Learning to Solve Large Scale Security-Constrained Unit Commitment

Alinson Xavier, **Feng Qiu,**
Energy Systems Division
Argonne National Laboratory

Shabir Ahmend, Xiaoyi Gu,Santanu S. Dey
Industrial & Systems Engineering
Georgia Institute of Technology

The work in this presentation is based on:

Learning to solve large-scale security-constrained unit commitment problems
Álinson Xavier, Feng Qiu, Shabir Ahmed
INFORMS Journal on Computing 33 (2), 739-756

Exploiting Instance and Variable Similarity to Improve Learning-Enhanced Branching
Xiaoyi Gu, Santanu S. Dey, Feng Qiu, Alinson Axavier,
(in preparation)

A relevant presentation
10-12AM on July 19 Tuesday, Governor's square 9
Frontier of Power System Optimization and Simulation
Solving large-scale SCUC with MIPLearn+UnitCommitment.jl
Alinson Xavier

Learning to Optimize

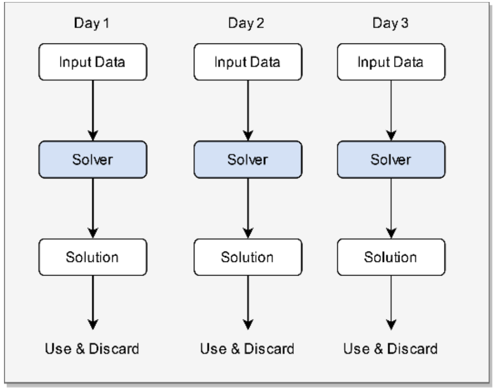
Problem in focus

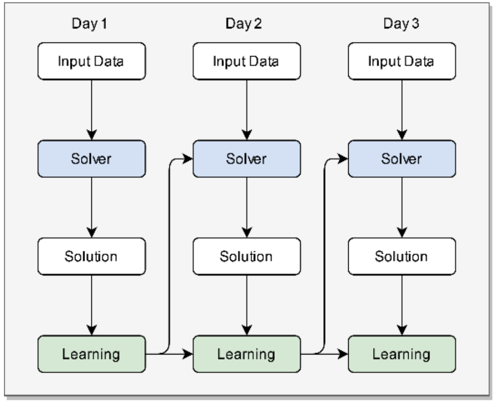
- Security constrained unit commitment (SCUC)
  - seek most cost-effective generator commitment and production output levels
  - the most fundamental mixed-integer programming problem in power systems
    - electricity market clearing
      - $400 billion annually; 0.1 optimality gap in 20 mins; often ends up with large gaps
    - reliability analysis, production cost modeling, etc.
  - Increasingly challenging
    - new energy components, e.g., combined cycle, energy storage, distributed energy resources
    - sub-hourly commitment, e.g., 15-min commitment
    - uncertainties
- Things that don't work
  - Cutting planes
  - Strong formulations
  - Decomposition

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{g \in G} c_g(x_{g\bullet}, y_{g\bullet}) && && (1)\\
\text{Subject to} \quad & (x_{g\bullet}, y_{g\bullet}) \in \mathcal{G}_g && \forall g \in G && (2)\\
& \sum_{g \in G} y_{gt} = \sum_{b \in B} d_{bt} && \forall t \in T && (3)\\
& -F_l^c \le \sum_{b \in B} \delta_{lb}^c \left(\sum_{g \in G_b} y_{gt} - d_{bt} \right) \le F_l^c && \forall c \in L \cup \{0\}, l \in L, t \in T. && (4)\\
& x_{gt} \in \{0,1\} && \forall g \in G, t \in T && (5)\\
& y_{gt} \ge 0 && \forall g \in G, t \in T && (6)
\end{aligned}
$$

Learning to Optimize

- Relevant work (prior to 2017)
    - Solve SCUC using Artificial Neural Networks:
        - Sasaki, Watanabe, Kubokawa, Yorino & Yokoyama (1992)
        - Wang & Shahidehpour (1993)
        - Walsh & O'mally (1997)
        - Liang & Kang (2000)
    - Use ML to enhance MILP solvers:
        - Alvarez, Wehenkel, Louveaux (2014)
        - Alvarez, Louveaux, Wehenkel (2017)
        - Khalil, Dilkina, Nemhauser, Ahmed, Shao (2017)
- Our perspective
    - General framework but NOT application-agnostic
    - Help solvers become progressively better over time





Learning to Optimize

- **Learning security constraints**
  - N-1 contingency criteria (transmission/security constraints) are fundamental reliability requirements enforced by NERC
  - Security constraints have large impact on performance:
    - Quadratic number, typically very dense
    - Very few are actually binding, but hard to tell in advance

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

  - **Contingency Oracle:** Predict which contingency constraints should be added to the relaxation and which should be omitted
  - **Training phase**:
    - Solve problem without any transmission constraints
    - Add small subset of most-violated constraints and resolve
    - Repeat until no further violations are found
  - **Test phase**: If constraint was necessary for at least 1% of training cases, add at start, then follow previous algorithm

Learning to Optimize

- **Learning initial feasible solutions**
  - Primal bounds are still a bottleneck
    - Modern formulations/solvers usually yield very strong dual bounds
    - Most time is spent finding high-quality primal solutions
  - Warm start: find, among large set of previous solutions, ones that are likely to work well as warm starts in MILP solvers
  - Training phase (instance-based learning):
    - Solve each training instance and store its solution
  - Test phase:
    - Find k training instances closest to the test instance
    - Use their k optimal solutions (or partial optimal solutions) as warm starts

Learning to Optimize

- **Learning affine subspaces**
  - Optimal SCUC solutions have a number of patterns:
    - Some units are operational throughout the day
    - Some units are only operational during peak demand
  - Affine subspace: Find subspaces (described by a set of hyperplanes) where the solution is very likely to reside
  - Training phase (instance-based learning):
    - Consider a fixed set of candidate hyperplanes
    - Build supervised models (e.g., SVM) to predict if hyperplane is satisfied. Discard models with low precision or recall
  - Test phase:
    - Predict which hyperplanes are likely to satisfied using previous model and add them to the relaxation

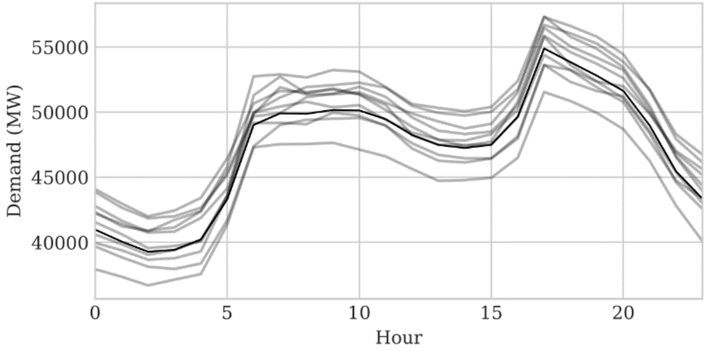
Learning to Optimize

- Learning affine subspaces
  - Hyperplanes considered:
    - $x_{gt} = 0$
    - $x_{gt} = 1$
    - $x_{gt} = x_{g,t+1}$
  - Classifier: Support Vector Machines
  - Training high-quality models:
    - Discard hyperplanes with very unbalanced labels
    - Measure precision and recall using k-fold cross validation
    - Discard models with low recall or precision

	Commitment Variables					
Instance	Total	Fix Zero	Fix One	Fix Next	Free	Precision
case1888rte	7128.0	17.9%	52.0%	24.7%	5.4%	99.8%
case1951rte	9384.0	20.0%	45.9%	27.8%	6.3%	99.8%
case2848rte	13128.0	23.9%	44.8%	23.2%	8.2%	99.7%
case3012wp	12048.0	18.5%	54.5%	21.3%	5.6%	99.8%
case3375wp	14304.0	14.4%	57.8%	22.4%	5.5%	99.8%
case6468rte	31080.0	7.6%	72.4%	12.8%	7.2%	99.9%
case6470rte	31920.0	7.3%	70.3%	16.2%	6.2%	99.8%
case6495rte	32928.0	5.8%	75.4%	13.1%	5.7%	99.9%
case6515rte	33312.0	6.1%	75.5%	12.0%	6.4%	99.9%
Average	20581.3	13.5%	60.9%	19.3%	6.3%	99.8%

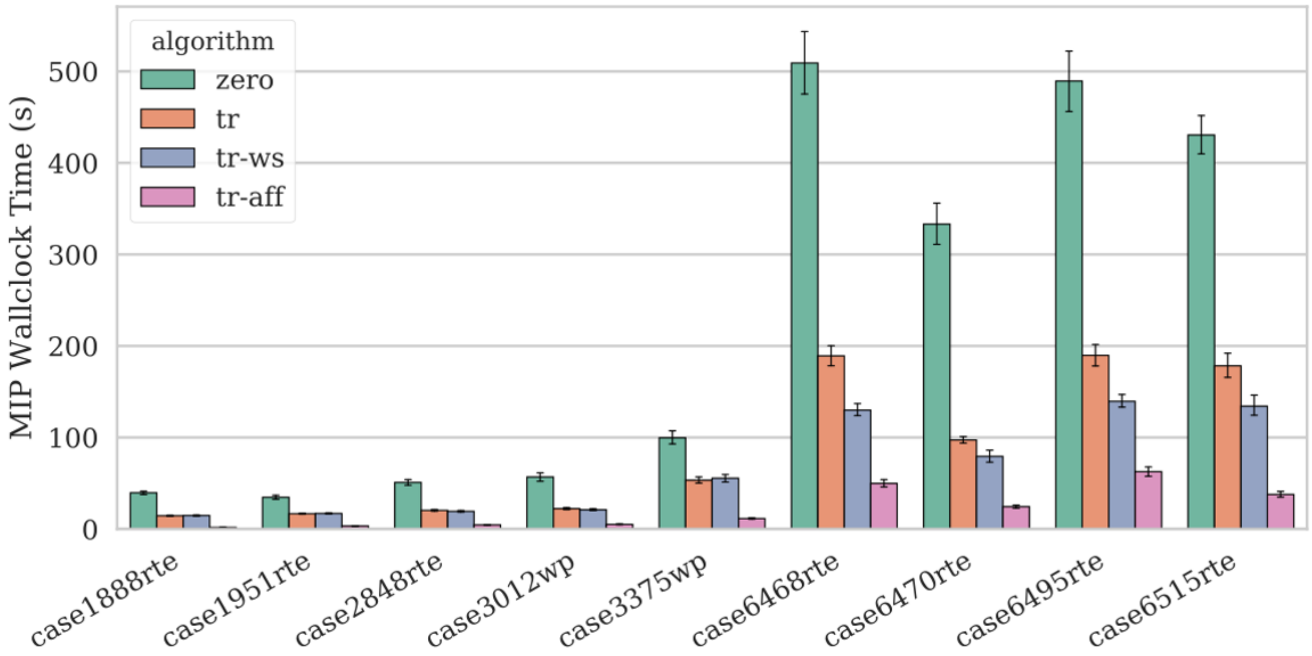
- **Computational results**
- **SCUC:** Find cost-efficient power generation schedule, subject to:
  - Production during each hour must satisfy demand
  - Power flows must be within safe limits
  - Other physical, operational & economic constraints
- **Widely used in planning and operations:**
  - Day-ahead electricity markets, reliability assessment
- **Benchmark set:** 9 realistic, large-scale cases from MATPOWER
- **Training instances:** 300 random variations
- **Test instances:** 50 random variations
- **Randomized parameters:**
  - Peak system-wide load
  - Production and start-up costs
  - Geographic load distribution
  - Temporal load profile

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



Learning to Optimize

- **Computational results**
  - **The best record for solving large-scale SCUC**



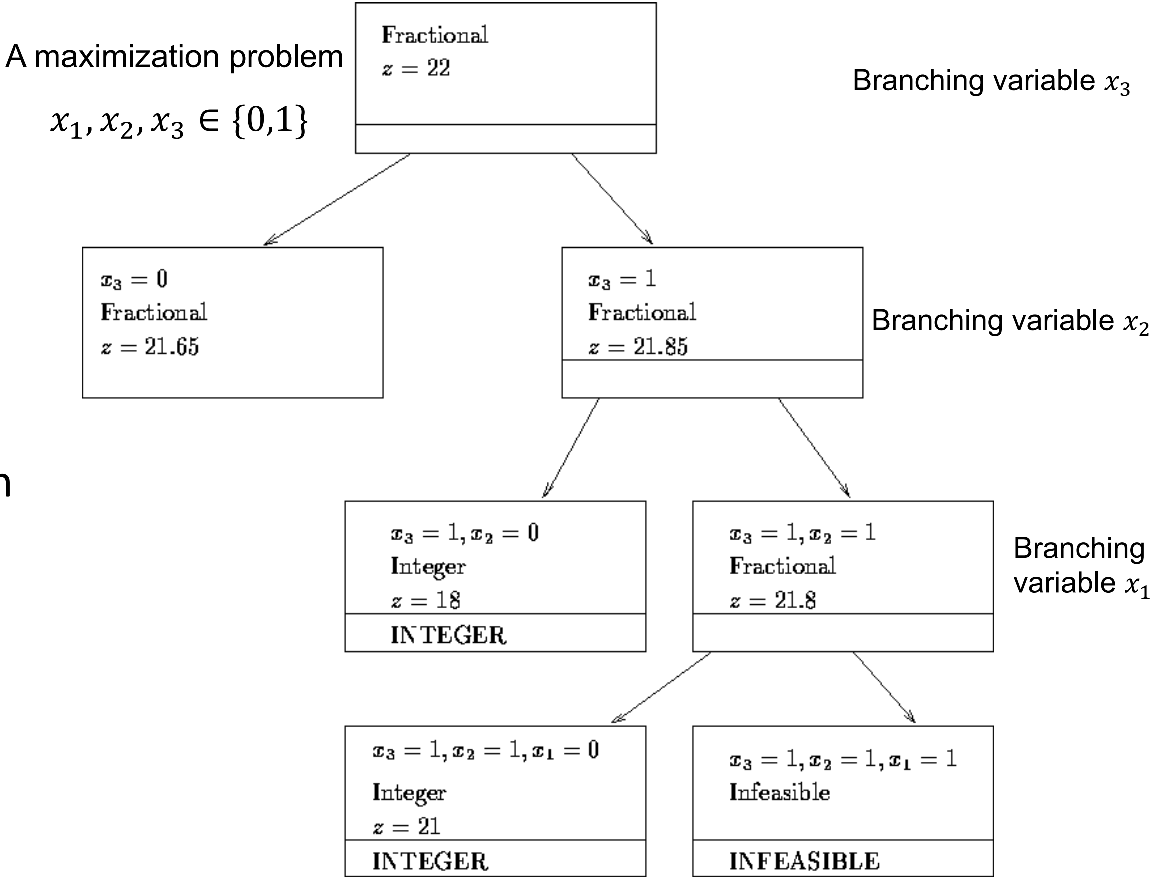
Learning to Solve Large-Scale Unit Commitment (Xavier, Qiu, Ahmed (2020))

Learning to Optimize

- **MIPLearn**
  - Flexible, extensible, and easy-to-use open-source framework for learning-enhanced integer programming
  - MIPLearn components
    - Initial feasible solutions
    - Lazy constraints and user cuts
    - Branching priorities
    - Optimal value
  - Modeling languages: JuMP, Pyomo, Gurobi Python API
  - Compatible MIP solvers:
    - Commercial: Gurobi, CPLEX, XPRESS
    - Non-commercial: SCIP, Cbc
  - Repository:
    - https://github.com/ANL-CEEESA/MIPLearn
    - **Alinson S. Xavier, Feng Qiu.** *MIPLearn: An Extensible Framework for Learning-Enhanced Optimization*. Zenodo (2020). DOI: 10.5281/zenodo.428756
    - License: Open source (3-clause BSD)

Branch & Bound

- A schema that exhaustively search a solution space in a mixed-integer programming problem
- Combining with bounding techniques, it provides a solution with an optimality gap
- Market transparent and fairness
- A better branching strategy can help B&B convergence



https://mat.tepper.cmu.edu/orclass/integer/node13.html

Branch

- Two decisions in branching: node selection and variable selection
- Node selection: best known strategy: always choose the nodes with best lower bounds
- Variable selection:
  - Most infeasible (fractional) branching (MIB): cheap but worst
    $$S_{\{MIB\}}(i, I) = \min\{x_i, 1 - x_i\}$$
  - Strong branching: best (smallest b&b tree) but expensive
    $$S_{\{MIB\}}(i, I) = \max\{\widetilde{\Delta}_i^-, \epsilon\} * \max\{\widetilde{\Delta}_i^+, \epsilon\}$$
    $\widetilde{\Delta}_i^-$: objective value change when branch down
  - Reliability branching: a light version of strong branching (RB: λ: η)
    (1) at most λ variables will be probed at a node
    (2) for a given variable, η number of probes are deemed sufficient

Benchmark ML-enhaced branching

- Alvarez, Alejandro Marcos and Louveaux, Quentin and Wehenkel, Louis (2017), A machine learning-based approximation of strong branching, INFORMS Journal On Computing, 29(1):185–195
- Key idea: use machine learning to mimic strong branching; a universal model for all MIPs
- Features
  - Static problem features
    - Computed from c, A, b
  - Dynamic problem features
    - The solution of the problem at the current B&B node
    - E.g., up and down fractionalities of a variable
  - Dynamic optimization features
    - Overall state of the optimization
    - E.g., statistic features of objective value changes regarding a certain variable

Exploit instance similarity in routinely solved optimizazation

- Motivation
  - Most industry applications are routinely solved optimization problems sharing high similarity, e.g., constraint matrix, right-hand sides
  - Dedicated ML models for a routinely solved optimization problem should work better than a generic ML model
- Revised learning to branch approach
  - Per variable: each variable has its own ML model
  - Per group: a group of relevant variables share a ML model
    - Per generator: time index is ignored; e.g., is_on[g,-] v.s. is_on[g,t])
    - Per time: generator index is ignored; e.g., is_on[,t] v.s. is_on[g,t])
    - Per type: time and generator indicies are ignored; e.g., is_on v.s. is_on[g,t]

Experiment setup

- Home-made branch&bound MIP solver, using Gurobi for solving LPs
- ML model: Extremely Randomized Trees (ExtraTree)
- 5 realistic power systems, 24-hour SCUC

Network	Hours	Generators	Buses	Lines	Variables	Rows	Binaries
case1888rte	24	296	1,888	2,531	235,591	196,783	41,232
case1951rte	24	390	1,951	2,596	266,088	244,220	54,144
case2848rte	24	544	2,848	3,776	377,760	340,228	71,904
case3012wp	24	496	3,012	3,572	357,146	305,076	59,712
case3375wp	24	590	3,374	4,161	413,161	357,065	71,856

- $Rb{:}\,100{:}\,inf$ (practical strong branching) used to collect data

Branching sore prediction experiments

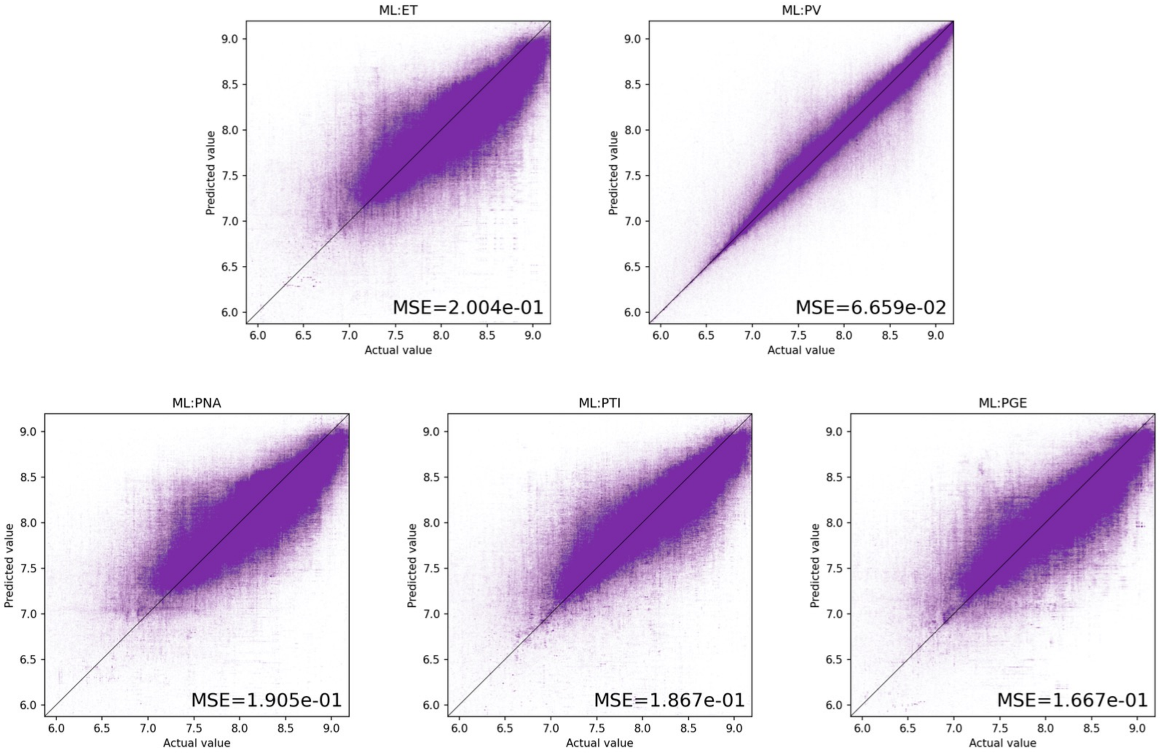


Figure 1 Cross-Validation Evaluation (case1888rte, 24h, value after logarithm) with MSE

Impact on solving SCUC with B&B

Table 4 **Relative MIP Gap**, `node_limit=1000`

Instances		Relative MIP gap (%)						
Hours	Network	MIB	RB:100:8	ML:ET	ML:PNA	ML:PTI	ML:PGE	ML:PV
24	case1888rte	1.78	0.96	1.31	1.35	0.90	1.22	0.88
24	case1951rte	0.41	0.20	0.23	0.24	0.20	0.22	0.20
24	case2848rte	0.83	0.42	0.54	0.59	0.45	0.58	0.41
24	case3012wp	0.29	0.02	0.08	0.01	0.02	0.01	0.04
24	case3375wp	0.48	0.12	0.52	0.46	0.50	0.41	0.48
Average		0.76	0.35	0.54	0.53	0.41	0.49	0.40

Impact on solving SCUC with B&B

Table 5 **Relative MIP Gap,** `node_limit=10000`

Instances		Relative MIP gap (%)					
Hours	Network	MIB	ML:ET	ML:PNA	ML:PTI	ML:PGE	ML:PV
24	case1888rte	1.75	1.13	1.20	0.66	1.03	0.61
24	case1951rte	0.37	0.11	0.11	0.08	0.11	0.08
24	case2848rte	0.77	0.43	0.47	0.33	0.46	0.30
24	case3012wp	0.27	0.05	0.01	0.01	0.01	0.02
24	case3375wp	0.45	0.51	0.42	0.47	0.31	0.44
Average		0.72	0.45	0.44	0.31	0.39	0.29

* Similar performance can be observed in pre-solved instances

Learning to Optimize

- **Future work**
  - LP relaxation
    - Large-scale MIPs
  - Cut generation
    - Find valid and useful cuts
    - Generate cuts

ACKNOWLEDGEMENT

We appreciate the funding support from Argonne LDRD program and the Advanced Grid Modeling program (AGM) under DOE Office of Electricity

THANK YOU!

Contact

Feng Qiu
Principal Computational Scientist & Group Manager
Email: fqiu@anl.gov