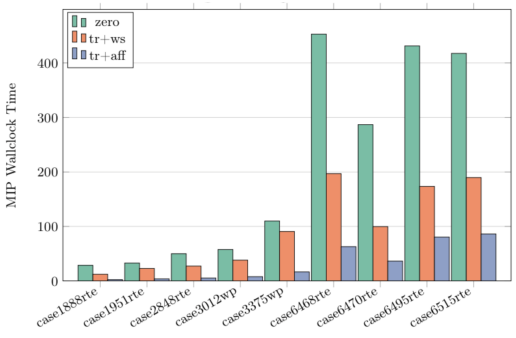


Figure: Solution Time

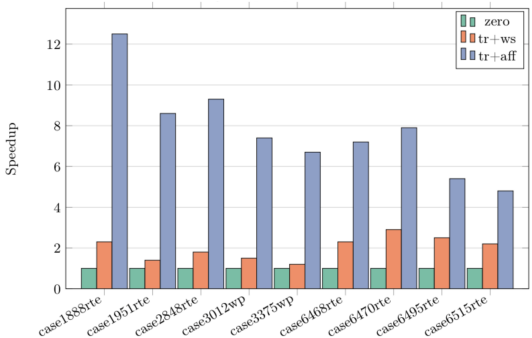


Figure: Speedups Over Benchmark Performance

Performance Evaluation

- Solution quality

	zero		tr+ws			tr+aff		
Instance	Time (s)	Gap (%)	Time (s)	Speedup	Gap (%)	Time (s)	Speedup	Gap (%)
case1888rte	28.6	0.09	12.3	2.3x	0.08	2.3	12.5x	0.06
case1951rte	32.8	0.10	22.9	1.4x	0.09	3.8	8.6x	0.04
case2848rte	49.9	0.09	27.2	1.8x	0.09	5.3	9.3x	0.07
case3012wp	57.6	0.09	38.1	1.5x	0.09	7.8	7.4x	0.04
case3375wp	109.9	0.08	90.7	1.2x	0.09	16.5	6.7x	0.05
case6468rte	453.0	0.07	196.9	2.3x	0.09	62.9	7.2x	0.03
case6470rte	286.8	0.06	99.7	2.9x	0.09	36.4	7.9x	0.02
case6495rte	431.3	0.05	173.5	2.5x	0.08	80.5	5.4x	0.04
case6515rte	417.6	0.08	189.7	2.2x	0.08	86.2	4.8x	0.05

Figure: Impact of machine-learning oracles on running-time and solution quality

Performance Evaluation

- Number of transmission constraints

	zero		tr+ws	
Instance	Constraints	Iterations	Constraints	Iterations
case1888rte	11.7	4.07	24.0	1.00
case1951rte	2.7	2.17	7.0	1.00
case2848rte	4.6	2.57	6.7	1.03
case3012wp	9.1	3.33	21.0	1.00
case3375wp	7.0	3.17	20.0	1.00
case6468rte	20.0	5.90	32.0	1.00
case6470rte	11.5	4.13	17.0	1.00
case6495rte	17.5	5.60	28.7	1.00
case6515rte	14.1	4.27	16.9	1.00

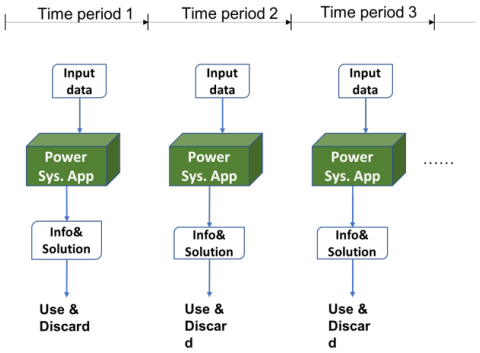
Figure: Transmission oracle performance

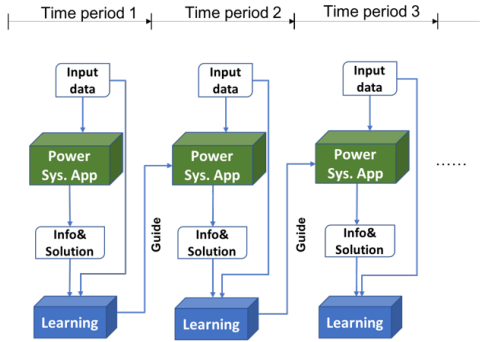
	Commitment Variables				Performance	
Instance	Total	Fix-one	Fix-zero	Free	Recall (%)	Precision (%)
case1888rte	7128.0	4180.5	1928.2	1019.3	85.4	99.6
case1951rte	9384.0	4630.8	2770.6	1982.6	78.6	99.6
case2848rte	13128.0	6672.9	4182.6	2272.6	82.4	99.7
case3012wp	12048.0	7079.5	3089.0	1879.5	84.2	99.8
case3375wp	14304.0	8356.1	3294.9	2653.1	81.2	99.7
case6468rte	31080.0	23268.2	2827.7	4984.1	83.9	99.9
case6470rte	31920.0	24317.5	2643.0	4959.4	84.4	99.9
case6495rte	32928.0	25597.3	2462.3	4868.3	85.1	99.9
case6515rte	33312.0	25471.9	2453.3	5386.8	83.8	99.9

Figure: Precision and recall of variable-fixing oracle

Learning Enabled Operations (LEO)

- Existing way to run operational applications
  - Valuable information is discarded after solve
  - Optimal solutions, solver information (e.g., branch-and-bound trees)
- Learning from daily operations to gain knowledge and intelligence for smarter operations and quicker responses to disturbances





Conclusion and Future Research

- Summary:
  - Machine-learning methods to accelerate solution of SCUC
  - 12 times speedup on large-scale, realistic instances
  - No negative impact on solution quality
  - `https://arxiv.org/abs/1902.01697`
- Take-away
  - Learning can be effective for routinely solved optimization problems
  - More general AI framework for optimization

Acknowledgement

This work is supported by Argonne National Laboratory
Laboratory Directed Research and Development (LDRD) Swift Program

Thank You