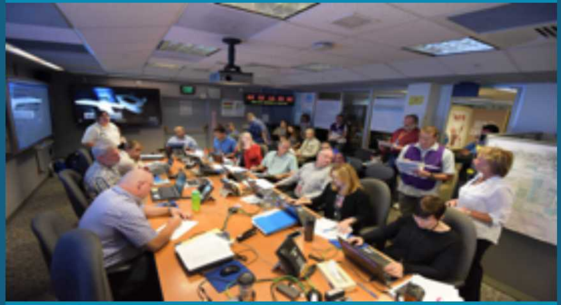
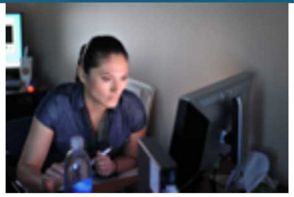
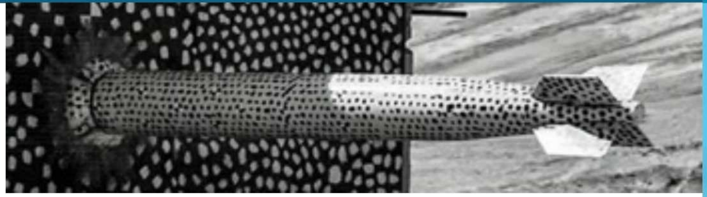
SAND2019-10499PE

# EGRET: Unit Commitment and {Economic,…} Dispatch







*PRESENTED BY*

Bernard Knueven and Jean-Paul Watson

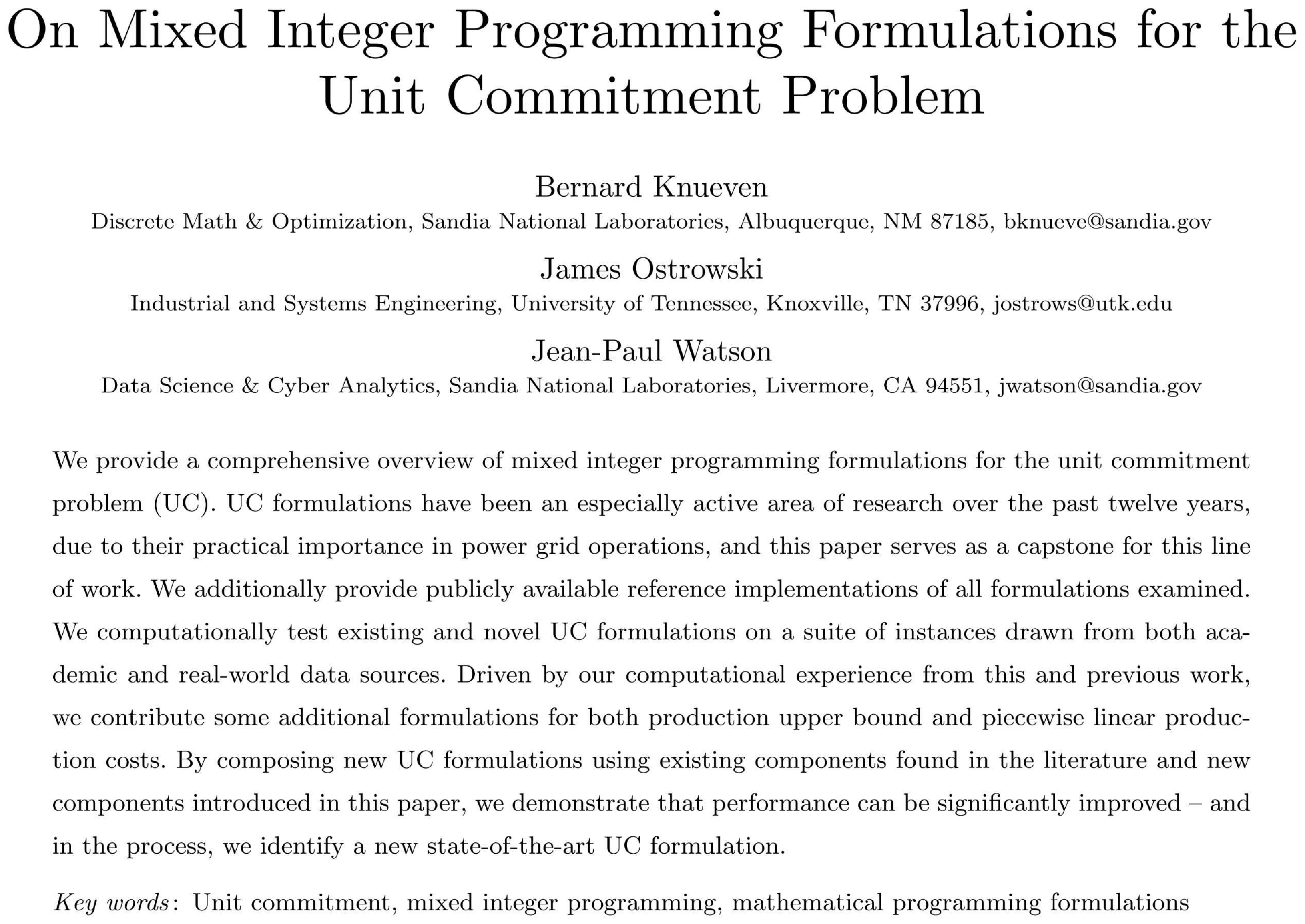
Co-authors: James Ostrowski (UTK)

Michael Bynum, Anya Castillo, Carl Laird (SNL)

## Talk Overview

## Some Preliminaries…

## Key Paper (IOHO)



# On Mixed Integer Programming Formulations for the Unit Commitment Problem

Bernard Knueven
Discrete Math & Optimization, Sandia National Laboratories, Albuquerque, NM 87185, bknueve@sandia.gov

James Ostrowski
Industrial and Systems Engineering, University of Tennessee, Knoxville, TN 37996, jostrows@utk.edu

Jean-Paul Watson
Data Science & Cyber Analytics, Sandia National Laboratories, Livermore, CA 94551, jwatson@sandia.gov

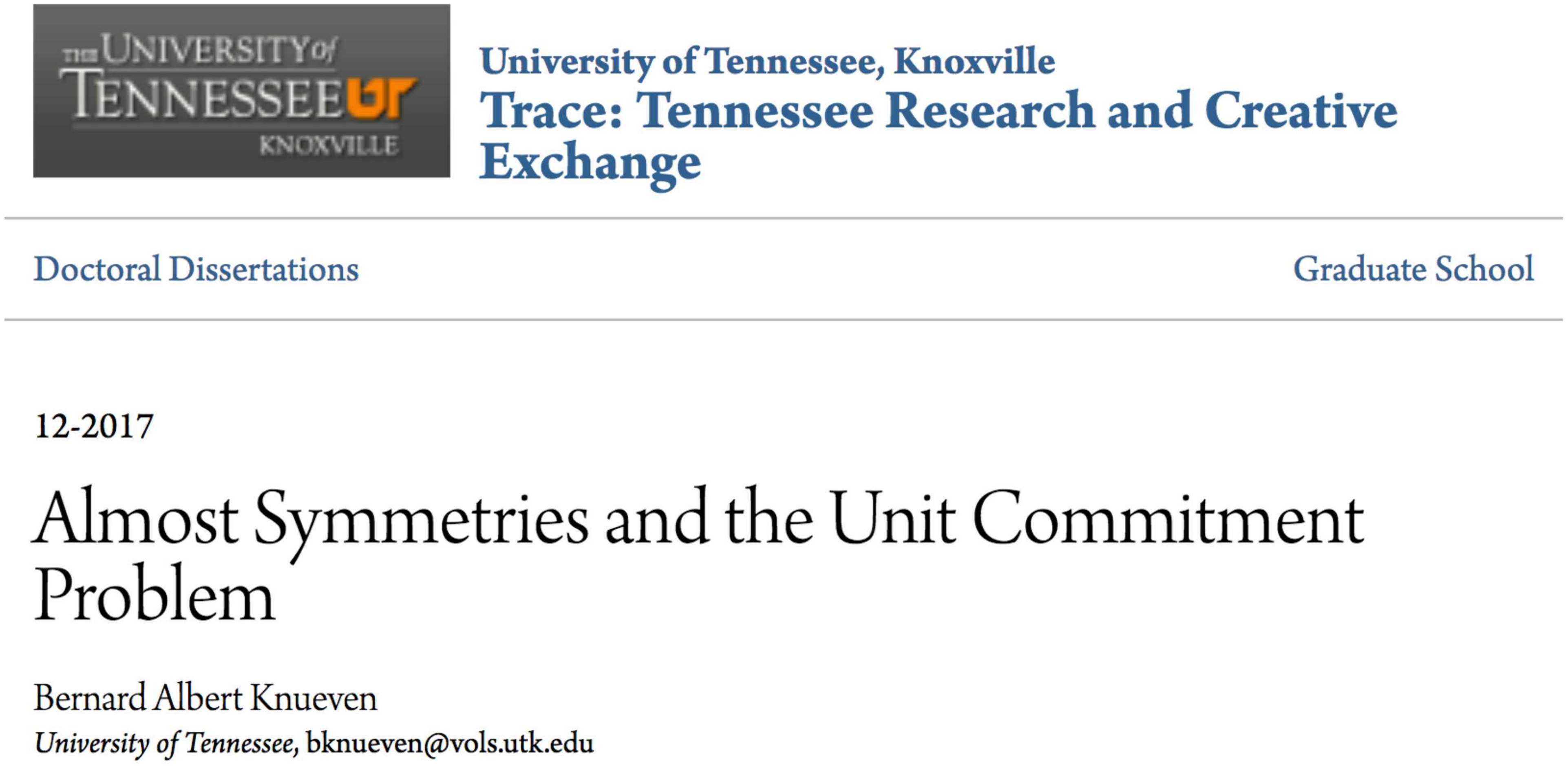
We provide a comprehensive overview of mixed integer programming formulations for the unit commitment problem (UC). UC formulations have been an especially active area of research over the past twelve years, due to their practical importance in power grid operations, and this paper serves as a capstone for this line of work. We additionally provide publicly available reference implementations of all formulations examined. We computationally test existing and novel UC formulations on a suite of instances drawn from both academic and real-world data sources. Driven by our computational experience from this and previous work, we contribute some additional formulations for both production upper bound and piecewise linear production costs. By composing new UC formulations using existing components found in the literature and new components introduced in this paper, we demonstrate that performance can be significantly improved – and in the process, we identify a new state-of-the-art UC formulation.

*Key words*: Unit commitment, mixed integer programming, mathematical programming formulations

Available on *Optimization Online*: [http://www.optimization-online.org/DB_HTML/2018/11/6930.html](http://www.optimization-online.org/DB_HTML/2018/11/6930.html)

Under revision with the INFORMS Journal on Computing

## Another Key "Paper"



12-2017

Almost Symmetries and the Unit Commitment Problem

Bernard Albert Knueven
*University of Tennessee*, bknueven@vols.utk.edu

## EGRET: Electrical Grid Research and Engineering Tools

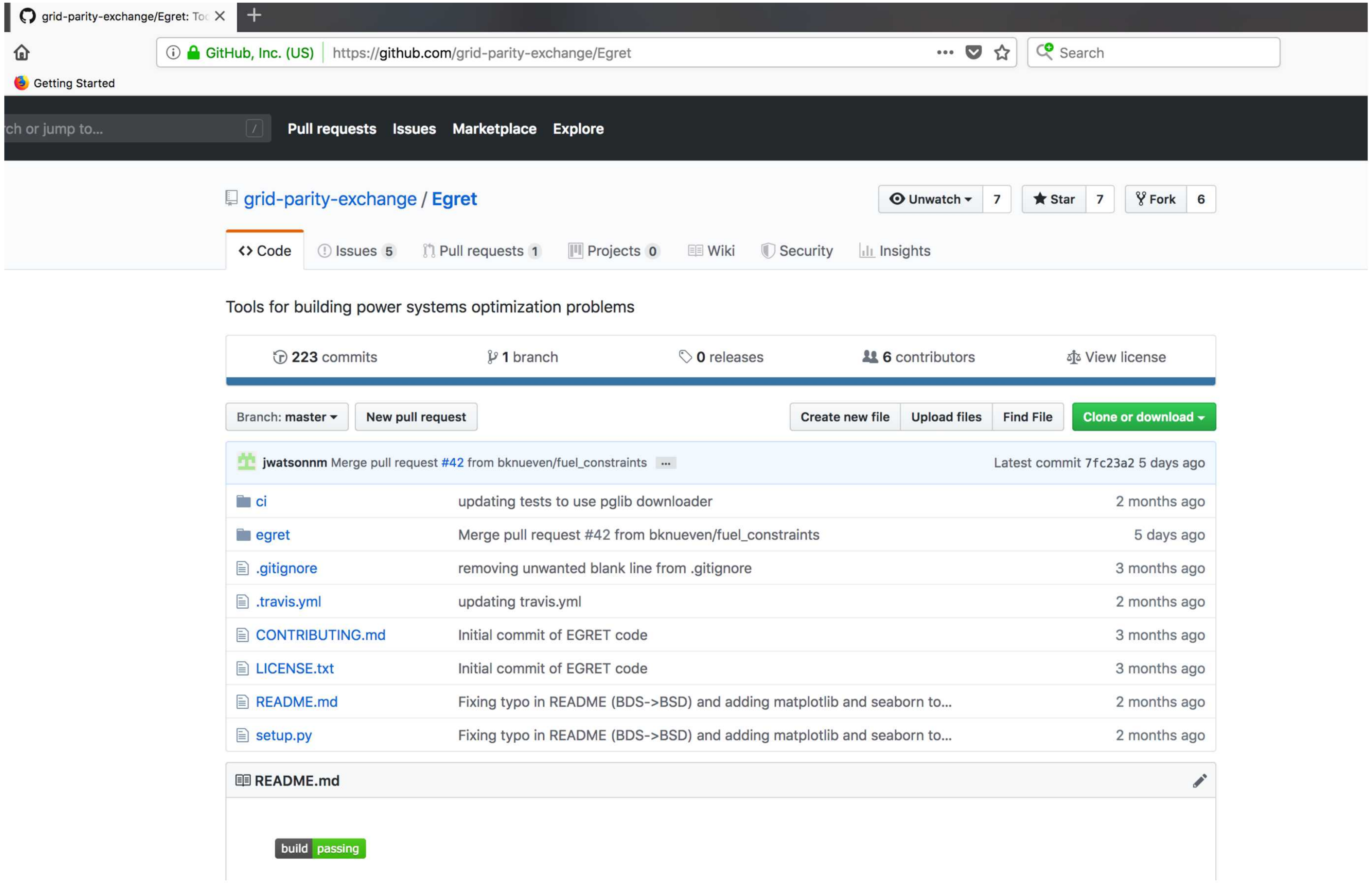
EGRET is a Python-based package for electrical grid optimization based on the Pyomo optimization modeling language. EGRET is designed to be friendly for performing high-level analysis (e.g., as an engine for solving different optimization formulations), while also providing flexibility for researchers to rapidly explore new optimization formulations.

Major features:

- Expression and solution of unit commitment problems, including full ancillary service stack
- Expression and solution of economic dispatch (optimal power flow) problems (e.g, DCOPF, ACOPF)
- Library of different problem formulations and approximations
- Generic handling of data across model formulations
- Declarative model representation to support formulation development

EGRET is available under the BSD License at https://github.com/grid-parity-exchange/Egret

## EGRET: GitHub Home Page (Existence Proof)


grid-parity-exchange/Egret: To

GitHub, Inc. (US) https://github.com/grid-parity-exchange/Egret

Getting Started

Pull requests Issues Marketplace Explore

grid-parity-exchange / Egret

Unwatch 7 Star 7 Fork 6

Code Issues 5 Pull requests 1 Projects 0 Wiki Security Insights

Tools for building power systems optimization problems

223 commits 1 branch 0 releases 6 contributors View license

Branch: master New pull request Create new file Upload files Find File Clone or download

jwatsonnm Merge pull request #42 from bknueven/fuel_constraints ... Latest commit 7fc23a2 5 days ago

| | | |
|---|---|---|
| ci | updating tests to use pglib downloader | 2 months ago |
| egret | Merge pull request #42 from bknueven/fuel_constraints | 5 days ago |
| .gitignore | removing unwanted blank line from .gitignore | 3 months ago |
| .travis.yml | updating travis.yml | 2 months ago |
| CONTRIBUTING.md | Initial commit of EGRET code | 3 months ago |
| LICENSE.txt | Initial commit of EGRET code | 3 months ago |
| README.md | Fixing typo in README (BDS->BSD) and adding matplotlib and seaborn to... | 2 months ago |
| setup.py | Fixing typo in README (BDS->BSD) and adding matplotlib and seaborn to... | 2 months ago |

README.md

build passing

## EGRET Prerequisites

Python >= 3.6

- Technically might still work with Python 2.7.X, but no promises

Pyomo

- www.pyomo.org
- Version >= 5.6
- Use ”conda install” or ”pip install”

Solver

- Pyomo is an Algebraic Modeling Language (AML)
- EGRET expresses unit commitment and dispatch models using the Pyomo AML
- Need a mixed-integer linear (commitment) and linear (dispatch) solver
  - Commercial: Gurobi, CPLEX, Xpress
  - Academic: CBC, GLPK
  - In Between: SCIP

## EGRET: Commentary

Development of the commitment and dispatch models in EGRET started in 2012
- Funded by an ARPA-E GENI project (scalable stochastic unit commitment)

Predecessor models were validated against models in Alstom's Market Management System
- 2 months on-site effort at Alstom

B. Knueven (SNL) led effort to library-ize and extend these commitment and dispatch models
- Subsequent slides provide much detail

C. Laird, M. Bynum, and A. Castillo (SNL) have led ACOPF-related efforts in EGRET
- Not the topic of today's talk – and somewhat less mature than commitment/dispatch components

Documentation is lagging capability by a significant margin
- All function, no flash – yes, we are terrible at PR

Still migrating advanced models and capabilities into EGRET
- E.g., from ASU/Sandia/Nexant ARPA-E NODES project

Working with toward IEEE benchmark unit commitment cases
- With Carleton Coffrin (LANL)
- IEEE will host the instance data
- EGRET will provide reference model implementations

## Commitment vs. Dispatch: A Note

In our view, there is no real difference between commitment and dispatch

- This is reflected in the design of EGRET
- From the standpoint of mathematical optimization modeling

Commitment has binary variables representing unit on/off status

- Fix the binaries, and you get a dispatch model

Network representations can be used in conjunction with either commitment and dispatch models

- B-theta or PTDF

Multi-period versus single-period

- PCMs often run single-period dispatch
- ISOs usually run multi-period dispatch

## The Unit Commitment Model “Zoo”…

## Our Recent Contributions in Commitment / Dispatch

We catalog existing formulations for the UC problem as originally described by Carrion and Arroyo (2006).

- Improvements to this formulation have been the subject of several subsequent papers, including Ostrowski et al. (2012), Morales-Espana et al. (2013), Damci-Kurt et al. (2016), Pan et al. (2016), K. et al. (2018), K. et al. (2018), Atakan et al. (2018).

We preform computational experiments on 41 different UC formulations, some novel, and some from the literature, on 68 UC instances.

- Largest instance has 900+ generators over 48 hours with hourly time horizon
- This took approximately two weeks of wall-clock time!

We make publically available on GitHub reference implementations for all the formulations examined in the Pyomo modeling language in the EGRET library. We will also make publically available the UC instances considered.

We introduce two novel UC formulations, one of which is a new combination of existing components, and the other draws on on new components as well as existing formulations. The later formulation significantly improves on the performance of any previously reported UC formulation, establishing a new state-of-the-art.

## The Unit Commitment Problem (1)

The Unit Commitment Problem (UC) is a large-scale mixed-integer nonlinear program for finding a low-cost operating schedule for power generators.

These problems typically have quadratic objective functions and non-linear, non-convex transmission constraints.

- Typically both of these are linearized

Starting in 2005 with PJM, market operators in the United States have transitioned from using Lagrangian relaxation to solve UC to using mixed-integer programming (MIP) and a commercial solver such as CPLEX, Gurobi, or Xpress.

- MIPs usually have many equivalent formulations, and UC is no exception.

The day-ahead problem has an hourly time horizon which is solved for 36 to 48 hours ahead to prevent end-of-horizon effects, and has hundreds to thousands of generators and up to tens of thousands of buses.

In practice, it is desirable to have a UC solution in 10 to 15 minutes.

## The Unit Commitment Problem (2)

$$
\begin{aligned}
\min \sum_{g\in\mathcal{G}} \sum_{t\in\mathcal{T}} & c_g(t) \\
\text{s.t.} \ \sum_{g\in\mathcal{G}} & A_g(p_g, \overline{p}_g, u_g) + N(s) = L \\
& (p_g, \overline{p}_g, u_g, c_g) \in \Pi_g \qquad \forall g \in \mathcal{G}.
\end{aligned}
$$

UC is that of minimizing system operating costs subject to the system constraints and the technical constraints of the generators.

Generator technical constraints

- Convex (piecewise linear) production costs
- Minimum and maximum output levels
- Ramping constraints
- Minimum up/down time
- Downtime dependent startup costs

## Polyhedral Results for Generator Scheduling (1)

1-binary variable model (1-bin)

- We can write the feasible region of a generator using two variables per time period.
- $p(t)$ is the continuous variable representing the power output at time $t$.
- $u(t)$ is the binary variable representing if the generator is on/off.
- There is a known convex hull description for this polyhedron with simple bounds on output and minimum up/down times, but it is vary large (exponential).
- But, a polynomial time cutting-plane method exists (Lee et al. 2004).

3-binary variable model (3-bin)

- Add to the 1-bin model two additional variables:
- $v(t)$ is the binary variable representing a *turn on* at time $t$,
- $w(t)$ is the binary variable representing a *turn off* at time $t$.
- The two additional variables are redundant. But, they allow us to write tight descriptions of the generator polytope with minimum up/down times (Rajan and Takriti 2005), start-up and shutdown power constraints (Gentile et al. 2017), and convex piecewise production costs (K. et al. 2018) (with additional variables for each piecewise segment) with a linear number of constraints and variables.

## Polyhedral Results for Generator Scheduling (2)

Shortest path formulation

- Add additional variables $y(t_1, t_2)$ to represent at start-up at time $t_1$ and shutdown at time $t_2$, on continuously in between, and additional variables $\mathrm{z}(t_1, t_2)$ to represent at shutdown at time $t_1$ and start-up at time $t_2$, off continuously in between.
- These start-up/shutdown sequences can be linked using a full shortest-path formulation (Pochet and Wolsey, 2006), where the path is from "turn on" nodes to "turn off" and from "turn off" nodes to "turn on" nodes.
- This formulation has $O(T^2)$ edges (variables), but provides a convex hull description for downtime dependent start-up costs. There is a clear link to the $u, v, w$ variables, so the prior results on start-up/shutdown power and piecewise production costs carry through (note the edges themselves can enforce minimum up/down time).
- If we are okay with integer optimal, we can use the "matching" formulation from K. et al. (2018), which keeps the 3-bin variables and only those $z$'s which represent a hot/warm start-up. The benefit is far fewer variables (on order $(TC - DT) \cdot T$).

## Polyhedral Results for Generator Scheduling (3)

Extended formulation

- To the shortest path formulation, add additional variables $p(t, t_1, t_2)$ for the output of the generator at time $t$ given a start-up and time $t_1$ and a shutdown at time $t_2$, on continuously in between.
- Requires on order $T^3$ variables and constraints, but gives a convex hull representation for ramping constraints, and every other technical constraint mentioned (Frangioni and Gentile (2015), K. et al. (2018), Guan et al. (2018)).
- Very large but still polynomial. K. et al. (2018) uses it for cut-generation.

## Choosing a UC Formulation

To summarize:

- 1-bin formulation (smallest): Not a convex hull for any interesting phenomenon
- 3-bin formulation (small): Convex hull for minimum up/down times, convex piecewise production, and start-up/shutdown power
- Shortest path (large): Convex hull for above plus downtime-dependent start-up costs, smaller if integer optimal is sufficient
- Extended formulation (very large): Convex hull description for everything above plus ramping constraints, and more!

Computational experience to date indicates

- 1-bin: too weak
- EF: too large
- 3-bin: just right

Since the convex hull representation for a single generator is too large, it is important to consider which classes of (perhaps imperfect) constraints we are going to include in a practical formulation.

In this case, engineering is more important than math.

## A Modular Framework for UC Formulations in EGRET

```
from egret.model_library.unit_commitment.uc_model_generator \
                import UCFormulation, generate_model

## get the formulation from Carrion and Arroyo (2006)
formulation = UCFormulation(status_vars = 'CA_1bin_vars',
                            power_vars = 'basic_power_vars',
                            reserve_vars = 'CA_power_avail_vars',
                            generation_limits = 'CA_generation_limits',
                            ramping_limits = 'CA_ramping_limits',
                            production_cost = 'CA_production_costs',
                            uptime_downtime = 'CA_UT_DT',
                            startup_costs = 'CA_startup_costs',
                            )

## construct the model based on the data md
model = generate_model(md, formulation)
```

**Over 100,000 formulations**

This instantiates a Pyomo `ConcreteModel` (`model`) based on the data provided in the object `md`, which can be used as part of a script.

The eights components of `UCFormulation` can be changed as easily as modifying a string in this file. Runtime checks to ensure incompatible components are not combined.

Number of implemented formulations per component:

- `status_vars`: 5
- `power_vars`: 3
- `reserve_vars`: 4
- `generation_limits`: 9
- `ramping_limits`: 8
- `production_costs`: 12
- `uptime_downtime`: 5
- `startup_costs`: 9

## Test Instances

One academic test set and and two test sets based on real-world data. All are an hourly 48-hour day-ahead UC.

- RTS-GMLC: 73 thermal generators, 81 renewable generators, 73 buses, 120 transmission lines. Hourly day-ahead data for load and renewable generation for a year. We selected twelve representative days, considered both with and without transmission for a total of 24 test instances.
- CAISO: 410 schedulable thermal generators, 200 must-run thermal generators. We considered five demand/renewables scenarios under four reserve policies: 0%, 1%, 3%, and 5% of demand, for a total of 20 test instances.
- FERC/PJM: Generation set publically available from FERC, approximately 900 thermal units, with demand, reserve, and wind data publically available from PJM for 2015. We selected twelve representative days, and considered the wind data as-is (low-wind) and also scaled it to achieve 30% wind penetration for the year, for 24 test instances total.

The above makes for a total of 68 UC instances across three set of generators.

## Some Formulations Considered

**CA:** Carrion and Arroyo (2006)

**OAV-O:** Ostrowski et al. (2012) "Original," similar to Arroyo and Conejo (2000)

**OAV-UD:** Ostrowski et al. (2012) "Up/Downtime"

**OAV:** Ostrowski et al. (2012) with 2-period consecutive ramping inequalities from this paper

**OAV-T:** Ostrowski et al. (2012) with all ramping and generalized upper bound inequalities from this paper

**MLR:** Morales-Espana et al. (2013)

**ALS:** Atakan et al. (2018)

**KOW:** K. et al. (2018)

**T:** "Tight" New formulation using some ideas from the literature and the generalized upper bounds and production costs introduced in this paper

**Co:** "Compact" New formulation using ideas from the literature that emphasize compactness

**R1:** "Random 1" New formulation based on sampling formulations and testing against the RTS-GMLC instances

**R2:** "Random 2" Another new formulation based on sampling and testing against the RTS-GMLC instances

## Computational Platform

Dell PowerEdge T620 (circa 2013)

- Two Intel Xeon E5-2670 processors (16 cores/32 threads)
- 256GB RAM
- Ubuntu 16.04
- Gurobi 8.0.1

No other major jobs were running at the time of the computational experiments, and Gurobi settings were preserved at default except a time limit.

- Hence all UC MIPs we attempt to solve to 0.01% optimality gap.

The time limit was set at 300 seconds for the RTS-GMLC and CAISO instances, and a time limit of 600 seconds was imposed for the larger FERC instances.

## Computational Results: RTS-GMLC Instances

| Formulation | Time (s) | Opt gap (%) | Time outs | Times best | Times 2nd |
|---|---|---|---|---|---|
| **CA** | 300.0 | 13.806% | 24 | 0 | 0 |
| **OAV-O** | 172.8 | 0.3335% | 13 | 0 | 0 |
| **OAV-UD** | 163.2 | 0.2541% | 13 | 0 | 0 |
| **OAV** | 169.7 | 0.2352% | 12 | 0 | 0 |
| **OAV-T** | 178.0 | 0.2498% | 12 | 0 | 0 |
| **MLR** | 86.19 | 0.0165% | 5 | 0 | 0 |
| **ALS** | 58.29 | 0.0122% | 2 | 1 | 1 |
| **KOW** | 94.81 | 0.0165% | 4 | 0 | 1 |
| **T** | 50.65 | 0.0121% | 2 | 4 | 4 |
| **Co** | 43.11 | 0.0121% | **1** | 1 | 4 |
| **R1** | **32.37** | **0.0114%** | **1** | **14** | 6 |
| **R2** | 36.94 | 0.0114% | **1** | 4 | **8** |

The **R1** and **R2** formulations do well on this test set.

- Not too surprising since they were selected based on their performance on this test set.

## Computational Results: CAISO Instances

| Formulation | Time (s) | Opt gap (%) | Time outs | Times best | Times 2nd |
|---|---|---|---|---|---|
| **CA** | 300.0 | 1.0987% | 20 | 0 | 0 |
| **OAV-O** | 300.0 | 6.7647% | 20 | 0 | 0 |
| **OAV-UD** | 300.0 | 6.2269% | 20 | 0 | 0 |
| **OAV** | 300.0 | 3.6319% | 20 | 0 | 0 |
| **OAV-T** | 300.0 | 6.1679% | 20 | 0 | 0 |
| **MLR** | 117.1 | 0.0119% | 2 | 0 | 1 |
| **ALS** | 260.4 | 0.0577% | 12 | 0 | 0 |
| **KOW** | 79.02 | 0.0102% | 2 | 4 | **7** |
| **T** | **56.90** | **0.0100%** | **0** | **15** | 3 |
| **Co** | 100.6 | **0.0100%** | **0** | 0 | 5 |
| **R1** | 147.4 | 0.0102% | 1 | 0 | 1 |
| **R2** | 104.1 | 0.0100% | 1 | 1 | 3 |

Here the **T**, **Co**, **R2**, and **KOW** formulations perform well.

## Computational Results: FERC Instances

| Formulation | Time (s) | Opt gap (%) | Time outs | Times best | Times 2nd |
|---|---|---|---|---|---|
| **CA** | 600.0 | 43.333% | 24 | 0 | 0 |
| **OAV-O** | 599.7 | 11.716% | 23 | 0 | 0 |
| **OAV-UD** | 588.2 | 1.4575% | 20 | 0 | 0 |
| **OAV** | 588.4 | 1.3104% | 21 | 0 | 0 |
| **OAV-T** | 582.4 | 1.4389% | 22 | 0 | 0 |
| **MLR** | 340.5 | 0.0555% | 3 | 4 | 1 |
| **ALS** | 394.7 | 0.0933% | 7 | 2 | 2 |
| **KOW** | 390.2 | 0.0117% | 2 | 3 | 5 |
| **T** | **268.6** | **0.0104%** | **1** | **8** | 4 |
| **Co** | 309.5 | 0.0596% | 3 | 4 | 5 |
| **R1** | 308.9 | 0.0480% | 3 | 3 | **6** |
| **R2** | 373.1 | 0.0665% | 4 | 0 | 1 |

**T** is clearly the superior performer on this set of instances.

## Summary

Formulation **T** performs well across the test set.

- Weaker performance on the RTS-GMLC is mainly due to the network-constrained instances, which require quite a bit of enumeration. We used a $B$-$\theta$ representation for the network, for which Gurobi has a difficult time generating valid cuts.
- Typically operators use a PTDF formation for the network, which will have different behavior.
- Even in the worst case across the 68 instances, with formulation **T** Gurobi terminates at the time limit with only a 0.05% gap, which is the best worst-case performance.

Formulations **R1** and **R2** both performed well on RTS-GMLC, suggesting it may be possible to specialize the formulation used against typical instances.

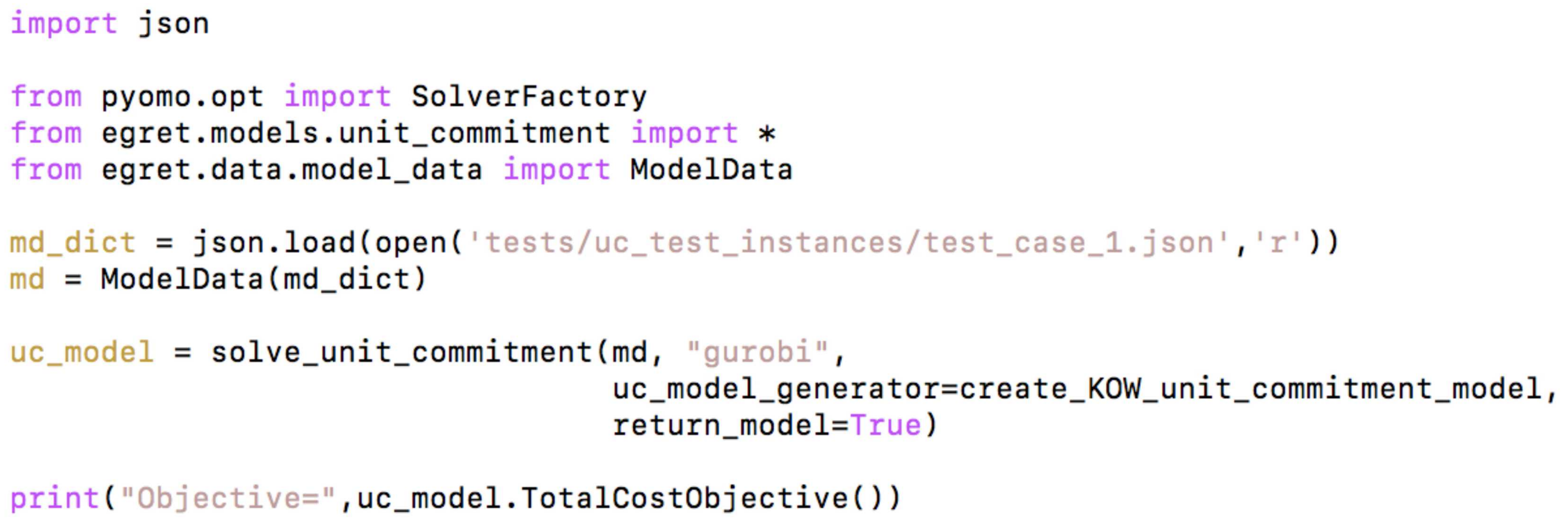
- Since the generation fleet often does not change, it may be possible to "fit" a formulation for typical or difficult day-ahead instances.

## Some Code Examples…

## Example Code – Constructing and Solving a UC Model

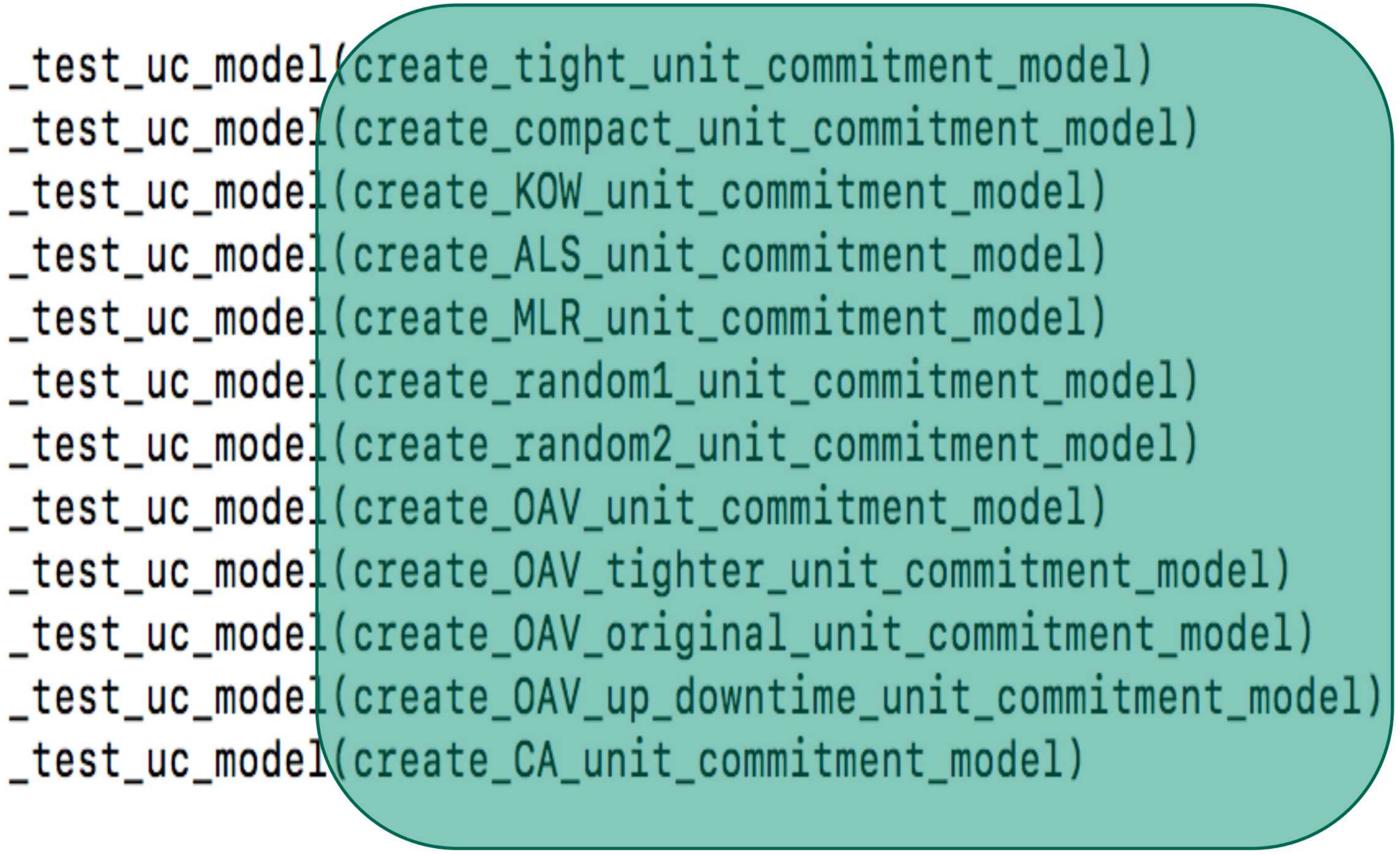
```python
import json

from pyomo.opt import SolverFactory
from egret.models.unit_commitment import *
from egret.data.model_data import ModelData

md_dict = json.load(open('tests/uc_test_instances/test_case_1.json','r'))
md = ModelData(md_dict)

uc_model = create_KOW_unit_commitment_model(md)

opt = SolverFactory('gurobi')
opt.solve(uc_model)

print("Objective=",uc_model.TotalCostObjective())
```

## If That Was Too Much Code…



```
import json

from pyomo.opt import SolverFactory
from egret.models.unit_commitment import *
from egret.data.model_data import ModelData

md_dict = json.load(open('tests/uc_test_instances/test_case_1.json','r'))
md = ModelData(md_dict)

uc_model = solve_unit_commitment(md, "gurobi",
                                 uc_model_generator=create_KOW_unit_commitment_model,
                                 return_model=True)

print("Objective=",uc_model.TotalCostObjective())
```

## Example Code - Alternative UC Model Build Functions

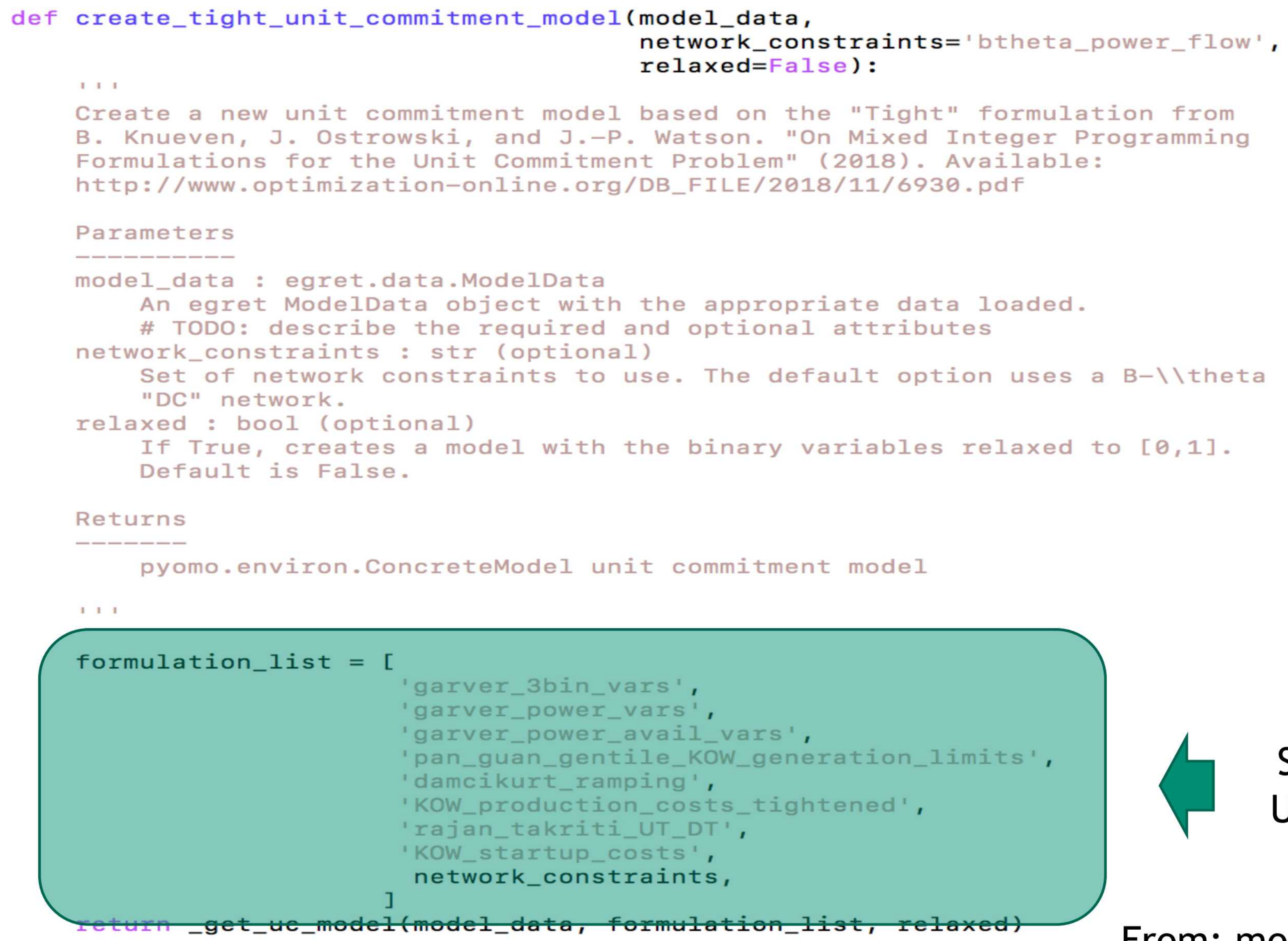


```
_test_uc_model(create_tight_unit_commitment_model)
_test_uc_model(create_compact_unit_commitment_model)
_test_uc_model(create_KOW_unit_commitment_model)
_test_uc_model(create_ALS_unit_commitment_model)
_test_uc_model(create_MLR_unit_commitment_model)
_test_uc_model(create_random1_unit_commitment_model)
_test_uc_model(create_random2_unit_commitment_model)
_test_uc_model(create_OAV_unit_commitment_model)
_test_uc_model(create_OAV_tighter_unit_commitment_model)
_test_uc_model(create_OAV_original_unit_commitment_model)
_test_uc_model(create_OAV_up_downtime_unit_commitment_model)
_test_uc_model(create_CA_unit_commitment_model)
```

Library of built-in UC model builders

From: models/tests/test_unit_commitment.py

## Example Code: Diving a Little Deeper

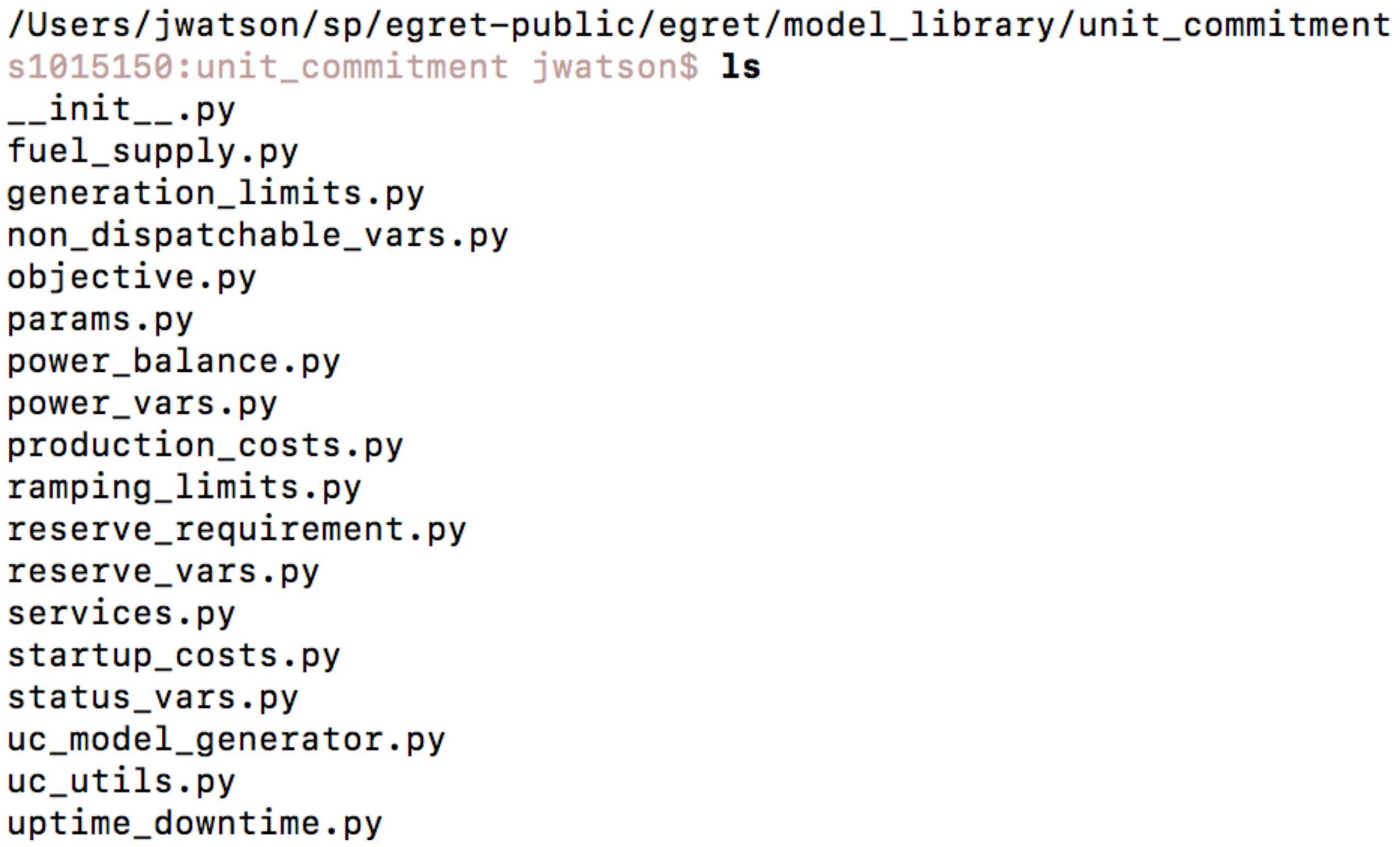


```
def create_tight_unit_commitment_model(model_data,
                                       network_constraints='btheta_power_flow',
                                       relaxed=False):
    '''
    Create a new unit commitment model based on the "Tight" formulation from
    B. Knueven, J. Ostrowski, and J.-P. Watson. "On Mixed Integer Programming
    Formulations for the Unit Commitment Problem" (2018). Available:
    http://www.optimization-online.org/DB_FILE/2018/11/6930.pdf

    Parameters
    ----------
    model_data : egret.data.ModelData
        An egret ModelData object with the appropriate data loaded.
        # TODO: describe the required and optional attributes
    network_constraints : str (optional)
        Set of network constraints to use. The default option uses a B-\\theta
        "DC" network.
    relaxed : bool (optional)
        If True, creates a model with the binary variables relaxed to [0,1].
        Default is False.

    Returns
    -------
        pyomo.environ.ConcreteModel unit commitment model

    '''

    formulation_list = [
                        'garver_3bin_vars',
                        'garver_power_vars',
                        'garver_power_avail_vars',
                        'pan_guan_gentile_KOW_generation_limits',
                        'damcikurt_ramping',
                        'KOW_production_costs_tightened',
                        'rajan_takriti_UT_DT',
                        'KOW_startup_costs',
                         network_constraints,
                       ]
    return _get_uc_model(model_data, formulation_list, relaxed)
```

Selection of different UC model components

From: models/unit_commitment.py

## UC Model Library: Structure



```
/Users/jwatson/sp/egret-public/egret/model_library/unit_commitment
s1015150:unit_commitment jwatson$ ls
__init__.py
fuel_supply.py
generation_limits.py
non_dispatchable_vars.py
objective.py
params.py
power_balance.py
power_vars.py
production_costs.py
ramping_limits.py
reserve_requirement.py
reserve_vars.py
services.py
startup_costs.py
status_vars.py
uc_model_generator.py
uc_utils.py
uptime_downtime.py
```

## EGRET: Capabilities Included But Not Covered Today

Full ancillary service stack

- Seven products
  - Reg-up, reg-down, spinning, non-spinning, supplemental, flex-up, and flex-down reserves

Storage

- Generic, full-fidelity models

Zonal definition support

- Relevant for reserve modeling

Network

- B-theta
- PTDF coming soon (already in power flow components of EGRET)

## Some Advanced Modeling Topics for Commitment and Dispatch…

## Fuel Constrained Modeling in EGRET

Current capability is an "instantaneous fuel supply", which is to model fuel availability for a specific time interval.

Exemplar is a natural gas hub, but could also be used for regional natural gas availability.

An arbitrary subset of generators can be attached to a fuel supply, though a single-fuel generator can only be attached to one.

Fuel consumed by generators (primary and auxiliary) at time *t* must be less than availability

$$\sum_{g \in \mathcal{G}_P(i)} f_g^P(t) + \sum_{g \in \mathcal{G}_A(i)} f_g^A(t) \leq S_i(t)$$

Modular modeling para nodels, e.g., on-site oil, higher fidelity NG model.

## Fuel Constrained Modeling in EGRET

Generator fuel consumption is tracked (for those generators that are fuel-constrained) by linear parameters based on minimum operating level, piecewise marginal generation, and start-up type.

- This requires only slight modification to the underlying unit commitment model, for a single fuel unit.

$$f_g^P(t) = \Delta(t) F_g^0 u_g(t) + \sum_{l \in \mathcal{L}_g} \Delta(t) F_g^l p_g^l(t) + \sum_{s \in \mathcal{S}_g} F_g^s \delta_g^s(t).$$

## Dual Fuel Modeling in EGRET

Allows for different incremental piecewise cost, no-load cost, and start-up cost based on fuel type

Tracks fuel consumption for both the primary (P) and auxiliary (A) fuel

Can model both on-line fuel switching and off-line only fuel switching

Dual-fuel units have their base start-up costs, no-load costs, and piecewise production costs replaced with a specialized dual-fuel model.

- Could replace other components as well, e.g., ramp-rates, min/max generation levels.

## Dual Fuel Modeling in EGRET

Either Primary or Auxiliary fuel when committed

Tracking differing no-load costs

$$u_g(t) = u_g^P(t) + u_g^A(t)$$

Tracking differing marginal costs

$$C_g^{R,P} u_g^P(t) + C_g^{R,A} u_g^A(t)$$

$$p_{l,g}^P(t) \leq (\overline{P}_{l,g}^P - \overline{P}_{l-1,g}^P) u_g^P(t) \qquad \forall l \in \mathcal{L}_g^P,\ \forall t \in \mathcal{T}$$

$$p_{l,g}^A(t) \leq (\overline{P}_{l,g}^A - \overline{P}_{l-1,g}^A) u_g^A(t) \qquad \forall l \in \mathcal{L}_g^A,\ \forall t \in \mathcal{T}$$

$$c^p(t) = \sum_{l \in \mathcal{L}^P} \Delta(t) C_{l,g}^P p_{l,g}^P(t) + \sum_{l \in \mathcal{L}^A} \Delta(t) C_{l,g}^A p_{l,g}^A(t) \qquad \forall t \in \mathcal{T}$$

$$p'(t) = \sum_{l \in \mathcal{L}^P} p_{l,g}^P(t) + \sum_{l \in \mathcal{L}^A} p_{l,g}^A(t) \qquad \forall t \in \mathcal{T}.$$

## Dual Fuel Modeling in EGRET

Tracking differing start-up costs

$$\sum_{t'=t-TC_g+1}^{t-DT_g} x_g^P(t',t) \le v_g^P(t)$$

$$\sum_{t'=t-TC_g+1}^{t-DT_g} x_g^A(t',t) \le v_g^A(t)$$

$$\sum_{t'=t+DT_g}^{t+TC_g-1} x_g^P(t,t') + \sum_{t'=t+DT_g}^{t+TC_g-1} x_g^A(t,t') \le w_g(t)$$

$$c_g^{SU}(t) = C_g^{S_g,P} v_g^P(t) - \sum_{s=1}^{S_g-1} (C_g^{S_g,P} - C_g^{s,P}) \left( \sum_{t'=t-\underline{T}_g^{s+1}+1}^{t-\underline{T}_g^s} x_g^P(t',t) \right)$$

$$+ C_g^{S_g,A} v_g^A(t) - \sum_{s=1}^{S_g-1} (C_g^{S_g,A} - C_g^{s,A}) \left( \sum_{t'=t-\underline{T}_g^{s+1}+1}^{t-\underline{T}_g^s} x_g^A(t',t) \right)$$

## Dual Fuel Modeling in EGRET

Fuel consumption tracking for both primary and auxiliary fuel based on start-up type, marginal production, and no-load

$$f_g^P(t) = \Delta(t) F_g^{P,0} u_g^P(t) + \sum_{l \in \mathcal{L}_g} \Delta(t) F_{l,g}^P p_{l,g}^P(t) + F_g^{P,S_g} v_g^P(t) - \sum_{s=1}^{S_g - 1} \sum_{t' = t - \underline{T}_g^{s+1} + 1}^{t - \underline{T}_g^s} (F_g^{P,S_g} - F_g^{P,s}) x_g^P(t', t)$$

$$f_g^A(t) = \Delta(t) F_g^{A,0} u_g^P(t) + \sum_{l \in \mathcal{L}_g} \Delta(t) F_{l,g}^A p_{l,g}^A(t) + F_g^{A,S_g} v_g^A(t) - \sum_{s=1}^{S_g - 1} \sum_{t' = t - \underline{T}_g^{s+1} + 1}^{t - \underline{T}_g^s} (F_g^{A,S_g} - F_g^{A,s}) x_g^A(t', t)$$

## Current Practice for Dual-Fuel Units

Current ISO/RTO practices allows dual-fuel units to bid-in different cost curves, but the units (or market participants) decide when to execute their fuel switch.

This can be inefficient, as natural gas generators share some gas resources.

With some (moderate) insight into the gas supply network, the ISO/RTO could instruct market participants when to fuel switch based on shared fuel supply constraints.

- This can enhance reliability and resilience, especially when fuel supply is a binding constraint.

## In Conclusion…

## Conclusions and Future Research Directions: General

- Comprehensive literature and computational survey of UC formulations is available
- Along with open-source implementations of all the examined (and many more unexamined) UC formulations
- Literature and computational survey uncovered a new state-of-the-art

- Additional challenges remain
  - Tightening the interaction between the unit commitment and the transmission system (Van den Bergh et al. 2014, Wu 2016)
  - Virtual transactions, which may weaken our ability to tighten system constraints (Chen et al. 2016)
  - More realistic modeling of the transmission system, including the need for reactive power support (Castillo et al. 2016)
  - Better modeling of ancillary service products, which are relatively neglected by the literature and tend to vary by market

## Future Research Directions: NAERM

Intellectual investment in commitment / dispatch libraries is massive

- Decades of person-years invested in aggregate
- Very intricate knowledge of mathematical programming required for performance advances
- Significant advances come from the lab community, not necessarily industry
- Screams for consolidation across the complex

Drive toward open and transparent models is critical

- Necessary condition for V&V of power systems operations simulation models

Resilient (i.e., Proactive) Commitment and Dispatch

- Based on past experience at SNL, discussions with NREL

Enhanced Fuel Supply Constrained Models

- Co-optimization of NG and BES
- Avoiding non-linearities of full NG system models