Fast Relax-and-Round Unit Commitment With Economic Horizons

Shaked Regev, Evan J. R. Brody, Charles Foltz, Eve Tsybina, Slaven Peleš Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, Tennessee, USA

***Abstract*—The US energy system is increasingly under pressure to serve expanding data loads and to accommodate a larger number of generating units with varying technologies and ownership structures. Therefore, developing new unit commitment methods remains a priority for reliable and affordable grid operations. We expand our novel computational method for unit commitment (UC) to include ramping constraints and long-horizon planning and provide a theoretical bound on its error. We introduce a fast novel algorithm to commit hydro-generators. We solve problems with thousands of generators at 5-minute market intervals. We show that our method can solve UC problems with over 20,000 generators in approximately 10 seconds on commodity hardware and that an increased planning horizon leads to sizable operational cost savings. We attain this runtime improvement by introducing a heuristic tailored for UC problems. Our method can be implemented using existing continuous optimization solvers and adapted for different applications. We prove a bound on the error of these solvers and show that it vanishes (in relative terms) as the problem becomes larger. We also introduce a fast and accurate hydro UC algorithm. Combined, these algorithms would allow an operator to make horizon-aware economic decisions for large systems with hydro units.**

***Index Terms*—optimization, power systems**

I. Introduction

The electric power system is under increasing operational pressure due to significant load volatility caused by the increasing demand for electricity and rapid expansion of data centers. The North American Electric Reliability Corporation classifies data centers as a distinct load category, recognizing their unique planning, operational, and balancing risks [1]. Unlike traditional industrial loads, data centers can ramp rapidly in response to computational workloads, creating steep intra-hour load changes [2].

In power systems with a high share of such loads, unit commitment (**UC**) changes in multiple ways. Connecting fast-ramping, often smaller, generators increases the size of UC problems. Further, otherwise "invisible" intra-hour volume generated by slow large unit ramping must be explicitly modeled to ensure generation volume transparency in UC. Finally, storage is operationally critical and must be committed to ensure availability during periods of rapid load adjustment.

Combined, these effects impact the temporal structure of UC problems. First, tighter and more reliable ramping requires a shift toward finer temporal resolution. There is an ongoing discussion about moving from the traditional 1 hour interval to sub-hourly intervals, possibly as short as 5 minutes [3], [4]. Further, incorporating flexible resources requires multi-period optimization, since they can strategically allocate their generation between several periods.

Increasing temporal granularity causes a proportional growth in problem size, i.e., the number of variables and constraints [5], [6]. Intertemporal coupling leads to a super-linear increase in runtime. Most large-scale UC formulations rely on mixed-integer program (**MIP**) solvers [7]. Their runtime scales poorly as spatial or temporal resolution increases [8]–[11]. Existing heuristic decomposition and acceleration methods' performance typically deteriorates when the objective function is strongly temporally coupled, as it is with multi-period UC with ramping constraints. In such settings, solving a single long-horizon UC problem is considerably more demanding than solving multiple shorter horizon problems [5], [6]. Therefore, many studies limit the number of generating units, temporal resolution, or planning horizon.

Another class of commercial MIP solvers, such as Gurobi, linearizes the objectives and constraints of the problem, relaxing the MIPs to mixed-integer linear programs [12]. This allows using heuristics such as branch-and-bound to decrease runtime compared to MIP solvers, but their worst case (and often in practice) scaling is still exponential in the number of variables (in this case, related to the number of generators) and constraints. Further, the resulting linearization does not capture voltage limits, reactive power constraints, and non-linearities in generator production cost. Our solver uses an interior point method common for continuous problems, which is not limited to linear functions and only requires twice-differentiable functions. In this paper, only our objective is non-linear, but in future work we will introduce transmission constraints in their non-linear (alternating current optimal power flow) form.

We further extend this formulation to include hydropower. We focus on hydropower reservoirs, which are charged uncontrollably by rainfall and discharged controllably, subject to water balance constraints [13]. Pumped hydropower, which functions as a battery with controllable charging and discharging [14], is deferred to future work.

 Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the US Department of Energy.

Corresponding author: Shaked Regev (e-mail: regevs@ornl.gov).

Section II introduces our long-horizon planning UC formulation. Section III introduces a novel algorithm to adaptively commit hydro units before traditional units and tests its performance empirically. Section IV summarizes our results and future research directions. Appendix A shows a proof that our main RRUC algorithm in Section II has a relative error that vanishes as the problem size increases and gives a tighter bound on the deviation from the solution to the relaxed problem. Appendix B shows a proof that the original problem proposed in Section III is NP-Hard and that our proposed algorithm runs in log-linear time.

II. Horizon-Aware Unit Commitment

A. Mathematical Formulation

We expand our previous work's formulation [15] to explicitly include minimum run time, maximum daily starts, ramping constraints and generation costs in subsequent time periods. We use the mathematical model in Eq. (1) to solve UC.

$$\min_{P_i, P_{\delta,i}, u_i} \quad \sum_{i \in \mathcal{G}_m \cup \mathcal{G}_d} \left(a_i P_i^2 + b_i P_i + c_i\right) u_i u_{t-1,i} + \sum_{i \in \mathcal{G}_d} K_i (u_{t-1,i}(1-u_i) + (1-u_{t-1,i}) u_i) + \gamma \sum_{\delta} \sum_{i \in \mathcal{G}_m \cup \mathcal{G}_d} \left(a_i P_{\delta,i}^2 + b_i P_{\delta,i} + c_i\right) u_i \tag{1a}$$

$$\text{s.t.} \max(P_{\min,i}, P_{t-1,i} - r_{d,i}) \leq P_i \leq \min(P_{\max,i}, P_{t-1,i} + r_{u,i}) \tag{1b}$$

$$\forall i \in \mathcal{G}_d : u_i \in \{0,1\} \tag{1c}$$

$$\forall i \in \mathcal{G}_m : u_i = 1 \tag{1d}$$

$$\sum_{i \in \mathcal{G}_m} P_i + \sum_{i \in \mathcal{G}_d} \left[u_i P_i + (1-u_i) P_{\min,i}\right] u_{t-1,i} \geq D - S_r, \tag{1e}$$

$$\sum_i u_i P_{\max,i} \geq R_\uparrow \tag{1f}$$

$$\sum_i u_i P_{\min,i} \leq R_\downarrow \tag{1g}$$

$$\sum_i u_i P_{\delta,i} \geq D_\delta, \ \forall \delta, \tag{1h}$$

with $R_\uparrow \doteq D_{\max,48} + 3\sigma_D + R - S_{\max,r}$ and $R_\downarrow \doteq D_{\min,48} - \sigma_D - S_{\min,r}$.

The extension of our previous work is generator ramping and mechanical constraints and the last term in Eq. (1a) and Eq. (1h). These balance a UC solution that is efficient now with one that will efficiently meet future demand. By our convention, all variables and parameters refer to the current time period being solved, unless otherwise noted. $\mathcal{G}_m$ and $\mathcal{G}_d$ and are sets of must-run and discretionary generators, respectively. The decision variables are:

- $P_i \in [P_{\min,i}, P_{\max,i}]$ for $i \in \mathcal{G}_m \cup \mathcal{G}_d$: power output of must-run or discretionary generator i
- $P_{\delta,i}$: Production level by generator i at demand level δ. The set of δs considered can be all demand points in the 48 hour time window, or a representative sample.[2]
- $u_i \in \{0,1\}$ for $i \in \mathcal{G}_d$: commitment status of discretionary generator i. $u_i = 0$ means staying off or starting to ramp down. $u_i = 1$ means staying on or starting to ramp up.

The parameters supplied to the optimization solver are:

- a_i, b_i, c_i: quadratic cost coefficients usually resulting from fuel costs or strategic bidding behavior.
- $P_{\max,i}, P_{\min,i}$: maximum capacity and minimum stable output, determined by a unit's physical characteristics.
- D, σ_D: demand volume and standard deviation
- K_i: commitment change penalty for generator i.[3]
- $r_{u,i}$ and $r_{d,i}$: generator i's limits for ramping up and down, normalized so that $\Delta t = 1$.
- γ: bias factor to select units that will be more efficient at meeting future demand. In our model $\gamma = 1$.
- $u_{t-1,i}$: indicator that is 1 if and only if generator i was ramping up or on in the previous time period
- $P_{t-1,i}$: unit i's production level at the previous time period.
- S_r: supply produced by generators that are already ramping before the solve (controlled). Follows ramping profiles in Fig. 1.
- $S_{\max,r}$: total capacity of ramping up unit (ramping down generators will be off before the contingency)
- $S_{\min,r}$: minimum working capacity of ramping up units
- $D_{\min,48}$ and $D_{\max,48}$: minimum and maximum volumes of predicted demand in next 48 hours.
- D_δ: discrete representation of demand probability in next 48 hours.
- $R = \max_i P_{\max,i}$: reserve margin, used to handle $N-1$ contingencies of generator outages.

Eq. (1) is a MIP, which in general are NP-hard. Branch-and-bound methods are typically used to approximate solutions [8]. Our algorithm temporarily relaxes $u_i \in \{0,1\}$ to $y_i \in [0,1]$ to solve Eq. (1) in a similar fashion to branch-and-bound methods, but we use different heuristics that are specific to UC problems. We first re-order discretionary generators $\mathcal{G}_d$ so that $y_1 \geq \ldots \geq y_{|\mathcal{G}_d|}$. We then find the first $\tilde{m}$ discretionary generators such that

$$\sum_{i \in \mathcal{G}_m} P_{\max,i} + \sum_{\substack{1 \leqslant i \leqslant \tilde{m} \\ i \in \mathcal{G}_d}} P_{\max,i} \ \geq \ R^\uparrow \tag{2}$$

Some previously committed units must operate due to runtime and power ramping constraints (the set $\mathcal{G}_m$). Therefore, the minimum number of generators to operate is $m \doteq |\mathcal{G}_m| + \tilde{m}$. We can calculate $m_{\max}$, the maximum number of generators that can operate, using Eq. (1g) analogously to the way we used Eq. (1f) to derive Eq. (2). This gives us Eq. (3):

$$\sum_{i \in \mathcal{G}_m} P_{\min,i} + \sum_{\substack{1 \leqslant i \leqslant \tilde{m}_{\max} \\ i \in \mathcal{G}_d}} P_{\min,i} \ \geq \ R^\downarrow, \tag{3}$$

[1] We choose these contingencies as an example, but our model is adaptable to any contingency.

[2] This utilizes the cost curves' monotonicity. If a set of units can supply 2 demands efficiently, it can supply any demand between them efficiently.

[3] The true startup and shutdown costs are rarely the same. Averaging them prevents market distortion and removes artificial incentives to turn on or off.

with $m_{\max} \doteq |\mathcal{G}_m| + \tilde{m}_{\max}$. Note Eq. (1h) is satisfied by construction, because $\forall \delta : D_{\min,48} \leq D_\delta \leq D_{\max,48}$. We then solve economic dispatch with all options of generators that can supply the load. Meaning, $\forall k : m \leq k \leq m_{\max}$, we set $y_{j \leq k} = 1$ and $y_{j>k} = 0$ and solve:

$$\min_{P_j} \quad \sum_{j=1}^{k} \left(a_j P_j^2 + b_j P_j + c_j\right) \tag{4a}$$

$$\text{s.t.} \quad \max(P_{\min,i}, P_{t-1,i} - r_{d,i}) \leq P_i \leq \min(P_{\max,i}, P_{t-1,i} + r_{u,i}) \tag{4b}$$

$$\sum_{j=1}^{k} P_j \geq D. \tag{4c}$$

Finally, we take the solution with lowest objective. Note that we account for the on/off cost terms in line 3 of Eq. (1a) within the rounding scheme and add them to the objective of their respective economic dispatches (Eq. (4)), before deciding the minimal objective. Eq. (4) only optimizes over the current period, because Eq. (1) is only used to decide which units to turn on or off based on projected demand. It does not commit generators to producing P_δ power when demand D_δ arrives, it merely ensures they can do so efficiently, when needed.

Algorithm 1 summarizes RRUC. Line 3 is not strictly necessary, but when the linear program (**LP**) solver is set to a low tolerance (in our implementation, 10^{-9}), this step helps with numerical precision in the y_i values, which reduces iterations of the for-loop on line 9. See a bound on the error in Appendix A.

Algorithm 1 Runtime and ramp constrained RRUC

1: **for each period do**
2: Solve Eq. (1) with $u_i \in \{0,1\} \rightarrow y_i \in [0,1]$
3: Fix all solution values except y_i, solve the induced LP
4: Rank generators in descending order ($y_1 \geq \ldots \geq y_n$)
5: $\ell \leftarrow \min\{k : \sum_{i=1}^{k} P_{\max,i} \geq R_\uparrow\}$
6: $h \leftarrow \max\{k : \sum_{i=1}^{k} P_{\min,i} \leq R_\downarrow\}$
7: $\ell' \leftarrow \max\{k : y_k = 1\}$
8: $h' \leftarrow \max\{k : y_k > 0\}$
9: **for** $k \leftarrow \max(\ell, \ell')$ **to** $\min(h, h')$ **do**
10: Solve Eq. (4) with generators $1, \ldots, k$
11: Add state change costs to objective
12: **if** objective is lowest so far **then**
13: Update best solution and objective
14: **end if**
15: **end for**
16: Update generator states
17: Find must run units (with minimum on time not yet reached or with $P_{t-1,i} - P_{\min,i}$ too large to shut off)
18: Find can't start generators (those that have exceeded maximum allowed daily starts or are already ramping)
19: **save** Best solution and best objective
20: **end for**

Fig. 1 shows a typical production cycle of a unit and the allowed transitions. Each unit can only be in one operating state in any time period, so we round up ramping times

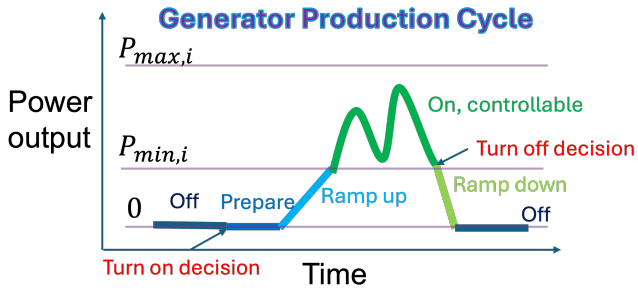

Fig. 1. Unit i's typical production cycle. When on, $P_i \in [P_{\min,i}, P_{\max,i}]$, though it cannot exceed its ramping power. It can turn off if $P_{t-1,i} - r_{d,i} \leq P_{\min,i}$. The only other decision can be made when the unit is off, to start ramping up. During other periods, it is uncontrollable.

from [16] to whole periods. Rounding up ensures our solution is a more restrictive UC than the actual UC, meaning it is necessarily feasible with respect to ramping constraints[4]. This means a unit must occupy each state for at least one time period. We use preprocessing to account for the accumulated runtime and ramping periods. We store information about the duration of off, on, or ramping for each unit and use it when solving Eq. (1).

B. Simulation and Results

Our simulations use historical PJM data for load and generating unit bids. PJM recorded its historical on June 23rd, 2025 at 5 pm, with total load of 160.2 GW [16]. We apply this value to scale the system using a 20% cold reserve margin for resource adequacy. Our demand timeframe is June 18th through June 24th, divided into 5-minute periods. PJM historical data is hourly, so we use linear interpolation to increase the resolution of the data.

Our unit selection includes 42 real representative PJM generators and amounts to 9047.9 MW capacity for the entire system. We fitted the piece-wise linear bids from PJM to quadratic cost curves. To scale up the problem, we used noisy copies of the original 42 generators. We create these by changing continuous generator up to $\pm 10\%$ and runtime constraints ± 1 hour randomly (clamping them to at least 1 and at most 24 hours).

To incorporate the runtime constraints, we augmented PJM generator data with runtime parameters summarized in Table I. PJM data had no clear assignment of units to gas or coal fuel, so we manually classified the units using the following rules: No unit larger than 1000 MW was considered gas because, according to EIA 860 [17], there were no operable gas units of this size in PJM territory. All units with a minimum runtime of 24 hours are assigned coal, unless their starting price in the supply curve is above \$10/MWh. The cost-based fuel assignment relies on the assumption that coal units have lower operating costs than gas units, although coal prices can be above gas prices on equivalent energy content [18]. Hot, warm,

[4]This is a limitation on using discrete time periods to model a continuous process that is not particular to our solver. Solving UC at smaller time steps reduces this issue, along with others, motivating faster methods such as ours.

and cold start costs are available from PJM [16]. We adopt cold start values for our model.

TABLE I
RUNTIME ASSUMPTIONS AND DATA SOURCES

Parameter	Source	Coal	Gas
Min runtime [min]	[16]	240-1440	0-1440
Max daily starts	[16]	1-3	1-24
Cold start [min]	[19]	360-720	120-300

We gathered the original ramping characteristic data in % of $P_{\max}$ per minute, and then converted to MW/min for input into the simulation. We compile the percentage values from the academic reports of Deutsches Institut für Wirtschaftsforschung [19] and the manufacturer catalog of GE Vernova [20]. These values provide the upper bound of the physical ramp rates, as a survey of large thermal units suggests that in real systems power plants are often not able to deliver the nominal ramp rates specified for equipment [21].

We used approximations based on size and the assumed fuel used by a turbine (gas or coal steam). We assigned larger turbines and coal steam turbines lower ramp values. First, we assigned the ramp percentages to specific generating units. Then, we multiplied $P_{\max}$ of those generators the ramp rate to get the MW/min value we used in the simulation.

If the system starts with all generators off, Eq. (1) is infeasible. We initialized the system by solving a non-ramp-constrained UC instance and then treating the selected generators as if they had been on for the past 24 hours, for the purpose of runtime constraints. If operators are warm starting using Algorithm 1 they can use the current system state.

TABLE II
RAMP ASSUMPTIONS AND DATA SOURCES

Parameter	Source	Coal	Gas
Ramp up rate [%/min]	[19], [20]	1-6	2-12
Ramp down rate [%/min]	[19]	5	15

We implement our method in Julia [22] with the MadNLP nonlinear programming solver [23], which built upon [24], [25], with the MA57 linear solver from HSL [26] and the JuMP modeling language [27]. We use the Gurobi linear program solver [12]. Our computational experiments use an Apple M5 MacBook with 10 cores and 24GB RAM.

We study a system with 462 units to determine the last term in Eq. (1a)'s impact on the overall operation costs. Each 48-hour planning period contains 576 5-minute intervals, each with a load value. To ensure that we can meet this demand efficiently with the units selected for the current period, we include select representative demands in the last term of the objective function, Eq. (1a).[5] Together, the values constitute a load probability distribution over a 48-hour forecast. We use a stochastic collocation method to approximate the probability distribution with a set of discrete points [29].

[5]We select demand points using the nodes of the Clenshaw-Curtis quadrature rule [28], mapping each node to its corresponding quantile within the sorted set of discrete 48-hour demand forecasts.

Fig. 2 shows the cost reduction and runtime tradeoff using a serial implementation of RRUC. It is possible that parallelization would lead to further run time reductions. The results show an approximately 27% decrease in cost from from the 0 future point model (**FPM**) to 1-FPM, an approximately 5% lower cost when using 2-FPM and 3-FPM, and negligible advantage from more future points. Runtime scales linearly in the number of points used. This means that including the mean, minimum, and maximum projected demands in the planning horizon helps the solver reduce overall costs. Including more demands helps a little, but increases solve time a lot. Different applications may select a different tradeoff, but in light of these results, we use 3-FPM for the rest of the paper.

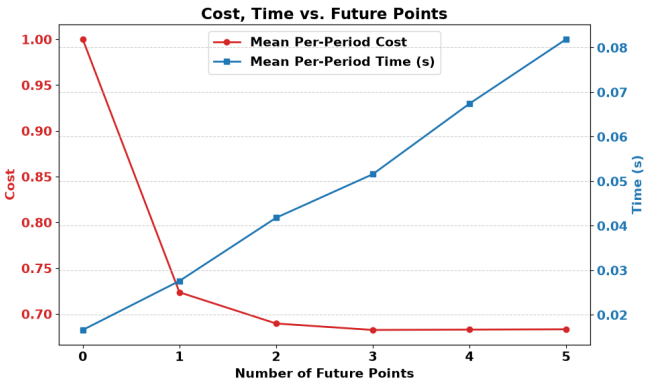

Fig. 2. RRUC's objective and runtime scaling with different FPMs on a 462-generator test case. Runtime increases roughly linearly and the objective levels off. This motivates taking a demand sample D_δ instead of all demands in Eq. (1h).

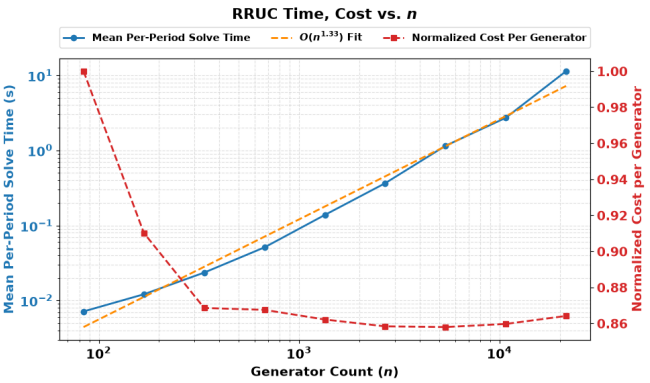

Fig. 3. Objective and runtime scaling of the 3-FPM from 84 to 21,504 generators. The 3-FPM retains accuracy at scale. RRUC's runtime scales as $\approx O(n^{4/3})$, where n is the number of units. Cost values per-generator are normalized by the the smallest case (84 generators). Demand data is from PJM's historical data from June 18th to June 24th, 2025.

We run Algorithm 1 with 3-FPM for 2016 time periods at 5 minute intervals (7 days) for June 18th to 24th, 2025, which covers the PJM historical peak. Fig. 3 shows that 3-FPM retains accuracy at scale. Its normalized objective decreases slightly with an increase in system size.

RRUC's time complexity is dominated by the numerical linear algebra in the continuous optimization we use to solve

the relaxed Eq. (1). It is estimated to be $O(n^{1.5})$ [8]. Algorithm 1 meets and outperforms this scaling slightly. This may be because linear systems generated by Eq. (1) are sparser than typical optimization problems due to the $P_{\delta,i}$s which do not interact much with the other variables.

To benchmark our speed and accuracy, we compare our results with results obtained by using the Gurobi optimization software [12] by replacing our solver (lines 1-14 in Algorithm 1) in our Julia code with a MIP solver from Gurobi's Julia package, using 10 threads for parallelism. As an example, we run with varying generator counts for 7 days each, with a 48-hour planning window. One can choose any planning window without increasing runtime due to FPM sampling.

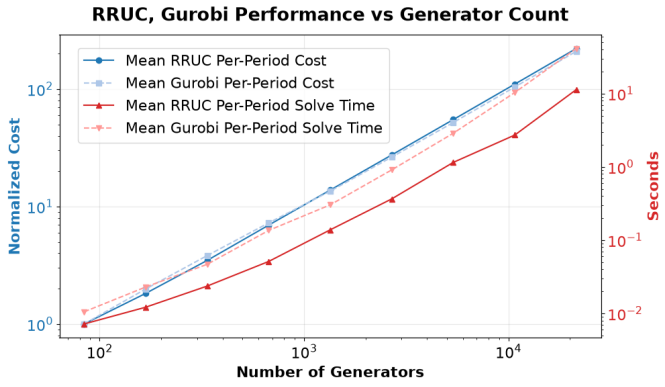

Fig. 4. RRUC's solve time and normalized operation cost compared with Gurobi's with 84-21,504 generators. Values are averaged over 2 days (576 5-minute periods). Objective values are normalized relative to the smallest 84-generator case. The algorithms produce similar quality solutions, but RRUC is faster. This gap increases with more units.

Fig. 4 shows that Gurobi and RRUC produce numerically indistinguishable solutions, though RRUC is twice as fast on the largest problem tested. RRUC achieves this using one parallel process compared to Gurobi's 10. Both algorithms scale well in the number of variables, however, Gurobi's runtime scales exponentially in the number of constraints. For the largest problem tested, Gurobi took half the time on the 2-FPM that it did on the 3-FPM. Using the 5-FPM, Gurobi took almost four times as long as the 3-FPM. This leads us to believe that on AC transmission-constrained problems, where the number of constraints grows linearly in the problem size, RRUC will have a substantial advantage. We save this investigation for future work.

III. Hydro-generator commitment

A. Mathematical Formulation

We now include hydro units in our UC. Our approach commits the hydro units and then traditional units, using Algorithm 1 as in Section II. This decoupling is justified because hydro units recharge by natural water flow and cannot indefinitely hoard water due to the dam's capacity. So as long units maintain high water levels long term, there is little opportunity cost. Hydro units are fast ramping [30] and typically have a narrow efficient operation regime [31].

Here, we make the simplifying assumptions that hydro units can only be on at max capacity or off, the starting water levels are far away from 0 and the maximum dam capacity, and water evaporation rates are independent of water levels. We denote by $W_j([T])$ the amount of water projected to flow into reservoir j over a certain time window $[T] \doteq \{1, \ldots, T\}$. We denote by $\tilde{W}_j([T])$ the water required for hydro unit j to produce electricity in one time period. The number of time periods it can operate within window $[T]$ is $\pi_j \doteq \min(W_j([T])/\tilde{W}_r([T]), T)$. This reduces our problem to finding in which periods $t \in [T]$ to operate each unit.

Hydro generators not being able to run all (or even most of) the time means that the FPM approach (Algorithm 1) is not viable for committing them. They require optimizing over long-term, for example yearly, cycles. We therefore design a method to commit hydro generators based on shifting predicted demands. For this purpose, we introduce the notion of the marginal cost of the system.

The marginal cost of the system Eq. (1) is the cost difference to produce slightly more or less power. All on units i not at the feasibility boundary must have the same marginal cost $u_i(2a_iP_i + b_i)$ at the optimal solution. Otherwise, shifting one unit's production up and one down could reduce cost. We define the marginal cost of the system therefore as $u_i(2a_iP_i + b_i)$ for one such of these units i. Because $\forall i : a_i, b_i \geq 0$, a good solver for Eq. (1) should produce solutions where higher demand is positively correlated with higher prices.

Fig. 5 shows how commonly certain demands coincide with certain marginal costs. A high demand always leads to high marginal costs and low demand always leads to low costs. There are small deviations in this trend due to periods that are less constrained by future generation requirements and "free" power from ramping generators.

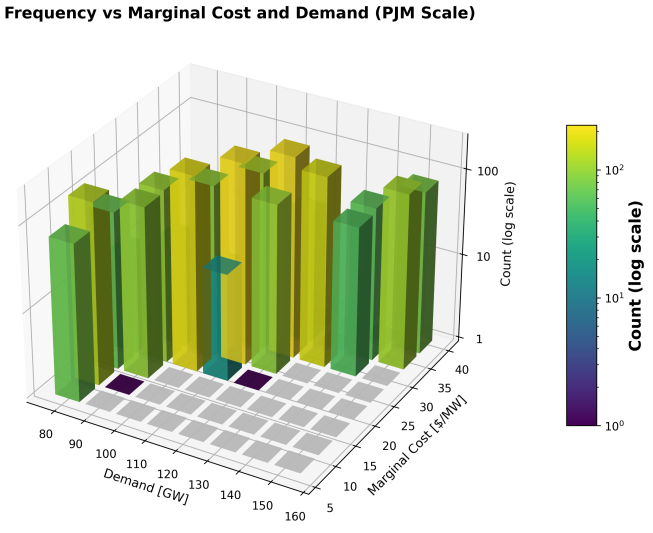

Fig. 5. Histogram of the frequency of demand-marginal cost pairs (counts) at the PJM scale. Marginal cost and demand are highly positively correlated. All rows and columns have at most 3 adjacent nonzero boxes, most have 2. This shows that outliers do not deviate much.

Fig. 6 shows 1, 2, and 3 degree polynomial fits of the marginal cost to the demand, with their adjusted R-squared

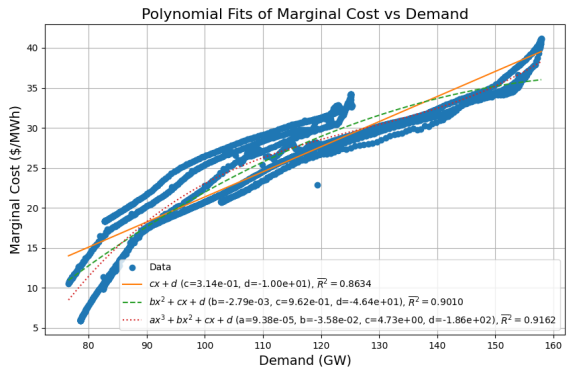

Fig. 6. Polynomial fits of marginal cost as a function of demand. Marginal cost and demand are highly positively correlated, with a few outliers. The degree 3 polynomial has the largest adjusted R-squared ($\bar{R}^2$) and is visually most similar to the data. The relative scale of the axes also suggests a cubic fit. The linear term is always dominant and positive. This suggests it is almost always better to shift net demand from high demand periods to low demand periods with hydropower or storage.

$\bar{R}^2$. The $\approx 2\times$ difference between minimum and maximum demand to $\approx 8\times$ difference between minimum and maximum marginal cost further motivates a cubic fit. A 4th degree increased $\bar{R}^2$ less than 0.1% and made no visual difference.

The linear term is dominant in all the fits shown here. The linear term of the 4 degree fit has a non-dominant negative coefficient, a clear sign of over-fitting, based on visual inspection of the data. This fit motivates the heuristic of using hydropower to prefer reducing demand from high demand periods over low demand periods. It also shows that even if pumped storage (or any other battery) does not retain energy perfectly, i.e. it can discharge a smaller amount than it charges, it can still be well worth-while to shift demand from low to high demand periods.

These findings motivate using hydro generators to flatten load or peak shave. We define the demand remaining after hydro-generator commitment at time t optimized over $[T]$ as $\bar{D}_t = D_t - \sum_{j=1}^{n} u_{j,t}\kappa_j$, where κ_j is the power hydro-generator j can produce. Smoothing out $\bar{D}_t$ will lead to lower overall costs, because the marginal cost is highest when demand is highest. This leads to the following optimization problem:

$$\min_{u_{j,t}} \quad \text{Var}\left(D_t - \sum_{j=1}^{n} u_{j,t}\kappa_j\right) \tag{5a}$$

$$\text{s.t.} \quad \forall 1 \le j \le n : \sum_{t=1}^{T} u_{j,t} = \pi_j, u_{j,t} \in \{0,1\} \tag{5b}$$

Solving Eq. (5) is NP-hard (see Theorem 2 in Appendix B). Algorithm 2 approximates a solution by "greedily" assigning a which still has remaining periods to the period with the highest remaining demand. This is equivalent to locally optimizing Eq. (5). Algorithm 2 runs in log-linear time (see Theorem 3 in Appendix B).

Algorithm 2 Demand Balancing

Require: Capacities κ_j, maximum operating periods π_j for $j = 1, \ldots, n$; demands d_t for $t = 1, \ldots, T$

$G_t \leftarrow 0 \quad \forall t$ ▷ Generated hydro-power at period t
Initialize indexed max-heap H with keys $k_t = D_t$
$\text{used}_t \leftarrow 0 \; \forall t$ ▷ Tracks assignments
Re-index such that $\kappa_1\sqrt{\pi_1} \ge \ldots \kappa_n\sqrt{\pi_n}$ ▷ Improves numerical stability
for $j \leftarrow 1$ **to** n **do**
  $\text{placed} \leftarrow 0$
  while $\text{placed} < \pi_j$ **do**
    $\tau \leftarrow \text{argmax}_t \; k_t$
    if $\text{used}_\tau = j$ **then**
      Set $k_\tau \leftarrow \infty$ and update H
    else
      $\text{used}_\tau \leftarrow j$
      $\text{placed} \leftarrow \text{placed} + 1$,
      $G_\tau \leftarrow G_\tau + \kappa_j$
      Update $k_\tau \leftarrow D_\tau - G_\tau$ in H
    end if
  end while
  for $t \leftarrow 1$ **to** T **do**
    if $k_t = \infty$ **then**
      Update $k_t \leftarrow D_t - G_t$ in H
    end if
  end for
end for
return $(L_t), (D_t - G_t), \text{Var}(D_t - G_t)$

B. Simulation and Results

We use hydro unit data from the US Energy Information Administration (**EIA**) [17]. From this set, we take all hydro units in PJM. In this work we consider all hydro-generators as reservoirs, as opposed to pumped storage, which we save for future work. We aggregate the nameplate capacities for each plant from [17]. The capacity factor is the percentage of time a hydro-generator can operate.

In the contiguous US, the median capacity factor is $36-39\%$, with large variance [32]. In the absence of more granular data, we assign different capacity factors between 28 and 47% to the generators to test Algorithm 2. This gives us a base set of 56 hydro-generators for PJM's demand.

For our scaling experiments, we multiply the total demand by a factor N, and create $N-1$ noisy copies of the generators to commit. The copies have a capacity that shifted up to $\pm 10\%$ from the original generator, and can operate up to ± 10 periods within a 24 hour time window compared to the original generator. Fig. 7 shows normalized $\sigma_N \propto \sqrt{\text{Var}(\bar{D}_\mathcal{T})}/m$, where m is the number of generators. σ_N is normalized so that with 56 generators allocated over 7 days, $\sigma_N = 1$.

Algorithm 2 retains accuracy in its objective and its run time scales almost linearly. It can commit 1792 generators for an entire year in 20 seconds. This is useful because rainfall follows yearly cycles. So it can easily run at 5-minute intervals to keep up with evolving demand prediction and dam water levels (which impact how many periods within the year they

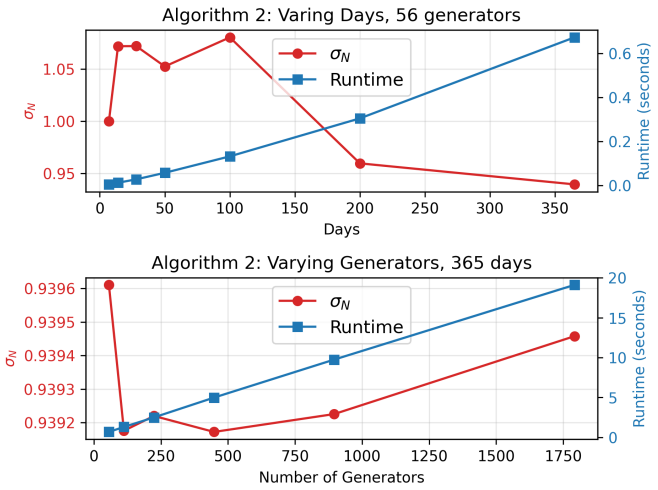

Fig. 7. Algorithm 2's normalized objective and runtime varying the number of days and generators. Its performance is stable as the demand (and number of generators to supply it) increase and in optimizing over longer stretches. Its runtime increases linearly in the number of generators, and slightly faster than linearly in the number of days (or periods) as Theorem 3 guarantees.

can operate).

We compare our solver for Eq. (5) to the commercial integer program solver Hexaly, which is faster than Gurobi for scheduling problems with many constraints such as this one [33], in Fig. 8. Hexaly consistently gets marginally worse solutions, but is still within 10^{-4} relative error of the known lower bound (our prescribed tolerance), and takes much longer. Hexaly reported timeout errors (even before it met the time limit) for problems larger than in Fig. 8.

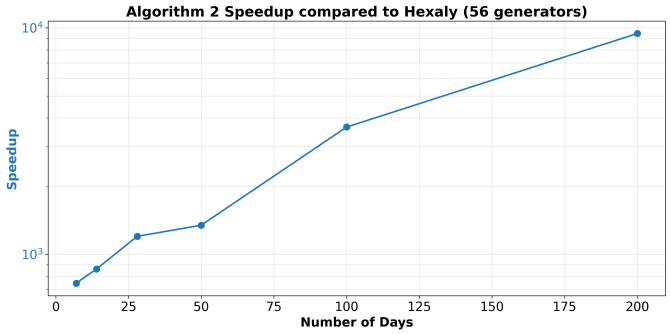

Fig. 8. Algorithm 2 speedup compared to Hexaly integer program solver for committing 56 generators for different time horizons. Algorithm 2 always gets slightly better solutions, though both are within tolerance of optimal. It gets much faster solutions, a gap that increases roughly linearly in the problem size. It was almost $10,000$ times faster on the largest problem Hexaly could run, which was about 60 times smaller than the largest problem we tested Algorithm 2 on.

Our solutions had relative error of about 10^{-7} from the lower bounds to the optimal solution given by Hexaly for the smallest problems we tested. The errors shrank further for larger problems. Fig. 8 shows that our greedy algorithm produces almost indistinguishable solutions from an exact algorithm, though it is much faster.

IV. Conclusions and Further Work

We extended the RRUC formulation from our previous work [15] to Eq. (1), which incorporates mechanical constraints and future cost considerations into UC. We consider representative demands, which we show capture the distribution well. This keeps runtime tractable, even when we increase the time granularity of the problem. We show this formulation maintains accuracy as the problem size increases and exhibits $\approx O(n^{4/3})$ runtime scaling. We introduce a provably scalable algorithm to commit hydro-generators. We verify the algorithm by checking its objective and scaling of in practice.

In future research, we intend to incorporate transmission constraints in UC. Using the insights of Fig. 6, we will incorporate batteries, including hydro-storage, in our formulation. We will attempt to leverage parallelism, including GPUs, to accelerate our solver. Finally, we will further examine RRUC's speed-accuracy trade-offs.

Acknowledgements

We thank Nicholson Koukpaizan for improving the manuscript with his review. We thank James Nutaro for helping write the proposal that funded this work.

Appendix A

We show that Algorithm 1 is asymptotically optimal for solving Eq. (1).

Theorem 1: In Eq. (1), set $\gamma = 0$. Consider a relaxed instance of Eq. (1) where $u_i \in \{0,1\} \leftrightarrow y_i \in [0,1]$. Suppose the continuous stage of Algorithm 1 solves this relaxed instance optimally, and let y_i^*, for $1 \le i \le n$, be optimal values of the y_i variables. Let P_i^* be optimal values of the P_i variables.

Suppose there is a feasible solution to Eq. (1) such that $P_i = P_i^*$ and $\{y_i^* = 1\} \subseteq \{u_i = 1\} \subseteq \{y_i^* > 0\}$. That is, $\{u_i\}$ values are rounded from $\{y_i^*\}$. Then, let ALG be the objective value of the solution to Eq. (1) produced by Algorithm 1. We have

$$\text{ALG} \le \text{OPT} + (3+s)C_{\max}.$$

with OPT the optimal objective value of the (non-relaxed) instance, s is the number of considered demand levels δ, and $C_{\max}$ is the maximum possible coefficient of any u_i in Eq. (1a). We formally define

$$C_{\max} \doteq \max_i \left\{ \max\left(a_i P_{\star,i}^2 + b_i P_{\star,i} + c_i, K_i\right) \right\},$$

$$P_{\star,i} \doteq \min(P_{\max,i}, P_{\min,i} + r_{u,i} + r_{d,i}).$$

The definition of $P_{\star,i}$ comes from the fact that any unit producing more than $P_{\min,i} + r_{d,i}$ is not discretionary, because it cannot ramp down to $P_{\min,i}$ in one step. A unit starting at $P_{i,t-1} \le P_{\min,i} + r_{d,i}$ can ramp up to $P_i \le P_{\star,i}$. $C_{\max}$'s definition is meant to upper bound the coefficient of any u_i in Eq. (1a), which we will use in the proof of Theorem 1.

Proof. First observe that, in a relaxed instance, when the non-integer decision variables are fixed, the problem is a linear program with decision variables $0 \le y_i \le 1$. We use this

fact to bound the maximum number of fractional variables $0 < y_i^* < 1$ in an optimal solution.

Observe that Eq. (1) has $3+s$ constraints that are linear in u_i. Applying a standard result in linear programming theory (see, for example, Section 17.2 of [34]), there are at least $n-(3+s)$ of the variables y_i^* such that $y_i^* \in \{0,1\}$. Therefore, there are at most $3+s$ of the variables y_i^* such that $0 < y_i^* < 1$.

Let OPT_C be the minimum objective value of the relaxed instance. Note that $\text{OPT}_C \leq \text{OPT}$. Note also that the (perhaps infeasible) solution generated by setting $u_i = 1$ if and only if $y_i^* = 1$ has an objective value at most OPT_C, since the objective is non-decreasing in u_i. It follows that any feasible solution generated from rounding the $3+s$ fractional y_i^* values has objective value at most $\text{OPT}_C + (3+s)C_{\max}$, since each fractional y_i^* value that is rounded up can contribute at most $C_{\max}$ to the final objective value.

Let R be the feasible solution assumed to exist by the theorem, and let u_i^R be its selection values. Recalling that we assume $\gamma = 0$, R will then have objective value

$$\begin{aligned} &\sum_i \left(a_i \left(P_i^*\right)^2 + b_i P_i^* + c_i\right) u_i^R u_{t-1,i} \\ &+ \sum_i K_i \left(u_{t-1,i}\left(1 - u_i^R\right) + (1 - u_{t-1,i}) u_i^R\right) \end{aligned} \tag{6}$$

Recall that Eq. (6) $\leq \text{OPT} + (3+s)C_{\max}$. Algorithm 1 will run economic dispatch on rounding R (since it runs economic dispatch on all roundings), and the resulting solution will be

$$\begin{aligned} &\sum_i \left(a_i P_i'^2 + b_i P_i' + c_i\right) u_i^R u_{t-1,i} \\ &+ \sum_i K_i \left(u_{t-1,i}\left(1 - u_i^R\right) + (1 - u_{t-1,i}) u_i^R\right) \end{aligned} \tag{7}$$

where P_i' are the generation levels set by economic dispatch. Furthermore, since the values P_i^* are feasible for Eq. (1), they are feasible for Eq. (4), and so therefore the first sum of (7) is at most the first sum of (6), since these sums are the objective function minimized by Eq. (4). The second sum is identical between (6) and (7), so (7) is at most (6).

Algorithm 1 chooses the solution which minimizes the objective value, so ALG is at most (7). Combining inequalities

$$\begin{aligned} &\text{ALG} \leq (7) \leq (6) \\ &\leq \text{OPT}_C + (3+s)C_{\max} \leq \text{OPT} + (3+s)C_{\max} \end{aligned}$$

as desired. □

Corollary 1: Under the conditions of Theorem 1, and supposing that $C_{\max}$ is fixed and $0 < \text{OPT} \propto n$,

$$\frac{\text{ALG} - \text{OPT}}{\text{OPT}} \xrightarrow[n\to\infty]{} 0$$

Proof. By Theorem 1,

$$\frac{\text{ALG} - \text{OPT}}{\text{OPT}} \leq \frac{(3+s)C_{\max}}{\text{OPT}} \xrightarrow[n\to\infty]{} 0$$

where in the last step we used that $\text{OPT} \propto n$ and that $C_{\max}$ is constant with respect to n. □

Recall that we set $\gamma = 0$ in the proof. Fig. 2 shows that $\gamma > 0$ gives better results over multiple time periods (though they may be worse in the current period). This is because we are optimizing which units operate over multiple periods, but the capacity to operate them only over the current period, because we can more easily change it between periods.

In practice, Algorithm 1 produces solutions, on average, with an upper bound on the relative error $\frac{\text{ALG} - \text{OPT}_C}{\text{OPT}_C}$ well below the theoretical bound, as shown in Fig. 9. The last few points have an average upper bound of 0 up to solver tolerance. They are assigned a small constant so they are visible on the log scale.

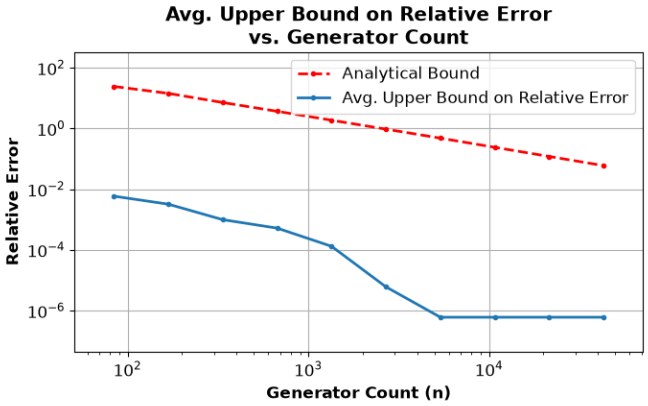

Fig. 9. Average upper bound on the relative error from the solution to the relaxed (continuous) Eq. (1) and the analytical bound over all periods vs. number of generators. The true error is below both lines.

Appendix B

Theorem 2: Solving Eq. (5) is NP-hard.

Proof. We reduce from the partition problem, which is NP-complete [35]. Given a vector $(c_1, \ldots, c_n) \in \mathbb{Z}^n$, the partition problem asks whether there exists $I \subseteq \{1, \ldots, n\}$ satisfying

$$\sum_{i \in I} c_i = \sum_{i \notin I} c_i \tag{8}$$

Without loss of generality, take $c_i \geq 0$ for all i.

Given an instance of the partition problem Eq. (8), consider an instance of Eq. (5) where $T = 2$, there are n generators, and $\kappa_j = c_j, \pi_j = 1$ for all j. Set $D_1 = D_2$ arbitrarily large. For this instance of Eq. (5), $\text{OPT} = 0$ if and only if $D_1 - \sum_i \kappa_i u_{i,1} = D_2 - \sum_i \kappa_i u_{i,2}$ if and only if $\sum_i \kappa_i u_{i,1} = \sum_i \kappa_i u_{i,2}$. This is possible if and only if there exists a set I satisfying the condition of the partition problem. □

Theorem 3: Algorithm 2 takes $O(n(\log n + T \log T))$ time, where n is the number of hydro units and T is the number of periods over which to optimize.

Proof. Lines $1-4, 24$ run in $O(n \log n + T)$ time. The outer for loop (line 5) runs n times. The inner while loop on line 7 runs $O(T)$ times, since it considers a different period on each iteration. The inner for loop (line 18) runs T times. All lines

inside loops run in $O(1)$ time, except lines 9 and 16, which run in $O(\log T)$ [36]. Overall, Algorithm 2 runs in

$$O\left(\underbrace{n\log n + T}_{\text{lines } 1-4,\,24} + \underbrace{n(T\log T + T)}_{\text{lines } 5-23}\right) = O(n(\log n + T\log T))$$

time. □

REFERENCES

[1] NERC, "Characteristics and Risks of Emerging Large Loads. Large Loads Task Force White Paper," U.S. Department of Energy, Tech. Rep., Jul 2025. [Online]. Available: https://www.nerc.com/globalassets/who-we-are/standing-committees/rstc/whitepaper-characteristics-and-risks-of-emerging-large-loads.pdf

[2] J. Sun *et al.*, "Energy dataset of frontier supercomputer for waste heat recovery," *Scientific Data*, vol. 11, no. 1077, 2024.

[3] Y. Chen *et al.*, "Security-constrained unit commitment for electricity market: Modeling, solution methods, and future challenges," *IEEE Transactions on Power Systems*, vol. 38, no. 5, pp. 4668–4681, Sept. 2023.

[4] M. Kazemi, P. Siano, D. Sarno, and A. Goudarzi, "Evaluating the impact of sub-hourly unit commitment method on spinning reserve in presence of intermittent generators," *Energy*, vol. 113, pp. 338–354, 2016.

[5] K. Kim, A. Botterud, and F. Qiu, "Temporal decomposition for improved unit commitment in power system production cost modeling," *IEEE Transactions on Power Systems*, vol. 33, no. 5, p. 5276 – 5287, 2018.

[6] F. Safdarian, A. Mohammadi, and A. Kargarian, "Temporal decomposition for security-constrained unit commitment," *IEEE Transactions on Power Systems*, vol. 35, no. 3, p. 1834 – 1845, 2020.

[7] T. Achterberg and R. Wunderling, "Mixed integer programming: Analyzing 12 years of progress," in *Facets of Combinatorial Optimization*, M. Jünger and G. Reinelt, Eds. Berlin, Heidelberg: Springer, 2013.

[8] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, 2016.

[9] E. Ridha, L. Nolting, and A. Praktiknjo, "Complexity profiles: A large-scale review of energy system models in terms of complexity," *Energy Strategy Reviews*, vol. 30, 2020.

[10] T. Berthold, *Heuristic algorithms in global MINLP solvers*. Verlag Dr. Hut, 2015.

[11] T. Koch, T. Berthold, J. Pedersen, and C. Vanaret, "Progress in mathematical programming solvers from 2001 to 2020," *EURO Journal on Computational Optimization*, vol. 10, p. 100031, 2022.

[12] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2026. [Online]. Available: https://www.gurobi.com

[13] E. C. Finardi and E. L. da Silva, "Solving the hydro unit commitment problem via dual decomposition and sequential quadratic programming," *IEEE Transactions on Power Systems*, vol. 21, no. 2, p. 835 – 844, 2006.

[14] Y. Ji and H. Wei, "An approximate dynamic programming method for unit-based small hydropower scheduling," *Frontiers in Energy Research*, vol. Volume 10 - 2022, 2022.

[15] S. Regev, E. Tsybina, and S. Peles, "A fast relax-and-round approach to unit commitment for data center own generation," 2026. [Online]. Available: https://arxiv.org/abs/2511.16420

[16] PJM, "Data miner, energy market generation offers."

[17] EIA, "EIA form 860," December 2024.

[18] ——, "Eia monthly update," 2025. [Online]. Available: https://www.eia.gov/electricity/monthly/update/print-version.php

[19] S. Andreas, K. Friedrich, M. Jan, M. Roman, and von Hirschhausen Christian, "Current and prospective costs of electricity generation until 2050." Berlin: Deutsches Institut für Wirtschaftsforschung, 2013.

[20] GE, "Ge vernova gas power products and services," 2025.

[21] PowerLine, "Reality check: Posoco report assesses the ramping capability of coal-based units, accessed 12/9/2025," 2019. [Online]. Available: https://powerline.net.in/2019/07/01/reality-check/

[22] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[23] F. Pacaud and S. Shin, "GPU-accelerated dynamic nonlinear optimization with ExaModels and MadNLP," in *Proceedings of the 63rd IEEE Conference on Decision and Control (CDC)*. IEEE, 2024, pp. 5963–5968.

[24] S. Regev, N. Y. Chiang, E. Darve, C. G. Petra, M. A. Saunders, K. Świrydowicz, and S. Peleš, "HyKKT: a hybrid direct-iterative method for solving KKT linear systems," *Optimization Methods and Software*, vol. 38, no. 2, pp. 332–355, 2023. [Online]. Available: https://doi.org/10.1080/10556788.2022.2124990

[25] K. Świrydowicz, N. Koukpaizan, M. Alam, S. Regev, M. Saunders, and S. Peleš, "Iterative methods in GPU-resident linear solvers for nonlinear constrained optimization," *Parallel Computing*, vol. 123, p. 103123, 2025.

[26] I. S. Duff, "Ma57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Trans. Math. Softw.*, vol. 30, no. 2, p. 118–144, Jun. 2004. [Online]. Available: https://doi.org/10.1145/992200.992202

[27] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[28] C. W. Clenshaw and A. R. Curtis, "A method for numerical integration on an automatic computer," *Numerische Mathematik*, vol. 2, pp. 197–205, 1960. [Online]. Available: https://doi.org/10.1007/BF01386223

[29] F. Nobile, R. Tempone, and C. G. Webster, "An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data," *SIAM J. Numer. Anal.*, vol. 46, no. 5, p. 2411–2442, Jun. 2008. [Online]. Available: https://doi.org/10.1137/070680540

[30] S. Kincic *et al.*, "Hydropower modeling gaps in planning and operational studies," Pacific Northwest National Laboratory, Tech. Rep., 11 2022. [Online]. Available: https://www.osti.gov/biblio/1922507

[31] J. Kong, I. Iliev, and H. I. Skjelbred, "Including lifetime hydraulic turbine cost into short-term hybrid scheduling of hydro and solar," *Energies*, vol. 17, no. 21, p. 5246, 2024. [Online]. Available: https://doi.org/10.3390/en17215246

[32] R. Uría-Martínez, M. M. Johnson, and R. Shan, "U.S. hydropower market report," U.S. Department of Energy, Water Power Technologies Office, Oak Ridge, TN, Tech. Rep., 01 2021. [Online]. Available: https://www.energy.gov/sites/prod/files/2021/01/f82/us-hydropower-market-report-full-2021.pdf

[33] Hexaly, *Gurobi vs Hexaly*, 2026, accessed: 2026-07-22. [Online]. Available: https://www.hexaly.com/gurobi

[34] V. V. Vazirani, *Approximation Algorithms*. Springer Publishing Company, Incorporated, 2010.

[35] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972, pp. 85–103. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9

[36] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.