5-2022

Optimization Methods for Day Ahead Unit Commitment

Jonathan David Schrock
jschroc1@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Jonathan David Schrock entitled "Optimization Methods for Day Ahead Unit Commitment." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

Jim Ostrowski, Major Professor

We have read this dissertation and recommend its acceptance:

Mingzhou Jin, Hugh Medal, Hector Pulgar

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Jonathan D Schrock entitled "Optimization Methods for Day Ahead Unit Commitment." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

Jim Ostrowski, Major Professor

We have read this dissertation
and recommend its acceptance:

Mingzhou Jin

Hugh Medal

Hector Pulgar

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

Optimization Methods for Day Ahead Unit Commitment

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Jonathan D Schrock

May 2022

I would like to dedicate this dissertation to my parents and the rest of my family for the support they have given me through all of my schooling and beyond.

And to my Tiziu who has put up with me and my never-ending schooling

Acknowledgements

Foremost, I would like to thank my advisor Dr. Jim Ostrowski for his support and guidance on my doctoral work. He guided me towards this topic, and was always willing to help me understand the material and make progress on the research. He has also helped me to get my internship and has been working with me to find employment after graduation.

I would also like to thank the rest of my dissertation committee: Dr. Mingzhou Jin, Dr. Hugh Medal, and Dr. Hector Pulgar. I'm grateful for their willingness to work with me and the insights they brought.

I am also grateful for my friends and teammates at UTK, including Amelia McIlvenna, Tony Rodriguez, Ben Knueven, Ethan Deakins, Rebekah Herrman, and Lorna Treffert.

Abstract

This work examines a variety of optimization techniques to better solve the day ahead unit commitment problem. The first method looks at the impact of almost identical generators on the problem and how to exploit that fact for computational gain. The second work seeks to improve the fidelity of the problem by better modeling the impact of pumped storage. Lastly, the relationship between the length of the planning horizon and the quality of the solutions is investigated.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The unit commitment problem (UC) is that of finding a minimum cost production schedule for a set of power generators subject to both (1) the technical constraints of the individual generators, such as minimum and maximum power output, minimum up and down time, ramping limits, and reserve qualification and (2) the system operational requirements, which include load satisfaction, transmission (deliverability) constraints, and reserve requirements (Garver, 1962; Chen et al., 2016; Anjos et al., 2017; Knueven et al., 2020). Due do to the operational characteristics of large thermal electric generators, electric grid system operators, such as the Midcontinent Independent System Operator (MISO) in the United States, must make decisions well ahead of real-time which generating units should be committed to meet the system operational requirements. Such decisions are typically made a day in advance, ensuring the following day's operational requirements will be met (Chen et al., 2014); this particular optimization problem is typically referred to as the *day-ahead* UC.

In the electricity markets in the United States and Canada, and the organized power exchanges in Europe, the system operator serves as an intermediary between generators and loads, but does not typically own any generating assets. In the context of day-ahead UC, the system operator typically operates a *day-ahead market*, which

ideally schedules generators to minimize total cost while providing a price signal to support that schedule (Johnson et al., 1997). While the purpose of the present study has little to do with price formation, it is important to note that in many contexts a sub-optimal UC solution not only wastes resources, but brings into question *market fairness* between different market participants, e.g., generator units (Sioshansi et al., 2008; Eldridge et al., 2019). Therefore obtaining high-quality solutions to the day-ahead UC problem is of upmost importance to system operators and the stakeholders they serve.

As Garver (1962) demonstrated long before it was practical, UC admits a natural mixed integer linear programming (MILP) formulation. While in the past system operators relied on approximate solution techniques to solve the day-ahead UC, advancements in MILP theory and computation (Jünger et al., 2009; Bixby, 2012) have enabled system operators to switch to off-the-self commercial MILP solvers (O'Neill, 2017). The last two decades have seen a explosion of work in finding effective MILP formulations for UC (Malkin, 2003; Lee et al., 2004; Rajan and Takriti, 2005; Carrion and Arroyo, 2006; Frangioni et al., 2009; Ostrowski et al., 2012; Morales-España et al., 2013; Damcı-Kurt et al., 2016; Wu, 2016; Silbernagl, 2016; Gentile et al., 2017; Brandenberg et al., 2017; Knueven et al., 2018c; Atakan et al., 2018; Knueven et al., 2020); i.e., MILP formulations which enable branch-and-cut solvers to rapidly find and certify high-quality solutions.

Despite the research conducted on the unit commitment problem, there are still avenues to explore in improving the solution times and quality when solving these problems. In Chapter 2, exploitation of underlying similarities between generators is leveraged to produce improved solutions quickly over the standard methods used by MISO. Chapter 3 looks at the benefits to the costs and profits of more realistically representing pumped storage hydropower units in the unit commitment model. Finally, Chapter 4 looks at the effects of varying time horizon lengths in the UC problem on solution time and quality.

Chapter 2

Exploiting Almost Symmetries Day Ahead Unit Commitment

2.1 Introduction

Symmetries within UC instances are typically caused by generating units with identical parameters. It is well known that without proper mitigation, symmetries within combinatorial optimization problems can slow down the branch-and-cut solution process by requiring the solver to explore many optimal solutions (Margot, 2002; Ostrowski et al., 2011). While modern MILP solvers employ generally-applicable advanced symmetry-mitigation techniques, these cannot capture all the primal degeneracy sometimes found in UC instances (Knueven et al., 2018b). Several methodologies have been proposed to address these additional symmetries and degeneracies, including specialized branching methods (Ostrowski et al., 2015) and (sub-)symmetry breaking inequalities (Lima and Novais, 2016; Bendotti et al., 2020). Another approach, undertaken by (Knueven et al., 2018b), and extended in the present work, involves reformulating the unit commitment problem such that identical units are aggregated into a single set of generator variables and constraints. Due to advancements in convex hull formulations for generating units (Gentile et al., 2017;

Guan et al., 2018; Bacci et al., 2019; Knueven et al., 2018a, 2020), such reformulations can be done in a way that ensures primal feasibility and optimality to the original problem. In this context, the present work makes the following contributions.

- We relax the condition that the symmetric reformulation preserve primal optimality from (Knueven et al., 2018b) and consider a *symmetric relaxation* of the original UC instance. This allows for aggregation of generators whose feasible regions are identical but with differences in costs. These differences in cost break the exact symmetry relied upon for the aforementioned symmetry mitigation techniques, but if the cost differences are slight enough, this will have a similar impact on branch-and-cut algorithms as unmitigated exact symmetry. We explore trade-offs between solution quality and unit aggregation, as well as methods for tightening the proposed symmetric relaxation.

- We demonstrate that both the exact and proposed almost-symmetric reformulations are practically applicable to large-scale real-world day-ahead unit commitment problems with the full set of operational constraints used by MISO. While transmission constraints, in general, force only co-located generators to be aggregated, we demonstrate that this is not a practical issue. Further, the considered symmetric reformulations and relaxations can improve performance 20%-25% over existing practice on a test suite of real-world MISO day-ahead UC instances.

- Most UC instances are easily solved using standard formulations. However, there are occasional instances that take much longer to solve. We demonstrate that symmetry-exploiting methods can be an effective backup for the standard approach. Solving standard formulations and symmetry-exploiting formulations concurrently can yield much faster solution times as well as cheaper solutions than simply running multiple standard formulations.

The remainder of the chapter is organized as follows. In Section 2.2 we discuss the use of MILP for solving market-clearing problems, and the impact of identical

market participants. In Section 2.3 we apply this framework to the unit commitment problem, discussing specific formulation issues for aggregating generators. Section 4.3 introduces the problem test suite and reports the main numerical results of the paper. Finally we discuss the implications of our findings in Section 4.4.

2.2 A Framework

While the focus of this work is primarily for UC models, we present the framework in terms of a more general market clearing model. Specifically, we consider problems of the form:

$$z_{clearing} = \min \sum_{p \in P} c_p(x_p, y_p) \tag{2.1a}$$

$$\text{s.t.} \sum_{p \in P} A_p x_p + \sum_{p \in P} G_p y_p \geq b \tag{2.1b}$$

$$(x_p, y_p) \in \Pi_p, \quad p \in P \tag{2.1c}$$

$$x_p \geq 0, \quad \text{integer} \tag{2.1d}$$

$$y_p \geq 0 \tag{2.1e}$$

In this context, (x_p, y_p) denotes the actions of a player p from the set of players P while Π_p represents the set of feasible actions for that player. The rows of (3.1b) represent the market products with constraints ensuring that enough of each product of is produced. The cost function $c_p(x_p, y_p)$ represents how much player p is bidding to perform (x_p, y_p).

Consider the permutation $\pi_{p,p'}$ which swaps the values of (x_p, y_p) with $(x_{p'}, y_{p'})$ respectively while keeping the remaining variables fixed.

Remark. If $A_p = A_{p'}$, $G_p = G_{p'}$, $\Pi_p = \Pi_{p'}$, and $c_p = c_{p'}$, then $\pi_{p,p'}$ is a symmetry of the market-clearing problem above. That is, swapping the values of (x_p, y_p)

with $(x_{p'}, y_{p'})$ does not change the feasibility of the solution nor does it change the corresponding objective value.

We let $\mathcal{G}_{swap}$ denote the group generated by all such permutations, i.e.,

$$\mathcal{G}_{swap} \stackrel{\text{def}}{=} \left\langle \pi_{p,p'} \mid A_p = A_{p'},\ G_p = G_{p'},\ \Pi_p = \Pi_{p'},\ \text{and } c_p = c_{p'} \right\rangle.$$

We note that while $\mathcal{G}_{swap}$ is only guaranteed to be a subset of (3.1)'s group, they are likely to be equal in any realistic application. In addition, unlike the full symmetry group, there is a simple polynomial-time algorithm to compute $\mathcal{G}_{swap}$.

The set $\mathcal{G}_{swap}$ can be used to partition the entire solution space into sets of equivalent solutions called *orbits*. The orbit of given solution to (3.1) with respect to $\mathcal{G}_{swap}$ contains that solution and any other equivalent solutions formed by any combinations of the swaps allowed by $\mathcal{G}_{swap}$.

While the permutations act on and partition vectors, they can also be used to partition the players into equivalence classes. We say that the two players share the same orbit with respect to $\mathcal{G}_{swap}$ if swapping their schedules is a symmetry. That is, p' is in the orbit of p with respect to $\mathcal{G}_{swap}$ if and only if there is a $\pi_{p,p'} \in \mathcal{G}_{swap}$. We let $\mathcal{P} = \{P_1,\ \ldots,\ P_k\}$ represent the orbital partition of the players, where each P_i represents an orbit of players. We will refer to players in the same orbit as being *identical*. Moreover, we let $\mathcal{R}$ represent the set of unique representatives for the partition $\mathcal{P}$.

In this work we examine how and when to exploit the presence of identical players. Ideally, we would like to aggregate the actions of identical players into one representative of each player orbit. This is equivalent to projecting the feasible region down to the set of representative players. To understand when this is allowable, we need to consider when a polytope has the *mixed-integer decomposition property*.

Definition. A polytope Π has the *mixed-integer decomposition property* (MIDP) if for any positive integer k and for any $(x, y) \in k\Pi$ with x integer, there exists $(x_i, y_i) \in \Pi$

with x_i integer for all $i \in \{1, \ldots, k\}$ such that $(x, y) = (x_1, y_1) + \cdots + (x_k, y_k)$. Such a polytope Π is said to be *mixed-integer decomposable*.

In the context of our market clearing model, the MIDP property ensures us that any solution found by aggregating the actions of identical players can be disaggregated into feasible solutions to the full model.

First, we show the following result:

Theorem 2.1. *Let $\mathcal{P}$ be the orbital partition of players with representative set $\mathcal{R}$.*

If Π_p is mixed-integer decomposable and c_p is linear in (x_p, y_p) for all p, then $z_{clearing}$ is equal to:

$$\min \sum_{r \in \mathcal{R}} c_r(x^r, y^r) \tag{2.2a}$$

$$s.t. \sum_{r \in \mathcal{R}} A^r x^r + \sum_{r \in \mathcal{R}} G^r y^r \geq b \tag{2.2b}$$

$$(x^r, y^r) \in |P_p| \Pi_r, \quad \textit{for } P_p \ni r, \quad \forall r \ \in \mathcal{R} \tag{2.2c}$$

$$x_r \geq 0, \quad \textit{integer } \forall \ r \in \mathcal{R} \tag{2.2d}$$

$$y_r \geq 0 \ \forall \ r \in \mathcal{R}. \tag{2.2e}$$

Note that we are using the superscript r to refer to the problem in the projected space and the subscript p to refer to the problem in its original space.

Proof. Proof: Let $(x^r, y^r)_r$ be an optimal solution to the restricted problem. We will show how $(x^r, y^r)_r$ can be disaggregated to create a feasible solution to the full model with the same objective value.

By the mixed-integer decomposition property of Π_r, for every $(x^r, y^r) \in k\Pi_r$ there exists a set of x-integral solutions $(x_r, y_r)^1, \ \ldots, \ (x_r, y_r)^k \in \Pi_r$ with $(x^r, y^r) = \sum_{j=1}^{k} (x_r, y_r)^j$. Using (2.2c), for $p_i \in P_p \ni r$ we create a solution in the original space by letting $(x_{p_i}, y_{p_i}) = (x_r, y_r)^i$. Note that $(x_{p_i}, y_{p_i}) \in \Pi_r = \Pi_{p_i}$, so constraints (3.1c) are satisfied.

Similarly, for $P_p \ni r$ we have:

$$A^r x^r + G^r y^r = \sum_{i=1}^{|P_p|} (A_r x_r^i + G_r y_r^i) = \sum_{i=1}^{|P_p|} (A_{p_i} x_{p_i} + G_{p_i} y_{p_i}),$$

since $A_r = A_{p_i}$ and $G_r = G_{p_i}$ for all p_i in P_p, implying that constraints (3.1b) are satisfied.

Furthermore, by the linearity of c_r we have that

$$c_r(x^r, y^r) = \sum_{i=1}^{|P_p|} c_r(x_r^i, y_r^i) = \sum_{i=1}^{|P_p|} c_{p_i}(x_{p_i}, y_{p_i})$$

Thus, any solution (x^r, y^r) can be disaggregated while preserving the feasibility and cost.

Similarly, any solution (x, y) can be aggregated into (x^r, y^r) by the following. For every $r \in \mathcal{R}$, let $x^r = \sum_{p \in P_p \ni r} x_p$ and $y^r = \sum_{p \in P_p \ni r} y_p$. □ □

Remark. Note that the above theorem relies on the mixed-integer decomposition property of the Π_p formulations to disaggregate the reduced solution. If Π_p is not integer decomposable then (2.2) will provide a lower bound for (3.1). Moreover, if only a subset of Π_p formulations are MIDP, then one may choose to restrict $\mathcal{G}_{swap}$ to only include permutations $\pi_{p,p'}$ where Π_p is MIDP.

Remark. A necessary, but not sufficient, condition for Π_p to admit the MIDP is that the polytope $\text{proj}_{x_p}(\Pi_p)$ is *totally unimodular* (Baum and Trotter Jr, 1978).

Remark. The aggregation can do more than just project symmetric solutions onto one representative solution. Consider an aggregated solution x^r, the aggregation of two players, with

$$x^r = (1, 2, 2, 2, 1).$$

Suppose

$$x^1 = (0, 1, 1, 1, 0), \qquad x^2 = (1, 1, 1, 1, 1)$$

is a feasible aggregation of x^r. By symmetry between players one and two, we also have that the solution

$$x^1 = (1, 1, 1, 1, 1), \qquad x^2 = (0, 1, 1, 1, 0)$$

reduces to x^r. One can think of x^r as representing both of these symmetric solutions. However, x^r can represent more than just symmetric solutions. Observe

$$x^1 = (0, 1, 1, 1, 1), \qquad x^2 = (1, 1, 1, 1, 0),$$

also reduces to x^r. While these solutions are not symmetries of the formulation group, they do have identical objective values. Such solutions may lead to more tree exploration in the branch-and-cut process. Symmetric aggregation allows us to consider these solutions simultaneously, saving computation time.

2.2.1 Differences in Bidding from Identical Players

It may be common for many players to have the same feasible region, but bid in different cost functions. In fact, it might be common to intentionally break problem symmetry by adding a small, random amount to the cost function. In doing so, the resulting MILP formulation has all of the negative aspects of a symmetric instance but without the structure to allow for any of the benefits. We look to hedge against this behaviour by generalizing the ideas of symmetry to allow us to deal

with *almost* identical players, whose swaps would result in schedules with almost identical objective costs. To do this, we relax the requirement that symmetries have to preserve objective values. Specifically, we look at permutations that are guaranteed to maintain feasibility *but not maintain optimality.*

We let $\mathcal{A}_{swap}$ denote the group generated by all such permutations, i.e.,

$$\mathcal{A}_{swap} \stackrel{\text{def}}{=} \left\langle \pi_{p,p'} \mid A_p = A_{p'},\ G_p = G_{p'},\ \text{and } \Pi_p = \Pi_{p'} \right\rangle.$$

Note that $\mathcal{A}_{swap}$ can also be thought of as the symmetry group of an MILP when the c_p functions are set to zero. Let $\mathcal{A} = \{P_1^A,\ \ldots, P_l^A\}$ be the orbital partition of players associated with the group $\mathcal{A}_{swap}$ with representative set $\mathcal{R}_{\mathcal{A}}$. Since $\mathcal{A}_{swap}$ is a relaxation of $\mathcal{G}_{swap}$, we have that $\mathcal{G}_{swap}$ is a refinement of $\mathcal{A}_{swap}$ (and $k \geq l$).

We construct a *symmetric relaxation* of (3.1) as follows. Let p be in P_i^A. We let $\underline{c}_p$ be any cost function with the property that:

$$\underline{c}_p(x_p, y_p) \leq \min \sum_{p' \in P_i^A} c_{p'}(x_{p'}, y_{p'}) \tag{2.3a}$$

$$\text{s.t.} \sum_{p' \in P_i^A} x_{p'} = x_p \tag{2.3b}$$

$$\sum_{p' \in P_i^A} y_{p'} = y_p \tag{2.3c}$$

$$(x_{p'}, y_{p'}) \in \Pi_{p'}\ \forall p' \in P_i^A. \tag{2.3d}$$

Notice that the $\underline{c}_p$ cost function is related to the entire partition P_i^A, so if $\underline{c}_p$ is valid for player p, it is also valid for player p' with p' also in P_i^A. With that in mind, the symmetric relaxation of (3.1) can be formed by assigning all players in P_i^A the same cost function:

$$z_{relaxation} \stackrel{\text{def}}{=} \min \sum_{\{r \in \mathcal{R}_{\mathcal{A}}\}} \sum_{\{p \in P_i^A \mid r \in P_i^A\}} \underline{c}_r(x_p, y_p) \tag{2.4a}$$

$$\text{s.t.} \sum_{p \in P} A_p x_p + \sum_{p \in P} G_p y_p \geq b \tag{2.4b}$$

$$(x_p, y_p) \in \Pi_p, \quad p \in P \tag{2.4c}$$

$$x_p \geq 0, \quad \text{integer} \tag{2.4d}$$

$$y_p \geq 0. \tag{2.4e}$$

This relaxation was created specifically to give all players in partition P_i^A the same objective function, meaning that permutation amongst these players are symmetries of the relaxed problem. As a result of Theorem 2.1, we have that

$$z_{relaxation} = \min \sum_{r \in \mathcal{R}_A} \underline{c}_r(x^r, y^r) \tag{2.5a}$$

$$\text{s.t.} \sum_{r \in \mathcal{R}_{\mathcal{A}}} A^r x^r + \sum_{r \in \mathcal{R}_{\mathcal{A}}} G^r y^r \geq b \tag{2.5b}$$

$$(x^r, y^r) \in |P_p^A| \Pi_r, \quad r \in \mathcal{R}_A, \quad \text{with } r \in P_i^A \tag{2.5c}$$

$$x^r \geq 0, \quad \text{integer} \quad \forall r \in \mathcal{R}_{\mathcal{A}} \tag{2.5d}$$

$$y^r \geq 0 \quad \forall r \in \mathcal{R}_{\mathcal{A}}. \tag{2.5e}$$

The hope is that by aggregating variables, the problem (2.5) can be solved significantly faster than the original model and that the lower bound, $z_{relaxation}$, will be reasonably tight. Moreover, given an optimal solution $(x^r, y^r)_r$ to this relaxation, upper bounds can be quickly obtained by solving the disaggregation problem for every $P_i^A \in \mathcal{A}$ with representative r:

$$z_r(x^r, y^r) = \min \sum_{p=1}^{|P_i^A|} c_p(x_p, y_p) \tag{2.6a}$$

$$\sum_{p \in P_i^A} (x_p, y_p) = (x^r, y^r) \tag{2.6b}$$

$$(x_p, y_p) \in \Pi_p \quad p \in P_p^A \tag{2.6c}$$

$$x_p \geq 0, \quad \text{integer} \ , \ \forall p \in P_p^A \tag{2.6d}$$

$$y_p \geq 0 \quad \forall p \in P_p^A, \tag{2.6e}$$

where the upper bound has objective value $\sum_{r \in \mathcal{R}_\mathcal{A}} z_r(x^r, y^r)$. Similarly, the (x_p, y_p) values in the disaggregate give the solution.

Theorem 2.2. *Let $(x^r, y^r)_r$ be the optimal solution to $z_{relaxation}$. We have that $z_{relaxation} \leq z_{clearing} \leq \sum_{r \in \mathcal{R}_\mathcal{A}} z_r(x^r, y^r)$.*

Proof. Proof: $z_{relaxation} \leq z_{clearing}$: This is clear when solving the disaggregated model, (2.4), for $z_{relaxation}$, as the only difference between this and the original model is that the cost function for (2.4) is not greater than the cost function for (3.1). $z_{clearing} \leq \sum_{r \in \mathcal{R}_\mathcal{A}} z_r(x^r, y^r)$: Each solution in $z_{relaxation}$ is feasible for (3.1), thus the true cost of each solution must be at least as large as $z_{clearing}$. □ □

The quality of such bounds are directly related to the quality of $\underline{c}_p$ as an underestimator of the true cost function.

Theorem 2.3. *If $\underline{c}_r(x^r, y^r) = z_r(x^r, y^r)$ for all $r \in \mathcal{R}_\mathcal{A}$, then we have that $z_{relaxation} = z_{clearing}$.*

Proof. Proof: This is easy to see by observing that the above condition is equivalent to stating that

$$z_{relaxation} = \sum_{r \in \mathcal{R}_\mathcal{A}} z_r(x^r, y^r). \square$$

□

Theorem 2.4. *If the feasible region to (2.6) is integer for all integer x^r inputs for all $r \in \mathcal{R}_A$, then there exists a piecewise linear function $\underline{c}$ s.t. $z_{relaxation} = z_{clearing}$.*

Proof. Proof: This can be seen by noting that if (2.6) is integer, then strong duality must hold. Note that the terms (x^r, y^r) will appear only on the objective function to the dual problem. As the cost of any dual feasible solution is linear in (x^r, y^r), taking the maximum of the finitely-many dual vertices will result in a cost function that is piecewise linear in (x^r, y^r).□

□

The integrality of (2.6) for every integer x may seem like an overly strong condition. However, if the c_p functions are linear, the integrality of (2.6) is a consequence of MIDP. The trick is in ensuring that nonlinear cost functions are modeled as tightly as possible.

Note that the proof of Theorem 2.4 gives a recipe for a Benders-type approach to improve the gap between $z_{relaxation}$ and $z_{clearing}$. If the difference between the relaxed cost $\underline{c}_r(x^r, y^r)$ is significantly less than the actual cost $z^r(x^r, y^r)$ then one can use the formulation (2.6) to generate cuts that improves the relaxed cost function.

2.3 The Unit Commitment Problem

The UC problem is the problem of minimizing the production cost of a set of power generators such that demand can be satisfied. The transmission network is represented by a flow network where each node has its individual demand over time and each edge has a given capacity. Generators are located at various nodes throughout the network. Transmission obeys alternating current (AC) power flow, which introduces a set of highly nonlinear constraints that represent how physics governs power flows through a network. These constraints, however, make real-world UC problems computationally intractable. As a result, linear approximations are typically used in practice. In the context of market clearing models, the transmission

network is represented by the set of equations (3.1b) while Π_g represents the feasible production schedules of generator g. The region Π_g is known as the *generator polytope*, something that has been well studied in the UC literature. Common constraints in Π_g include minimum up- and down-times, minimum and maximum power outputs, and ramping constraints. There are three different costs for operating a generator. The first is based on production, which is typically modeled as an increasing piecewise linear cost, the second is a fixed-running cost for commitment (no load cost), and the third represents the cost of starting a generator. Startup cost is usually a non-decreasing function of the generator off-time. A key point is that, for these generators, formulations of Π_g that have the MIDP property are known, Knueven et al. (2018b). Some generators have additional physical constraints such as a maximum daily production limit or a maximum number of startups in one day. These additional constraints may break the MIDP structure of the generator polytope, but fortunately, they are not commonly seen in practice. This can be easily dealt with by not aggregating generators with these additional constraints.

2.3.1 Symmetries in Unit Commitment

While it is possible for some symmetry to exist in the transmission network, the predominant source of symmetry in unit commitment comes from having identical generators. Specifically, generators that have identical:

- Production Costs
- Startup Costs
- Minimum Up and Down Times
- Ramping Rates
- Minimum and Maximum Power Output
- Impact on the Transmission Network.

Such generators do exist in practice as identical generators tend to be co-located and owned/operated by the same entity.

2.3.2 Almost Symmetries in Unit Commitment

Using orbits allows us to aggregate variables to reduce the size of the optimization model. However, there are strict requirement for two generators sharing the same orbit. Consider two generators with the same physical constraints, but have costs that are slightly different. We approach this by creating a *symmetric relaxation* of the UC model. Now, we partition the generators into $\mathcal{A} = \{P_1^A, \ \ldots, \ P_k^A\}$, where two generators g and g' are in a set P_j^A if and only if they have identical:

- Minimum Up and Down Times
- Ramping Rates
- Minimum and Maximum Power Output
- Impact on the Transmission Network.

The quality of our relaxed model is directly related to how we construct the $\underline{c}$ function for each set of nearly identical generators. Theorem 2.4 indicates that we can find the most accurate $\underline{c}$ by projecting the disaggregation problem onto the set of aggregated variables and aggregate cost function. However, this approach is computationally intractable. A more practical approach is to find a simple under-approximation and seek to iteratively improve it.

The simplest way of creating a symmetric relaxation is that for every almost-identical generator and for every type of cost, assign it the smallest value of each of the generators in the partition. We refer to this as the *naive cost reduction*. We note that the cost function associated with the naive cost approach is convex piecewise linear.

A natural approach would be to start with the naive cost model and then use cut generation to iteratively improve the approximation. Benders' decomposition

could be used to improve the quality of $\underline{c}$ through the generation of optimality cuts. However, using callbacks turns off advanced features in commercial solvers. Preliminary computational experiments demonstrated the impact of losing these advanced features outweighed the benefit of exploiting symmetry. Instead, we considered how to quickly and easily produce high quality cuts. In the data that we considered, we saw difference in all three different types of costs. We propose the following set of strategies to generating high quality $\underline{c}$ functions.

Ignore small differences: Many times the differences in no load costs and production costs are trivial with respect to the size of the model. Approximately half of all difference in no-load costs amongst nearly identical generators are less than \$50 per time period. Similarly, nearly half of the difference in production costs are less then a \$5 per hour. On the scale of the UC solutions, these do not matter in the objective and adding additional constraints to represent these costs only slows down the model with no benefit.

Break Partitions with Large Costs: Similarly, there are sets of generators that have significant differences in costs. This is especially true when considering startup costs. Rather than attempt to incorporate these difference in the $\underline{c}$ function, we simply break the partition and treat them as separate units. This does not end up hurting computationally, as this usually happens when a specific generator is extremely expensive and will easily be determined to be off in either pre-processing or in the linear relaxation. This can happen when a generator owner intends to perform maintenance on the generator unless the owner can profit considerably.

Piecewise Linear Cost Constraints: For generators whose price differences are not negligible, but also not extreme, we add piecewise linear cost constraints for both the non-negligible production and no load constraints. This is done by simply ordering the costs from smallest to lowest and assuming that the lowest cost is always chosen. For example, given three generators with no load costs of \$100, \$150, and \$200, we assume that if one generator is on, then the no load cost is \$100, if two generators are on then the no load cost is \$250, and if all three are on then the no load cost is

$450. This can be enforced by a simple piecewise linear cost function. Production costs are handled in a similar way.

2.4 Computational Results

To explore the impact of symmetries and almost symmetries in unit commitment problems, we examined eleven real-world instances seen by MISO in operation. Nine of these instances occurred during the 2014 polar vortex. These specific instances were chosen because they were examples of the most notoriously difficult problems seen by MISO in the past decade. Interestingly, using advanced UC formulations (Knueven et al., 2020) has greatly improved our ability to solve some of these problems. However, a subset are still very difficult. The remaining two instances were randomly chosen from the set of typical days. These instances were chosen and pre-screened by MISO.

While many parameters of the UC model such as cost and demand change from day to day, generator data, such as symmetry, tends to be relatively consistent across days. Table 2.1 gives the description of the symmetry information for a representative instance. The composition of pure orbits may vary slightly in different instances depending on the prices bid into the market, but the almost data is consistent among instances (though some generators do not always participate in the day ahead market). Unsurprisingly, most of the generator orbits are small, consisting of just pairs of identical generators, though there are some larger orbits. Relaxing from pure orbits to almost orbits does add a significant amount of symmetry, there are 41 pure orbits representing 119 generators while the 86 almost orbits represent 229 generators. Using cost cutoffs to break up almost orbits does not significantly change the amount symmetry found in the problem. While the concept of symmetry in UC has been studied before (Ostrowski et al., 2015; Lima and Novais, 2016; Knueven et al., 2018b; Bendotti et al., 2020), it has always been in the context of a copper-plated system, that is, a UC model without transmission constraints. It is interesting that

Table 2.1: Number of orbits by orbit size in a problem from our test suite.

Model / Orbit Size	2	3	4	5	6	7	8
Exact symmetry	22	12	3	1	0	2	1
Almost symmetry	59	11	11	0	2	2	1
Almost symmetry w/ cost cutoff	54	11	8	1	1	2	1

so much symmetry exists in real-world instances even after transmission constraints are included.

We emphasize that the difference between pure symmetries and almost symmetries are due only to the bidding behavior of generator owners. Pure symmetry can easily be broken by making small changes to any of the costs.

2.4.1 Experimental Setup

Testing of the above additions to the UC model were performed on several real world instances of the UC problem provided by MISO. We compare various models to the standard UC "Tight" formulation from Knueven et al. (2020). They are described as follows:

- Base (BA): The standard formulation used in practice, no consideration of symmetry except for Gurobi's internal methods.
- Pure Symmetry (PS): Aggregates generators that are identical.
- Naive Almost Symmetry(NAS): Aggregates generators that are identical after ignoring all costs.
- Cost Cutoff (CC): Aggregates generators are identical after ignoring small differences in costs.
- Tightened Cost Cutoff (TCC): Symmetry preserving constraints are added to CC to improve the relaxation.

We note that models BA and PS will always return the same solution value by (Knueven et al., 2018b) and construction (we do not aggregate units whose formulation is not known to have the MIDP). The solving of NAS will always return the weakest lower bound, followed by CC, then by TCC. Even though the formulation for BA does not consider symmetries, we note that these symmetries are still exploited in the solver, as Gurobi can mitigate symmetries in branching . However, there is

no such method to exploit almost symmetries. If generator owners chose to add random, small, perturbations to the prices of identical generators, neither Gurobi's native methods nor the PS method would be able to identify and exploit this almost symmetry, leading to a potential explosion in computational times for these methods.

When solving NAS, CC, and TCC, a standard model with some modifications was used to produce a disaggregated solution of the almost symmetric model. The time needed to disaggregate the solution was trivial and is not reported. For each instance several models were created and run 10 times with different seed values. These models all contained full transmission security constraints, and were solved using the larger of a 0.25% or $12,000 gap using Gurobi 8.1.1 with a time limit of 12,000 seconds. Computations were done on a Linux server hosted by MISO with 32GB of RAM and a 12-core Intel processor running at 2.5GHz.

Table 2.2 shows the average over ten random seeds of time required to solve the UC instances using each of the methods The results show exploiting symmetries can have a positive impact on overall solution times, but that a naive aggregation may not always produce the best results. Using pure symmetries does help in the hardest instances, but does not always offer improvements. This makes sense when considering what makes UC instances difficult. In any instance, the on/off status for most generators will be clear from the linear relaxation. Cheap generators will always be on, expensive generators will always be off. The computational difficulty comes from choosing which of the (near-)marginal generators, those with fractional on/off variables, are on at what times. If the set of marginal generators contains a nontrivial orbit, then choosing which of the marginal generators are on can be difficult. For example, suppose you knew that two out of a set of three generators should be on. Choosing any pair of those generators to be on would result in an equivalent solution, and thus be harder to prune from the search. The success in PS occurs in the difficult problems, specifically Instance 8, where this happens. The set of marginal units contains a nontrivial orbit, and the PS method is able to efficiently reduce the search over that orbit. We would expect no impact, like Instance 1 for

Table 2.2: Times to solve UC for each method measured in seconds

Instance	BA	PS	NAS	CC	TCC
1	123	121	115	113	132
2	280	294	285	292	307
3	182	397	147	156	346
4	1,200	1,249	1,222	1,202	1,150
5	3,650	3,100	2,651	2,798	3,324
6	831	823	704	735	824
7	820	762	939	787	878
8	3,139	1,428	3,305	1,275	1,345
9	1,414	1,393	1,431	1,345	1,042
10	164	166	124	150	168
11	125	131	121	119	140
Total	11,927	9,864	10,105	8,972	9,656

example, where the set of marginal generators does not include a nontrivial orbit. Interestingly, however, the naive aggregation (NAS) under performs the PS method. This seems counter intuitive as it is more likely that the set of marginal generators consists of almost-identical generators. We think there are two explanations for this behavior. First, that more generators are marginal in the NAS method. By reducing the costs of some expensive generators, some generators that were clearly off now become marginal. So, while the set of marginal generators do contain groups of almost-identical units, the benefit of exploiting that structure does not outweigh the cost of increasing the set. Secondly, recall that the integrality gap was the maximum of 0.25% and \$12,000. For all but two instances, the relative gap of 0.25% was the binding gap. Reducing the cost of a subset of generators can noticeably decrease both the lower bound and the optimal objective value. In practice, this lowers the absolute optimality gap needed, increasing solution times.

Both CC and TCC are meant to avoid both of the drawbacks of NAS at the expense of a more conservative aggregation procedure. By performing CC, we limit the changes in costs for each generator, avoiding the issues of adding to the set of marginal units as well as limiting the change in bounds. Despite this limit, CC may still be too much of a relaxation, so TCC is developed by adding additional constraints that are meant to increase the quality of the relaxation (increase dual bounds) while preserving the symmetry. Despite the fact that these constraints are sparse and that there are not too many of them, adding them does impact the solution time over CC.

We must note that while CC tends to solve the fastest overall, there is no guarantee that the solution obtained after disaggregation is within the specified tolerance gap. Table 2.3 show how often the symmetric relaxations provide a provably near-optimal solution. In this table, we compute the optimality gap to be the difference between the disaggregated solution's true cost and the lower bound provided by the symmetric relaxation. As expected, increasing the quality of the relaxation increased the quality of the solution (and it's bound).

Table 2.3: Successful gaps out of ten random seeds

Instance	NCA	CC	TCC
1	100%	100%	100%
2	0%	0%	90%
3	100%	100%	100%
4	100%	100%	90%
5	50%	30%	70%
6	70%	70%	20%
7	0%	80%	100%
8	0%	20%	50%
9	0%	100%	100%
10	0%	100%	100%
11	100%	100%	100%
Average	47.3%	74.5%	83.6%

Ensemble of Models

Recent advances in integer programming have made most UC instances reasonably easy. In the instances considered in this work, five of the eleven instances are easily solved within five minutes. Our experience in work with real data is that improvements in modeling and algorithms is not likely to produce any meaningful impact on most daily instances. The concern for ISOs is not about reducing the computational time needed for these easy instances, but in minimizing the impact of the hard instances. We classify four of the instances studied, Instance 4, 5, 8, and 9, as hard, as they take on average 20 minutes or longer to solve. The TCC model has the best average solution times in three of these four instances compared to BA and PS. A closer look at Instance 5, the only hard instance TCC does not win at, reveals that the average computational times for both BA and TCC are driven by anomalous behavior, one seed in TCC taking approximately 7,500 seconds and one seed in BA taking over 9,000 seconds. It is not entirely clear why this particular instance has such extreme behavior. If we were to consider it chance that BA and TCC experienced this anomalous result and PS did not, then by ignoring those two runs would pull down the average of the TCC instances to 2,864 seconds and the BA instances to 3,031 seconds, both faster than PS, with TCC being the fastest.

Given that BA will do well on most instances, it makes sense to consider alternative approaches as backups to BA, to be run in parallel and there to offer a solution when BA experiences hard instances. Running different algorithms concurrently not only has the benefit of improving solution times, but can also be used to improve the bounds. Even if a particular method fails to solve an instance in a given amount of time, it might have found a better solution or improved the lower bounds more so than the winning algorithm. In Table 2.4 we show the potential impact of running BA and TCC concurrently both in terms of improved speedup but also in terms of improved solutions. For a fair comparison, we compare this to running two seeds of BA concurrently. For these results, we let both seeds run to completion if their

Table 2.4: Impact of concurrently running BA and TCC versus 2 BA instances

	Time Improvement (s)		Cost Improvement ($)	
Instance	2BA	BA + TCC	2BA	BA + TCC
1	3.33	0.14	611.43	613.27
2	5.75	0.70	79.80	718.92
3	5.71	0.00	0.00	1,872.48
4	44.14	104.74	121.98	1,716.48
5	757.06	886.35	273.70	1,084.04
6	18.62	27.95	68.02	0.00
7	39.91	20.46	352.20	787.39
8	67.19	1,793.64	149.80	351.33
9	380.39	453.01	263.67	1,020.55
10	5.02	2.74	253.66	31.27
11	2.02	0.00	0.00	0.00
Total	1,329.13	3,289.73	2,174.26	8,195.74

completion time is under 20 minutes. After 20 minutes has passed, we halt both seeds as soon as one solves to optimality.

As the table indicates, running TCC concurrently with BA can offer significant protection in solving hard instances. Most notably, instance eight is solved 30 minutes faster on average. As expected, there is little to no improvement in solution times for the easy problems. However, concurrently solving can significantly impact the overall solution quality of even the easy instances. For example, while no seed from Instance 3 is solved faster using TCC, all seeds for this instance are solved within 20 minutes, and solving concurrently leads to an average cost improvement of approximately $1,700. While both cost and time savings are seen when solving two BA seeds concurrently, the extent of the savings does not compare to BA with TCC.

2.5 Discussion and Conclusions

Standard UC models, combined with modern MILP solvers are very good at solving most day ahead problems. However, there are always some days where these models take much longer than normal to solve. Our results indicates that one explanation for why some instances take much longer to solve than other lies with the composition of identical and nearly identical generators. Deciding which of a set of identical or nearly identical generators is on is difficult without symmetry-exploiting techniques. This inspires this use of symmetry/almost-symmetry exploiting models that, when used concurrently with standard models, are able to provide a backup to standard models, providing solutions when standard models run the risk of timing out. At the same time, this also exposes a potential flaw in the current day ahead market. By bidding with the specific intent to break symmetry, generator owners could potentially create situations where the day ahead model is not solvable in the allowed window. This would lead to using higher cost schedules, possibly allowing the generator owners to achieve higher profits.

We believe that the importance of these symmetry-exploiting methods will grow as the scale of day ahead UC models increase. Moving to longer time horizons (ex. 48 hours) and/or shorter time intervals (ex. 15 minutes) will stress standard models. At the same time, the symmetry-exploiting methods can continue to be improved as more instances will help us learn the tradeoff between tighter models and improved computational time.

Chapter 3

Pumped Storage Hydropower in Unit Commitment

3.1 Introduction

Pumped-storage hydropower (PSH) is a type of hydroelectric energy storage. These systems consist of two water reservoirs at different elevations to generate power (discharge) as water moves from the upper reservoir to the lower through a turbine; they draw power as water is pumped (charge) back to the upper reservoir. PSH systems can be classified as open loop, where a reservoir is connected to a natural body of water, or closed loop, where the reservoirs are isolated from outside sources of water. PSH units currently account for 95% of large-scale energy storage in the United States U.S. Department of Energy (2021).

Incorporating PSH constraints in the UC model is not a new idea. Many models have looked to incorporate PSH units as reserve for other renewable energy sources Bruninx et al. (2016); Brown et al. (2008); Jiang et al. (2012). Most of these models provide similar constraints to capture the charge/discharge behavior of PSH units; tracking the amount of energy stored in the reservoir and the operating states of the turbines and pumps Borghetti et al. (2008); Bruninx et al. (2016); Brown et al.

(2008); Jiang et al. (2012); Van den Bergh et al. (2014). Though some models do not prevent the system from charging and discharging simultaneously Brown et al. (2008); Van den Bergh et al. (2014), most include constraints to prevent this from occurring. While PSH formulations exist, they may not be used in practice. MISO, for instance, does not currently include advanced PSH models. Instead, they attempt to treat PSH generation as conventional generators. Doing so has the benefit of being easier to implement and makes the resulting MILP formulation easier to solve. However, treating PSH as conventional generation might lead to schedules that are not actually feasible and could lead to inefficient production schedules. Improvements in solving the current UC models, as cited in chapter 1, allows ISO's to investigate the impact of incorporating more detail in the models.

In this work, we show that more advanced PSH models do not significantly impact the time to solution on large-scale instances like those solved at MISO. However, one difficulty with the improved modeling is related to end of time horizon affects. UC instances are solved over a 36-hour time horizon. If optimized myopically, solutions would exhaust the stored energy in PSH units, leading to more expensive future days. We investigate how to use forecasted data to deal with these end of time horizon effects.

The remainder of the chapter is organized as follows. In Section 3.2 we introduce the PSH constraints added to MISO's unit commitment model. Section 4.3 introduces a variety of strategies to deal with the end of time horizon affects. Finally we discuss the implications of our findings in Section 4.4.

3.2 PSH Model In Unit Commitment

The prototypical unit commitment formulation is given as

$$z_{UC} = \min \sum_{g \in G} c_g(u_g, p_g) \tag{3.1a}$$

$$\text{s.t.} \sum_{g \in P} A_g(u_g, p_g) = D \tag{3.1b}$$

$$(u_g, p_g) \in \Pi_g, \quad g \in G \tag{3.1c}$$

$$u_g \geq 0, \quad \text{integer} \quad \forall g \in G \tag{3.1d}$$

$$p_g \geq 0, \quad \forall g \in G, \tag{3.1e}$$

where the cost of production, $c(*)$, over a collection of generators, G, is minimized in such a way that demand is met and each generator's production schedule, (u_g, p_g), is feasible. In this formulation, Π_g denotes the feasible region for each generator g, while (u_g, p_g) denotes its production (both the discrete and continuous parts). While G typically contains a variety of different types of generators, this paper focuses specifically on how to model PSH units.

Current practice: Modeling PSH via maximum daily energy constraints

Current MISO formulations decouple the charging and discharging decisions of PSH units. In doing so, the discharging (production) can be treated as a generic thermal generator. Charging decisions are treated as virtual bids. While fitting into the existing UC framework, decoupling the charging and discharging decisions may lead to infeasible schedules, as there is no way to ensure that the storage levels remain feasible (as there are typically minimum and maximum water levels that need to be maintained). As a proxy for maintaining the minimum water levels, *daily energy constraints* are included in the generator model that are designed to limit the total energy produced in a day. They have the form

$$\sum_{t \in T} p_{g,t} \leq Max\_Daily,$$

when g is a PSH unit. Note that these constraints are placed on each PSH unit.

Most generators associated with thermal units have large ramping capabilities, enough so that ramping constraints typically found in generator formulations become redundant. Technically, this occurs when the ramping capability exceed the difference between the maximum and minimum production levels. The formulations of generic generators without ramping constraints are perfect formulations. However, adding constraints of the form (3.2) breaks the perfect formulation, weakening the quality of the formulation.

Proposed Method: Modeling via State of Charge

We propose to replace the current formulation with a typical *state of charge formulation* that couples the charging and discharging decisions by explicitly modeling the state of charge at every time interval. This formulation for a given pump is:

$$s_{g,t+1} = s_{g,t} + q^{in}_{g,t}\sqrt{\epsilon_g} - \frac{q^{out}_{g,t}}{\sqrt{\epsilon_g}}, \qquad (\forall t, \forall g \in G) \tag{3.2a}$$

$$s_{g,1} = S^{init}_g, \qquad (\forall g \in G) \tag{3.2b}$$

$$\underline{S}_g \leq s_{g,t} \leq \overline{S}_g, \qquad (\forall t, \forall g \in G) \tag{3.2c}$$

$$0 \leq q^{in}_{g,t} \leq \overline{P}^{in}_g z_{g,t}, \qquad (\forall t, \forall g \in G) \tag{3.2d}$$

$$0 \leq q^{out}_{g,t} \leq \overline{P}^{out}_g u_{g,t}, \qquad (\forall t, \forall g \in G) \tag{3.2e}$$

$$u_{g,t} \leq (1 - z_{g,t}), \qquad (\forall t, \forall g \in G) \tag{3.2f}$$

$$u_{g,t}, z_{g,t} \in \{0, 1\}, \qquad (\forall t, \forall g \in G) \tag{3.2g}$$

Here t is over the number of time periods considered, and G is the set of PSH units. The variable $s_{g,t}$ tracks the state of charge of the reservoir that unit g sits on with ϵ_g as the round-trip efficiency of the unit. $q^{in}_{g,t}$ is the amount unit g is charging at a given time period, and $z_{g,t}$ indicates whether the unit is pumping at time t. Finally, $u_{g,t}$ indicates whether unit g is producing at time t, and $q^{out}_{g,t}$ is the amount of power g produces.

Similar to the maximum daily model, the state of charge constraints make the models more difficult, a perfect formulation is likely too large to be computationally tractable.

Exploiting Symmetry in PSH Models

Constraints in (3.2) assume that each generator is associated with its own reservoir. It is common that one reservoir will have multiple pumps and generators associated with it. In this case, the state-of-charge constraints are a function of the aggregate generation/pump. Let R denote the set of reservoirs in the model and G_r denote the set of PSH units associated with reservoir $r \in R$. We can model the aggregate of the PSH units as

$$s_{r,t+1} = s_{r,t} + \sqrt{\epsilon_g} \sum_{g \in G_r} q^{in}_{g,t} - \frac{1}{\sqrt{\epsilon_g}} \sum_{g \in G_r} q^{out}_{g,t}, \qquad (\forall t, \forall r \in R) \tag{3.3a}$$

$$s_{r,1} = S^{init}_r, \qquad (\forall r \in R) \tag{3.3b}$$

$$\underline{S}_r \leq s_{r,t} \leq \overline{S}_r, \qquad (\forall t, \forall r \in R) \tag{3.3c}$$

$$0 \leq q^{in}_{g,t} \leq \overline{P}^{in}_g z_{g,t}, \qquad (\forall t, \forall r \in R, \forall g \in G_r) \tag{3.3d}$$

$$0 \leq q^{out}_{g,t} \leq \overline{P}^{out}_g u_{g,t}, \qquad (\forall t, \forall r \in R, \forall g \in G_r) \tag{3.3e}$$

$$u_{g,t} \leq (1 - z_{g,t}), \qquad (\forall t, \forall r \in R, \forall g \in G_r) \tag{3.3f}$$

$$u_{g,t}, z_{g,t} \in \{0, 1\}, \qquad (\forall t, \forall r \in R, \forall g \in G_r) \tag{3.3g}$$

The advantage of this formulation is that we can use the potential symmetry in the model to aggregate decision variables. Knueven et. al. Knueven et al. (2018b) shows that variables associated with identical generators can be aggregated if they admit a perfect formulation. This is the case in formulation (3.3), as long as each PSH unit in G_r has similar charging/discharging capacities and efficiencies. As such, the entire collection PSH units can be modeled as

$$s_{r,t+1} = s_{r,t} + q^{in}_{r,t}\sqrt{\epsilon_r} - \frac{q^{out}_{r,t}}{\sqrt{\epsilon_r}}, \qquad (\forall t, \forall r \in R) \tag{3.4a}$$

$$s_{r,1} = S^{init}_r, \qquad (\forall r \in R) \tag{3.4b}$$

$$\underline{S}_r \leq s_{r,t} \leq \overline{S}_r, \qquad (\forall t, \forall r \in R) \tag{3.4c}$$

$$0 \leq q^{in}_{r,t} \leq \overline{P}^{in}_r z_{r,t}, \qquad (\forall t, \forall r \in R) \tag{3.4d}$$

$$0 \leq q^{out}_{r,t} \leq \overline{P}^{out}_r u_{r,t}, \qquad (\forall t, \forall r \in R) \tag{3.4e}$$

$$u_{r,t} \leq (|G_r| - z_{r,t}), \qquad (\forall t, \forall r \in R) \tag{3.4f}$$

$$0 \leq u_{r,t} \leq |G_r|, \qquad (\forall t, \forall r \in R) \tag{3.4g}$$

$$0 \leq z_{r,t} \leq |G_r|, \qquad (\forall t, \forall r \in R) \tag{3.4h}$$

Note that in this model only one integer variable is needed per time period to reflect the charge/discharge status of the entire collection of identical PSH units associated with the same reservoir.

Challenges with State-of-Charge Model

End-of-time-period affects can greatly impact the quality of solution produces by a UC model. MISO typically solves 36-hour UC instances, even though they intend to use only the first 24 hours. The additional 12 hours are included in the model to ensure that decisions aren't made too myopically; that the impact of tomorrow's schedule on future costs is considered in the decision making. Unfortunately, because PSH units are limited by their charging behaviour, decisions made tomorrow can have a significant impact on power production several days in the future. To account for this, we need to account for the state of charge at the end of the planning horizon in the model. We have three proposed strategies for dealing with this end-of-time-horizon affect.

Base: The first method is straight forward; the PSH constraints are added with no day to day expectations on the state of charge of the reservoirs. That is, we only provide the initial state on the first day and the required state on the final day.

Each intermediate day only has the requirement that its initial state of charge is the final charge of the previous day. The advantage of this model is that no additional information is needed.

Naive: The next method is a naive approach. Here we begin with a full week model that is relaxed and solved. This model allows us to calculate an estimate of the state of charge for the end of a day-ahead model using knowledge of the full week. The state of charge at hour 24 is then fixed in the day-ahead model which is solved, and the state of charge values at the end of the day are stored. A 6-day model is then constructed from the following day to the end of the week with the stored state of charge values used as the initial charges. This transition to a 6-day model as opposed to another 7-day model is due to restrictions in our access to MISO's data. The 6-day model is solved and the state of charge at hour 24 in the corresponding day-ahead model is fixed. This process continues through the week. In short, the multi-day model is used to determine a fixed state of charge for PSH p at time 24, $SOC_{p,fixed}$, and the following constraint is added to the day-ahead model:

$$soc_{p,24} = SOC_{p,fixed}.$$

Dual Pricing: The final method also relies on multi-day models, but with a soft constraint for the state of charge at time period 24. We still compute $SOC_{p,fixed}$ as in the naive method, but we also attempt to price deviations from the fixed amount. In the linear relaxation to the multi-day model, we compute the dual price associated with the state of charge constraint on the PSH unit p. Denote this price as π_p. We then add the following constraint to the day-ahead model:

$$t_1 - t_2 = SOC_{p,fixed} - soc_{p,24}$$

$$t_1, t_2 \geq 0$$

where t_1 and t_2 are new variables each with a cost of π_p in the objective function. These variables effectively penalize any deviation from the state of charge calculated by the linear relaxation but still permit it.

3.3 Computational Results

Our goal with this work is to see if there is any cost or profit benefits to more explicitly representing PSH units in MISO's UC model. To that end, we incorporate PSH constraints into the existing model, and look at different ways of giving the day-ahead model expected end of day charge values for the reservoirs. We need to make these comparisons across a week since the reservoirs in our models are expected to be full at the beginning of a week.

The following figures shows the price and charge schedules for a PSH unit in a typical summer week, Figure 3.1. Notice that every day the unit is able to fully cycle, discharging during the expensive peak times and fully charging at night when the price of electricity is cheap. In these weeks, the value of future information is not important for determining the optimal state of charge, so any of the three methods should return similar, if not identical results.

This "full-cycle" behavior is not always seen, however. Some times, pumps undergoing maintenance may not allow for the PSH unit to fully charge every night. In other instances, extreme weather events may break the cyclical pattern. Figure 3.2 gives the prices at a given node experienced during a polar vortex. In this case, using predictions of future behavior can have a significant impact on how we manage the PSH units.

3.3.1 Experimental Setup

For each of our state of charge methods, first a linear relaxation of a multi-day model is solved to compute $SOC_{p,fixed}$ and the dual price π_p. Then the model for the day

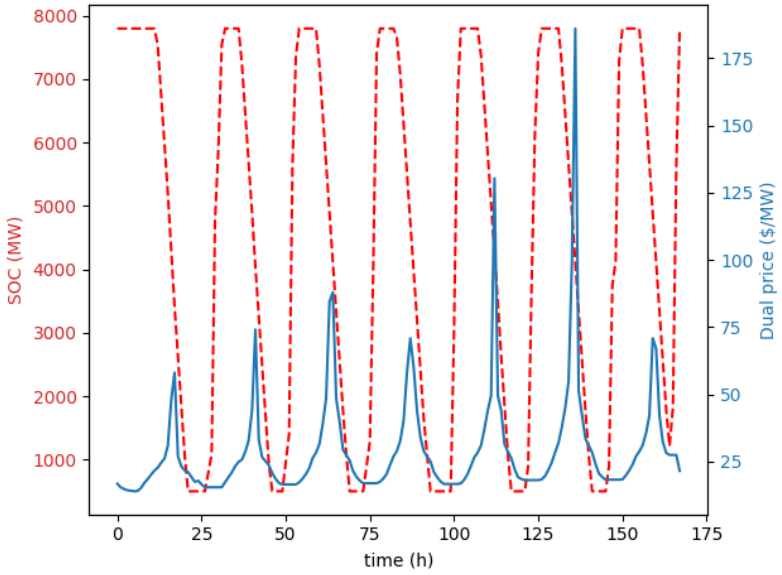

Figure 3.1: The week-long charging schedule for a PHS unit in week 1

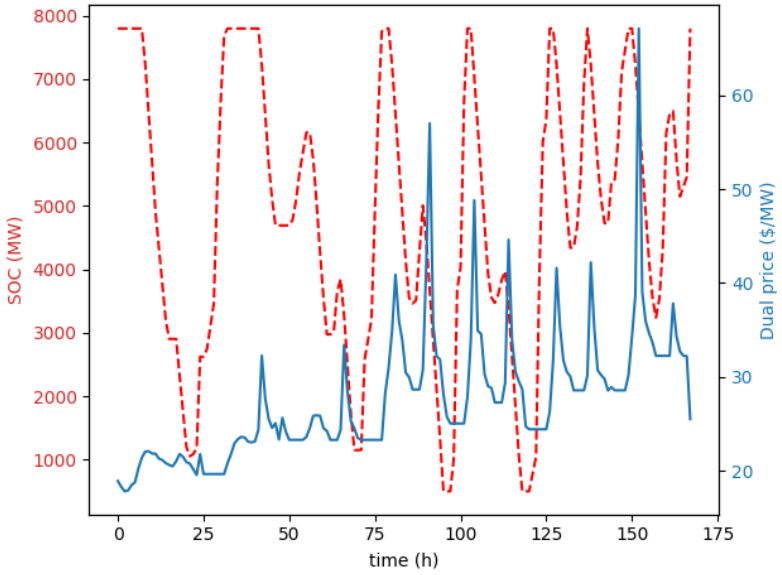

Figure 3.2: The week-long charging schedule for a PHS unit in week 2

under consideration is built and solved employing $SOC_{p,fixed}$ and/or π_p depending on the method used. The solution to this day-ahead model is used to calculate the 24 hour costs for the day and the profit on the PSH units as well as the initial state of charge for the following day.

For testing, each day-ahead model contained full transmission security constraints, and were solved using the larger of a 0.25% or $12,000 gap using Gurobi 8.1.1 with a time limit of 12,000 seconds. Computations were done on a Linux server hosted by MISO with 32GB of ram and 12-core Intel processor running at 2.5GHz. The multi-day models were run on the same system with the same relevant parameters. These multi-day models were built by combining data from several consecutive 36-hour instances. In each of these 36-hour daily instances, the last 12 hours were dropped and the days concatenated. We omitted the virtual bids and demand responses and limited the number of transmission constraints in our multi-day models; otherwise the corresponding linear program would be too large to store in memory/solve.

To test the effectiveness of our different methods, we looked at two weeks seen by MISO in operation. These weeks were chosen to represent normal operation (week 1) and extreme conditions (week 2). Each week was iterated through as described above, and each method was used in turn. The 24 hour costs and the PSH profits were collected for each day and method. Since our time frame was a week, we were most interested in the total costs and profits for the week as a whole.

The multi-day models described above are built using data from the day-ahead models. However, if building such models in real time, this fidelity of data would not be available. For a power system, the expected demands for a day become less certain the further in the future considered. To account for this, we looked at varying the demands for each day after the first. In operation, MISO has seen the demands vary up to 10% a week out. So, when building a multi-day model, the demands after the first day are allowed to drift up to 2% more each consecutive day. These drifts are randomly generated from a uniform distribution, and 10 such demand scenarios were created. These 10 models were solved and the dual prices and state of charge values

at hour 24 were averaged to be used in the corresponding day-ahead model. After these values were computed, the procedure continued as before.

3.3.2 Results

To show that the PSH constraints don't greatly increase the difficulty of the model, the computational performance of the original day-ahead models were compared against those with the new constraints. The original day-ahead model was built and run for each day in our test set 10 times with different seeds, and the times were averaged. The same was done for the PSH models with random initial and final state of charge values. Table 3.1 shows these averaged times. Though there are some days where the average performance is poorer, most days have similar run times. The poor performance is likely a result of the randomness introduced in setting initial and final state of charge values for the PSH units. It's possible that these values could increase of the difficulty of satisfying the demand for the day.

Table 3.2 compares the grid costs and PSH profits for the different methods with known demands; the percent change from the base method to the naive method and the base method to the dual method are reported. The percent difference is used due restrictions on what data can be disseminated which includes actual costs and profits. In week 1, which is normal operation, no method significantly outperforms the others in either measure. This is expected given Figure 3.1 which shows the reservoir depleting over the day and filling overnight. Knowledge of future days is not necessary. In week 2, this knowledge shows some utility. Again, the objective is not significantly improved, but the profits for the PSH operators benefits greatly. Both methods that rely on this future information provide substantial improvement here. This makes sense given the prices seen in figure 3.2. There is no longer a simple daily, cyclical structure. Instead, the prices are low early in the week discouraging discharge, but higher later which discourages charging. Knowledge of future days can then be used to better balance this behavior.

Table 3.1: Average times to solve the test weeks

Week 1		Week 2	
Original Time	PSH Time	Original Time	PSH Time
117.18	116.79	607.55	681.82
123.46	125.25	315.91	506.9
122.37	223.79	217.18	247.58
155.42	167.3	182.34	181.84
155.11	226.72	177.96	176.05
153.81	299.04	175.14	164.93
146.98	151.75	196.8	197.01

In table 3.3, the grid costs and PSH profits are compared for the different methods where the demands are stochastic. Again, the percent change from the base method to the naive method and the base method to the dual method are presented. The performance for each method is very comparable to the models where the demands are known. In week 1, the methods behave similarly with little difference in system costs and operator profits. Week 2 sees significant benefit, particularly in profits, to using forward knowledge to set an expected final state of charge for each day. This is what would be expected based on the models with demand knowledge, reinforcing the validity of the stochastic models.

To better measure the accuracy of the stochastic models, table 3.4 compares the performance of each method to those from the model with known demands. The table shows that the stochastic model provides values that are very close to the model using known demands with costs and profits differing by less than 1%. This suggests that the stochastic model can be used to effectively plan the final state of charge for each day-ahead model.

3.4 Discussion and Conclusions

We have seen in this work that incorporating more realistic constraints for PSH units into the day-ahead model does not severely impact the computational difficulty of these problems. Indeed, though some days appeared to solve slower with the constraints, the performance was generally comparable. One main reason for this discrepancy is the randomness of the initial and final state of charge values used in testing. Future studies can be done with more realistic charge values in this timing test. Beyond computational performance, even the objective values with the added constraints do not change drastically from the original approach. All this suggests that incorporating these constraints could provide a more realistic representation of power generation on a grid without serious effect on the computational performance and solution quality.

Table 3.2: Comparison of the PSH methods using known demands

	Naive	Dual	
Week 1	−0.044%	−0.041%	Grid Costs
	0.58%	1.09%	PSH Profits
Week 2	−0.25%	−0.28%	Grid Costs
	812%	867%	PSH Profits

Table 3.3: Comparison of the PSH methods using stochastic demands

	Naive	Dual	
Week 1	−0.085%	−0.0096%	Grid Costs
	1.31%	0.51%	PSH Profits
Week 2	−0.26%	−0.28%	Grid Costs
	824%	874%	PSH Profits

Table 3.4: Comparison of the PSH methods between stochastic and known demands

	Base	Naive	Dual	
Week 1	0.0%	−0.042%	0.031%	Grid Costs
	0.00046%	0.72%	−0.57%	PSH Profits
Week 2	0.0%	−0.0064%	0.00031%	Grid Costs
	−0.35%	0.94%	0.40%	PSH Profits

With the PSH constraints added to the day-ahead model, the initial and final state of charge values on the reservoirs need to be set for the day. As specified by MISO, the reservoirs on their part of the electric grid need to be full at the beginning of the week. This provides an initial and final charge for a week which leads to several approaches for setting these values each day. These are our base, naive, and dual methods. This work has shown that utilizing future knowledge, such as in the naive and dual methods, is beneficial in improving the profits of PSH owners and even on the total costs of the grid though this impact is fairly minimal. This can even be seen when the demands are stochastic which better models the uncertainty of expected loads several days in the future. When looking at comparing these methods using future data, the dual method outperforms the naive method though not as extremely as both over the base method. So, using a multi-day model to set the final state of charge for a day is beneficial to solving the day-ahead problem.

This work suggests that incorporating more accurate PSH constraints does not significantly effect computational difficulty of the day-ahead problem and can be used to improve overall system costs while improving operator profits. In models provided by MISO, there are only three such units. So, we could expect this to be of even greater benefit for ISOs with many PSH units in the parts of the grid they manage. Additionally, more accurate modeling could benefit work looking at using PSH units to back up renewable power sources.

Chapter 4

Impact of Planning Horizon Length in Unit Commitment

4.1 Introduction

Given the improvements in solving the MILP formulation of the UC problem presented in chapter 1, a natural extension is to look at the effect of using longer time horizons. The current day-ahead UC problem looks at 36 hours when planning. This horizon was a compromise between solving in a reasonable time frame and considering the effects of expected demands in finding a suitable schedule. Longer time horizons should improve this further by considering the impact of different generators being scheduled several days out. This is of particular utility for generators with long up-times. In normal operation, if a generator has an up-time greater than 36 hours and is scheduled on a given day, it may be forced to stay on through the next day. With a longer time horizon, a better schedule may be found that waits to use the generator until a more opportune time. Of course, there is a trade-off between longer time horizons and solution times. Increasing the length of time considered for a solution will increase the number of variables that need to be solved. Given that many of the

variables are integer, the branch and bound process will be slowed. However, some of this may be mitigated by relaxing integrality on later time periods.

The remainder of this chapter is organized as follows. In Section 4.2 we discuss the experiments for testing extended horizons. Section 4.3 reports the main numerical results of the paper. Finally we discuss the implications of our findings in Section 4.4.

4.2 Experimental Setup

Our goal in this work was to see if there are any cost benefits to extending the optimization horizon of the UC problem. This must be balanced against the time required to solve these bigger problems. To that end, we looked at three different horizon lengths and compared their objective values and solution times against those of the original UC formulation. The horizons chosen were 48 hours, 60 hours, and 72 hours. We also considered the effect of relaxing integer variables at the end of the horizon. We looking at relaxing the last 12 hours, the last 24, and everything after hour 36.

These extended horizon models were built by combining data from several consecutive 36-hour instances. In each of these 36-hour daily instances, the last 12 hours were dropped and the days concatenated. We omitted the virtual bids and demand responses and limited the number of transmission constraints in these models past hour 36. This was necessary to ensure the corresponding linear program would fit in memory and be solvable in a reasonable time frame.

To test the effect of these different horizons, we looked at two weeks seen by MISO in operation. These weeks represent normal operation (week 1) and extreme conditions (week 2). Full weeks were needed in order to combine consecutive days into longer horizon models. For each week, only five days were compared to ensure that all time horizons can be tested. The first day was solved with a given horizon and relaxation; the 24 hour costs and solution times were collected. Then the generator schedule was used to set the initial conditions for the following day which was in turn

solved. This process continued for each of the five days in each week. Once both weeks were run for a specific horizon and relaxation, this process was repeated for the next combination until all were tested. Finally, this process was repeated using the standard UC model to establish a base.

For testing, each UC model was solved using the larger of a 0.25% or $12,000 gap using Gurobi 8.1.1 with a time limit of 3,600 seconds. Computations were done on a Linux server hosted by MISO with 32GB of ram and 12-core Intel processor running at 2.5GHz.

4.3 Computational Results

Table 4.1 and table 4.3 show the percent difference between the costs for the various extended horizon models and the costs from the standard UC model. Table 4.1 shows the differences for a week under normal conditions. The results suggest that increasing the time horizon can be beneficial seeing around 3.6% reduction in the system costs for five consecutive days. This becomes more significant when considering a potential 3% reduction in costs weekly over the course of a year or more. Though the results on week 2 will show that such a reduction is unlikely every week. Interestingly, the 48-hour model appears to outperform the other horizon lengths. It may be that there are no generators with up or down times significantly greater than 48 hours in these instances making the longer horizons less useful, especially if they timeout when solving. Table 4.2 shows the average time to solution over the 5 days of week 1 for each horizon. Given this and the solution improvements, the 48-hour seems to produce much better solutions without significant increase in solution times.

Table 4.3 shows the differences for a week under extreme conditions. The results again suggest that increasing the time horizon can be beneficial seeing around 1.8% reduction in the costs for the five consecutive days. This is less significant than for week 1, as mentioned, but it shows that even during extreme conditions these horizons can provide improvement. Here we see that the 48-hour model still outperforms the

Table 4.1: Percent differences of the costs for different horizon lengths against the standard UC model for week 1

Horizon Length (h) \ Hours Relaxed	0	12	24	36
72	-3.46%	-3.52%	-3.50%	-3.40%
60	-3.46%	-3.41%	-3.44%	*
48	-3.58%	-3.59%	*	*

Table 4.2: The average time in seconds to solve the five days from week 1

Horizon Length (h) \ Hours Relaxed	0	12	24	36
72	596.87	738.29	929.28	1261.86
60	327.49	354.53	352.28	*
48	189.81	173.87	*	*
36	131.25	*	*	*

other horizon lengths. Table 4.4 shows the average time to solution over the five days of week 2 for each horizon. Here we see that the average times to solution are much different than for the standard UC model even with a 48-hour model. For the 48-hour model, only one day solves significantly slower which brings the whole average up. However, the other two horizons solve slowly for at least three of the five days. This is surprising but due to the extreme weather conditions of the week, it's possible that longer horizon lengths struggled to find solutions that satisfied the security constraints on the system.

4.4 Conclusions

In this work, we have seen that using longer time horizons in the UC problem can improve solution quality. Especially in a week with normal operating conditions though even under extreme conditions there can be some benefit. As expected, this comes with a cost in time to solution. This is particularly apparent for week 2 where even a slight increase in the time horizon can lead to bad average solution times; though that is mostly due to one or two days taking an abnormally long time to solve. In general, this study suggests that a modest increase in the length of the time horizon can be beneficial while not severely impacting the time to solution for the problem. Future work could test additional days or even perform a study over a year to determine the benefits using longer time horizons.

A surprising result found during testing is that relaxing variables in later hours seems to hurt performance. These relaxations appeared to inhibit the pre-processing stage while solving with Gurobi. This issue might become less relevant with smaller mip gaps or longer time horizons. Though it could also be beneficial if using some open source optimization packages McIlvenna et al. (2020).

Table 4.3: Percent differences of the costs for different horizon lengths against the standard UC model for week 2

Horizon Length (h) \ Hours Relaxed	0	12	24	36
72	-1.64%	-1.71%	-1.58%	-1.71%
60	-1.64%	-1.53%	-1.56%	*
48	-1.77%	-1.78%	*	*

Table 4.4: The average time in seconds to solve the five days from week 2

Horizon Length (h) \ Hours Relaxed	0	12	24	36
72	2081.66	2405.04	2248.62	1795.86
60	1193.27	1573.35	1371.66	*
48	1077.41	913.42	*	*
36	218.53	*	*	*

Bibliography

Anjos, M. F., Conejo, A. J., et al. (2017). Unit commitment in electric energy systems. *Foundations and Trends® in Electric Energy Systems*, 1(4):220–310. 1

Atakan, S., Lulli, G., and Sen, S. (2018). A state transition MIP formulation for the unit commitment problem. *IEEE Transactions on Power Systems*, 33(1):736–748. 2

Bacci, T., Frangioni, A., Gentile, C., and Tavlaridis-Gyparakis, K. (2019). New minlp formulations for the unit commitment problems with ramping constraints. *Optimization Online*. 4

Baum, S. and Trotter Jr, L. E. (1978). Integer rounding and polyhedral decomposition for totally unimodular systems. In *Optimization and Operations Research*, pages 15–23. Springer. 8

Bendotti, P., Fouilhoux, P., and Rottner, C. (2020). Symmetry-breaking inequalities for ilp with structured sub-symmetry. *Mathematical Programming*, 183(1):61–103. 3, 17

Bixby, R. E. (2012). A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, (2012):107–121. 2

Borghetti, A., D'Ambrosio, C., Lodi, A., and Martello, S. (2008). An milp approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems*, 23(3):1115–1124. 28

Brandenberg, R., Huber, M., and Silbernagl, M. (2017). The summed start-up costs in a unit commitment problem. *EURO Journal on Computational Optimization*, 5(1-2):203–238. 2

Brown, P. D., PeÇas Lopes, J. A., and Matos, M. A. (2008). Optimization of pumped storage capacity in an isolated power system with large renewable penetration. *IEEE Transactions on Power Systems*, 23(2):523–531. 28, 29

Bruninx, K., Dvorkin, Y., Delarue, E., Pandžić, H., D'haeseleer, W., and Kirschen, D. S. (2016). Coupling pumped hydro energy storage with unit commitment. *IEEE Transactions on Sustainable Energy*, 7(2):786–796. 28

Carrion, M. and Arroyo, J. M. (2006). A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, 21(3):1371–1378. 2

Chen, Y., Casto, A., Wang, F., Wang, Q., Wang, X., and Wan, J. (2016). Improving large scale day-ahead security constrained unit commitment performance. *IEEE Transactions on Power Systems*, 31(6):4732–4743. 1

Chen, Y., Wang, Q., Wang, X., and Guan, Y. (2014). Applying robust optimization to miso look-ahead commitment. In *2014 IEEE PES General Meeting— Conference & Exposition*, pages 1–5. IEEE. 1

Damcı-Kurt, P., Küçükyavuz, S., Rajan, D., and Atamtürk, A. (2016). A polyhedral study of production ramping. *Mathematical Programming*, 158(1-2):175–205. 2

Eldridge, B., O'Neill, R., and Hobbs, B. F. (2019). Near-optimal scheduling in day-ahead markets: pricing models and payment redistribution bounds. *IEEE transactions on power systems*, 35(3):1684–1694. 2

Frangioni, A., Gentile, C., and Lacalandra, F. (2009). Tighter approximated MILP formulations for unit commitment problems. *IEEE Transactions on Power Systems*, 24(1):105–113. 2

Garver, L. L. (1962). Power generation scheduling by integer programming-development of theory. *Power Apparatus and Systems, Part III. Transactions of the American Institute of Electrical Engineers*, 81(3):730–734. 1, 2

Gentile, C., Morales-Espana, G., and Ramos, A. (2017). A tight MIP formulation of the unit commitment problem with start-up and shut-down constraints. *EURO Journal on Computational Optimization*, 5(1–2):177–201. 2, 3

Guan, Y., Pan, K., and Zhou, K. (2018). Polynomial time algorithms and extended formulations for unit commitment problems. *IISE transactions*, 50(8):735–751. 4

Jiang, R., Wang, J., and Guan, Y. (2012). Robust unit commitment with wind power and pumped storage hydro. *IEEE Transactions on Power Systems*, 27(2):800–810. 28, 29

Johnson, R. B., Oren, S. S., and Svoboda, A. J. (1997). Equity and efficiency of unit commitment in competitive electricity markets. *Utilities Policy*, 6(1):9–19. 2

Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A. (2009). *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media. 2

Knueven, B., Ostrowski, J., and Wang, J. (2018a). The ramping polytope and cut generation for the unit commitment problem. *INFORMS Journal on Computing*, 30(4):739–749. 4

Knueven, B., Ostrowski, J., and Watson, J.-P. (2018b). Exploiting identical generators in unit commitment. *IEEE Transactions on Power Systems*, 33(4). 3, 4, 14, 17, 19, 32

Knueven, B., Ostrowski, J., and Watson, J.-P. (2018c). A novel matching formulation for startup costs in unit commitment. 2

Knueven, B., Ostrowski, J., and Watson, J.-P. (2020). On mixed-integer programming formulations for the unit commitment problem. *INFORMS Journal on Computing*, 32(4):857–876. 1, 2, 4, 17, 19

Lee, J., Leung, J., and Margot, F. (2004). Min-up/min-down polytopes. *Discrete Optimization*, 1(1):77–85. 2

Lima, R. M. and Novais, A. Q. (2016). Symmetry breaking in MILP formulations for unit commitment problems. *Computers & Chemical Engineering*, 85:162–176. 3, 17

Malkin, P. (2003). Minimum runtime and stoptime polyhedra. *CORE, Université catholique de Louvain*. Working Paper. 2

Margot, F. (2002). Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94(1):71–90. 3

McIlvenna, A., Herron, A., Hambrick, J., Ollis, B., and Ostrowski, J. (2020). Reducing the computational burden of a microgrid energy management system. *Computers & Industrial Engineering*, 143:106384. 47

Morales-España, G., Latorre, J. M., and Ramos, A. (2013). Tight and compact MILP formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, 28(4):4897–4908. 2

O'Neill, R. P. (2017). Computational issues in ISO market models. Workshop on Energy Systems and Optimization. 2

Ostrowski, J., Anjos, M. F., and Vannelli, A. (2012). Tight mixed integer linear programming formulations for the unit commitment problem. *IEEE Transactions on Power Systems*, 27(1):39. 2

Ostrowski, J., Anjos, M. F., and Vannelli, A. (2015). Modified orbital branching for structured symmetry with an application to unit commitment. *Mathematical Programming*, 150(1):99–129. 3, 17

Ostrowski, J., Linderoth, J., Rossi, F., and Smriglio, S. (2011). Orbital branching. *Mathematical Programming*, 126(1):147–178. 3

Rajan, D. and Takriti, S. (2005). Minimum up/down polytopes of the unit commitment problem with start-up costs. *IBM Research Report*, RC23628 (W0506-050). 2

Silbernagl, M. (2016). *A polyhedral analysis of start-up process models in unit commitment problems*. PhD thesis, Technische Universität München. 2

Sioshansi, R., O'Neill, R., and Oren, S. S. (2008). Economic consequences of alternative solution methods for centralized unit commitment in day-ahead electricity markets. *IEEE Transactions on Power Systems*, 23(2):344–352. 2

U.S. Department of Energy (2021). Pumped storage hydropower. https://www.energy.gov/eere/water/pumped-storage-hydropower. Accessed: 2021-09-07. 28

Van den Bergh, K., Bruninx, K., Delarue, E., and D'haeseleer, W. (2014). A mixed-integer linear formulation of the unit commitment problem. *WP EN2014-07*. 29

Wu, L. (2016). Accelerating NCUC via binary variable-based locally ideal formulation and dynamic global cuts. *IEEE Transactions on Power Systems*, 31(5):4097–4107. 2

Appendix A

The Unit Commitment Model

A.1 Notation

Sets and indices

G set of generators.

T set of consecutive time periods.

S set of states in which a generator starts. E.g., $S = \{cold,\ warm,\ hot\}$.

VT set of virtuals.

DD set of dispatchable demands

Data

a_{it} constant production cost coefficient of generator i, no load cost.

b_{it} linear production cost coefficient of generator i.

c_{it} quadratic production cost coefficient of generator i.

d_{is} cost for starting up generator i in the state s.

cr_{it} unit cost of regulating reserve at generator i in period t.

cs_{it} unit cost of spin reserve at generator i in period t.

csn_{it} unit cost of online supplemental reserve at generator i in period t.

csf_{it} unit cost of offline supplemental reserve at generator i in period t.

cx_{it} unit cost of virtual i at time t.

cy_{it} unit cost of dispatchable demand i at t.

M penalty cost of shedding a unit of load.

D_t demand in period t.

R_t regulating reserve requirement at t.

RS_t spin reserve requirement at t.

RC_t contingency reserve requirement at t.

$\underline{P}_{it}$ minimum power output of generator i at t in economic mode.

$\overline{P}_{it}$ maximum power output of generator i at t in economic mode.

$\underline{PR}_{it}$ minimum power limit of generator i at t in regulating reserve mode, $\underline{P}_{it} \leq \underline{PR}_{it}$.

$\overline{PR}_{it}$ maximum power limit of generator i at t in regulating reserve mode, $\overline{PR}_{it} \leq \overline{P}_{it}$.

$\overline{RSF}_{it}$ maximum offline spin reserve on generator i at t.

$\underline{TD}_i$ minimum downtime of generator i.

$\overline{TD}_i$ maximum downtime of generator i.

$\underline{TU}_i$ minimum uptime of generator i.

$\overline{TU}_i$ maximum uptime of generator i.

RR_{it} ramp-rate limit of generator i at t.

PU_i startup capacity of generator i.

PD_i shutdown capacity of generator i.

$\underline{T}_i^{hot}, \underline{T}_i^{warm}, \underline{T}_i^{cold}$

the minimum number of periods for the hot, warm and cold starts for generator i.

$\overline{T}_i^{hot}, \overline{T}_i^{warm}, \overline{T}_i^{cold}$

the maximum number of periods for the hot, warm and cold starts for generator i. For example, hot start for generator i is between $\underline{T}_i^{hot}$ and $\overline{T}_i^{hot}$ periods after the a shutdown.

α_{ilt} shift factors for entity i and branch l at t.

$\overline{F}_{lt}, \underline{F}_{lt}$ upper and lower limit for branch l at t.

TP_i daily maximum energy generation for generator i.

TS_i daily maximal number of starts for generator i.

$\overline{X}_{it}$ energy capacity of virtual i at t.

vm_{it} multiplier for virtual i at t, 1 for injection and -1 for withdrawal.

$\overline{Y}_{it}$ energy capacity of dispatchable demand i at t.

$\ell_{i,b}$ The size of the bth energy offer by generator i.

$h_{i,b}$ The price of energy in the bth offer by generator i.

Derived data

h_{it}^k the slop of the kth segment of piecewise approximation of the production cost of generator i at time t.

p_{it}^k the x intercept of the kth segment of piecewise approximation of the production cost of generator i at time t.

Control variables

u_{it} binary variable, 1 if unit $i \in V$ is on at $t \in T$; 0 otherwise.

v_{it} binary variable, 1 if unit $i \in V$ starts up at t; 0 otherwise.

w_{it} binary variable, 1 if unit i shuts down at t; 0 otherwise.

$s_{i,t',t'}$ binary variable, 1 if unit i shut down in time t' and the next turn on was at time t.

x_{it} energy produced by virtual i at period t.

y_{it} energy dispatched by demand i at period t.

λ_{lt} amount of violation in constraint l at time t.

q_{it} amount of energy produced by generator i at time t.

r_{it} amount of energy for regulating reserve by generator i at t.

ur_{it} binary variable, 1 if unit i is committed to regulating reserve at t; 0 if o.w.

rr_{it} amount of regulating reserve in generator i at t.

rs_{it} amount of spin reserve in generator i at t.

rsn_{it} amount of online supplemental reserve in generator i at t.

rsf_{it} amount of offline supplemental reserve in generator i at t.

k_{itb} amount of power generator i produces in price bin b at time t.

Auxiliary variables

f_{it} Keeps track of generator costs at time period t.

A.2 Model

$$\min_{\substack{f,q,u,v,w,s,\lambda;\\ ,ur,rr,rs,rsn,rsf}} \sum_{i\in G}\sum_{t\in T} f_{it} + \sum_{t\in T}\sum_{i\in VT} vm_{it}cx_{it}x_{it} - \sum_{t\in T}\sum_{i\in DD} cy_{it}y_{it} \tag{A.1}$$

$$+ \sum_{i\in G}\sum_{t\in T}(cr_{it}rr_{it} + cs_{it}rs_{it} + csn_{it}rsn_{it} + csf_{it}rsf_{it}) + \sum_{t\in T} M\lambda_{lt} \tag{A.2}$$

$$\text{s.t. } f_{it} \geq C_t^S + a_i u_{it} + \sum_{b\in B_g} k_{gtb} \quad \forall i \in G, \forall t \in T \tag{A.3}$$

$$k_{gtb} \leq \ell_{g,b} u_{g,t} \ \forall g, b, t \tag{A.4}$$

$$\sum_{b\in B_g} k_{gtb} = p_{gt} \ \forall g, t \tag{A.5}$$

$$p_{i,t} \geq \underline{P}_{it} u_{it} + (\underline{PR_{it}} - \overline{P}_{it}) ur_{i,t} \quad \forall g, t \tag{A.6}$$

$$p_{i,t} + rr_{i,t} + rs_{i,t} + rsn_{i,t} \leq \overline{P}_{it} u_{it} + (\overline{PR_{it}} - \overline{P}_{it}) ur_{i,t} \quad \forall g, t \tag{A.7}$$

$$\sum_{i\in G}(q_{it}) + \sum_{i\in VT} vm_{it}x_{it} - \sum_{i\in DD} y_{it} = D_t \quad \forall t \in T \tag{A.8}$$

$$\sum_{i\in G} rr_{it} \geq R_t \quad \forall t \in T \tag{A.9}$$

$$\sum_{i\in G}(rr_{it} + rs_{it}) \geq R_t + RS_t \quad \forall t \in T \tag{A.10}$$

$$\sum_{i\in G}(rr_{it} + rs_{it} + rsn_{it} + rsf_{it}) \geq R_t + RC_t \quad \forall t \in T \tag{A.11}$$

$$\sum_{t'=t-\underline{TU_i}}^{t} v_{it'} \leq u_{it} \quad t = [\underline{TU}_i, |T|], \forall i \in G \tag{A.12}$$

$$\sum_{t'=t-\underline{TD_i}}^{t} w_{it'} \leq 1 - u_{it} \quad t = [\underline{TD}_i, |T|], \forall i \in G \tag{A.13}$$

$$v_{it} \leq \sum_{t'=t}^{t+\overline{TU}_i} w_{it'} \quad t = [0, |T| - \overline{TU}_i], \forall i \in G \tag{A.14}$$

$$u_{it} - u_{it-1} = v_{it} - w_{it} \quad \forall i \in G, t \in T \tag{A.15}$$

$$q_{it} + rr_{it} \leq \overline{P}_{it} u_{it} + (\overline{PR}_{it} - \overline{P}_{it}) ur_{it} \quad \forall i \in G_1, t \in T \tag{A.16}$$

$$q_{it} - rr_{it} \geq \underline{P}_{it} u_{it} + (\underline{PR}_{it} - \underline{P}_{it}) ur_{it} \quad \forall i \in G_1, t \in T \tag{A.17}$$

$$q_{it} - q_{it-1} \leq (RR_{it} + \underline{P}_{it-1})u_{it} - \underline{P}_{it-1}u_{it-1} + (\underline{P}_{it} - \underline{P}_{it-1} - \frac{1}{2}RR_{it})v_{it}$$

$$\forall i \in G, t \in T \setminus \{0\} \tag{A.18}$$

$$-q_{it} + q_{it-1} \leq -\underline{P}_{it}u_{it} + (RR_{it} + \underline{P}_t)u_{it-1} + (\bar{P}_{it-1} - RR_{it} - \underline{P}_{it})w_{it}$$

$$\forall i \in G, t \in T \setminus \{0\} \tag{A.19}$$

$$\sum_{t \in \{0,\cdots,23\}} q_{it} \leq TP_i, \sum_{t \in \{24,\cdots,35\}} q_{it} \leq TP_i \quad \forall i \in G \tag{A.20}$$

$$\sum_{t \in \{0,\cdots,23\}} v_{it} \leq TS_i, \sum_{t \in \{24,\cdots,35\}} v_{it} \leq TS_i \quad \forall i \in G \tag{A.21}$$

$$0 \leq x_{it} \leq \overline{X}_{it} \quad \forall i \in VT, \quad \forall t \in T \tag{A.22}$$

$$0 \leq y_{it} \leq \overline{Y}_{it} \quad \forall i \in DD, \quad \forall t \in T \tag{A.23}$$

$$ur_{it} \leq u_{it} \quad \forall i \in G \; \forall t \in T \tag{A.24}$$

$$0 \leq rr_{it} \leq \min\{\frac{1}{5}R_t, \frac{1}{12}RR_{it}\}ur_{it} \quad \forall i \in G, \; \forall t \in T \tag{A.25}$$

$$0 \leq rs_{it} \leq \min\{\frac{1}{5}RC_t, \frac{1}{6}RR_{it}u_{i,t}\} \quad \forall i \in G, \; \forall t \in T \tag{A.26}$$

$$0 \leq rsn_{it} \leq \min\{\frac{1}{5}RC_t, \frac{1}{6} * RR_{it}u_{i,t}\} \quad \forall i \in G, \; \forall t \in T \tag{A.27}$$

$$0 \leq rsf_{it} \leq \min\{\frac{1}{5}RC_t, \overline{RSF}_{it}(1 - u_{it})\} \quad \forall i \in G, \; \forall t \in T \tag{A.28}$$

$$\underline{F}_{lt} - \lambda_{lt} \leq \sum_{i \in G}(q_{it})\alpha_{ilt} + \sum_{i \in VT} vm_{it}x_{it}\alpha_{ilt} - \sum_{i \in DD} y_{it}\alpha_{ilt} \leq \overline{F}_{lt} + \lambda_{lt}$$

$$\forall l \in L \forall t \in T \tag{A.29}$$

$$\lambda_{lt} \geq 0 \quad \forall l \in L, t \in T \tag{A.30}$$

$$\sum_{t'=t-\underline{T}_i^{cold}+1}^{t-\underline{TD}_i} s_{i,t',t} \leq v_{it} \quad \forall t \in T \tag{A.31}$$

$$\sum_{t'=t+\underline{TD}_i}^{t+\underline{T}_i^{cold}-1} s_{i,t,t'} \leq w_{it}, s \in S \tag{A.32}$$

$$C_t^S = d_{i,cold}v_{it} + \sum_{k \in S \setminus \{cold\}} (d_{ik} - d_{i,cold})(\sum_{t'=t-\overline{T}_i^{k-1}+1}^{t-\underline{T}_i^k} s_{i,t',t}), \quad \forall t \in T. \tag{A.33}$$

Note that this is a fairly typical UC model with the small exception of the matching variables s that keep track of time-dependent startup costs and the k variables that keep track of how much power is produced at what price (a contrast to the piecewise linearization of the quadratic cost curve).

Vita

Jonathan Schrock was born in Chattanooga, TN to Randy and Karen Schrock. He has a younger and older sister: Sarah and Rachel respectively. He attended Wheatfield Elementary School in Wheatfield, IN followed by Kouts Middle and High School in Kouts, IN. After graduating, he began studying Mathematics and Computer Science at Taylor University in Upland, IN. Jonathan worked on several research projects with faculty at Taylor as well as a REU at the University of Illinois Urbana/Champaign. The projects covered topics such as knot theory, network mapping, and large scale power systems. Additionally, he worked as a tutor and teaching assistant during his time at Taylor. Jonathan obtained a Bachelor of Science degree from Taylor University in May 2011 in Mathematics and Computer Science. After which he worked for the Department of Defense before beginning work at Oak Ridge National Laboratory and graduate school at the University of Tennessee, Knoxville. He studied mathematics at UT, and worked on projects in quantum computing, graph generation, and HPC benchmarking at ORNL. Jonathan received his Master of Science degree from UT in May 2015. He continued working at ORNL before returning to UT seeking a PhD in Industrial Engineering. In pursuit of that degree, Jonathan is primarily working on the unit commitment problem using symmetry methods to improve solution time and quality. He also is employed as an intern with MISO.