Toward Scalable Stochastic Unit Commitment - Part 2: Assessing Solver Performance

Kwok Cheung, *Senior Member, IEEE,* Dinakar Gade, César Silva Monroy, *Member, IEEE,* Sarah M. Ryan, *Member, IEEE,* Jean-Paul Watson, *Member, IEEE,* Roger J.-B. Wets, and David L. Woodruff, *Member, IEEE*

***Abstract*—In this second portion of a two-part analysis of a computational approach to scalable stochastic unit commitment, we transition our focus from approximating accurate stochastic process models of load to solving the resulting stochastic optimization models in tractable run-times. Our solution technique is based on Rockafellar and Wets' progressive hedging algorithm, a scenario-based decomposition strategy for solving stochastic programs. To achieve high-quality solutions in tractable run-times, we describe key customizations of the progressive hedging algorithm for stochastic unit commitment. Using a variant of the WECC-240 test case, we demonstrate the ability of our approach to solve moderate-scale stochastic unit commitment problems with reasonable numbers of scenarios in less than 30 minutes of wall clock time on a commodity hardware. Further, we demonstrate that the resulting solutions are high-quality, with cost typically within 1-2% of optimal. Our optimization model and associated test cases are publicly available, serving as a basis for evaluating the relative effectiveness of stochastic unit commitment solvers.**

***Index Terms*—Unit Commitment, Load Uncertainty, Optimization, Stochastic Mixed-Integer Programming.**

I. Introduction

WHile there is a significant body of research on stochastic unit commitment (SUC) in the power systems literature (see [1], [2], [3], [4] for a representative sample), these efforts have not yet been successfully transferred to real-world industrial contexts. This is due in large part to the well-known computational difficulty of stochastic unit commitment, where even small cases with a handful of scenarios can take hours to solve [5]. A survey of prior algorithmic approaches to SUC is provided in [6]. An analysis of this survey indicates that the current state-of-the-art for SUC can tackle approximately 50 scenarios on instances with approximately a hundred thermal generation units, achieving solutions in several hours of run time. Further, those studies consider short (24-hour) time horizons, which greatly simplifies the SUC problem relative to the more typical 48 hour horizons executed in practice.

The primary purpose of this paper is to detail a research effort dedicated to developing a SUC solver ultimately capable of achieving solutions in tractable (e.g., less than 30 minute) run-times, given realistic numbers of time periods (e.g., 48 hour-long periods) on full-scale power systems (e.g., 1000 generation units). In this context, a "solution" refers to an implementable and non-anticipative generation schedule, with associated expected cost *and* optimality bound. We demonstrate significant progress toward this goal, considering a variant of the well-known WECC-240 test case [7]. We leverage modest-scale parallelism to achieve the required run-times, leveraging commodity computing capabilities that an ISO / utility either presently possesses or is likely to acquire in the near future. Fundamentally, our goal is to demonstrate the viability of SUC in a real-world applications context.

Our computational experiments proceed in the context of load scenarios generated via the algorithmic process described in the companion paper [8]. Accurate assessment of stochastic unit commitment solvers requires accurate load scenarios, as the latter can potentially impact algorithm performance. We limit our investigations of solver performance to given, pre-specified sets of load scenarios. Issues relating to scenario reduction and sampling, out-of-sample solution validation and, quantification of cost savings are beyond the scope of the present contribution. Rather, we focus on the goal of demonstrating that operational run-times can be achieved for SUC instances with reasonable numbers of realistic load scenarios – a key step toward ultimate commercial adoption of both the underlying uncertainty model and solver.

A secondary goal of this paper is to establish a publicly available SUC model and sets of corresponding instances. The availability of such instances – particularly with high-accuracy scenario sets – is critical to driving SUC solver research, in order to quickly identify and focus on the most promising algorithmic alternatives. To date, SUC research has not been performed in such a context, i.e., direct and accurate comparisons of solver run-times is absent in the literature.

The remainder of this paper is organized as follows. We describe the core deterministic and stochastic optimization models used in our analysis in Section II. Our solution approach is detailed in Section III. Our case study data, based on the WECC-240 test case, is described in Section IV. We describe our computational experiments and associated results in Section V. We then conclude with a summary of our results in Section VI.

II. Unit Commitment Models

We now introduce the core optimization models used in our analysis. We begin by introducing the deterministic UC optimization model in Section II-A. We then discuss the extension of this core model to a two-stage stochastic programming context in Section II-B.

Kwok Chueng is with Alstom Grid, in Redmond, Washington. Dinakar Gade is with Sabre Holdings in Southlake, Texas. Sarah Ryan is with Iowa State University, in Ames, Iowa. César Silva-Monroy and Jean-Paul Watson are with Sandia National Laboratories in Albuquerque, New Mexico. Roger Wets and David Woodruff are with the University of California Davis, in Davis, California.

Manuscript received November TBD, 2013; revised XXX.

A. Core Deterministic UC Model

As a basis for our stochastic unit commitment model, we adopt the deterministic UC model introduced by Carrion and Arroyo (CA) [9]. The purported benefit of this model relates to the limited number of binary variables used in the formulation, relative to the previously standard ”three-binary” deterministic UC model. While we leverage the CA formulation in this paper, we note that on large-scale instances we have not observed significant differences in the solve times between the CA formulation, the traditional three-binary formulation, and a more recent three-binary UC formulation [10].

We focus on reliability unit commitment in this paper, i.e., the process executed by an ISO following the close of the day-ahead market (DAM). Integration of uncertainty into the DAM requires modification of core market structure, which is beyond the present scope. We have implemented our version of the CA deterministic UC model in the open-source Pyomo algebraic modeling language [11]. Further, we have validated our implementation against Alstom's **e-terra***market* UC model, to ensure model correctness.

B. Two-Stage Stochastic UC Model

Following [1] and others, we extend the basic deterministic UC model into a two-stage stochastic UC model. To construct the stochastic UC model, the variables in the core deterministic UC model are partitioned into two stages, mirroring the structure of the corresponding real-world decision process. The first-stage variables consist of the thermal generator on/off state indicator variables, one for each each hour in the planning horizon. All remaining variables, including thermal generator dispatch levels, reserve allocations, and cost computations, are classified as second-stage variables. First-stage variables are required to be *non-anticipative* in a two-stage stochastic program, such that their value does not depend on the scenario that is ultimately realized. Given a set of S scenarios, in our case obtained using the procedures described in the companion paper [8], a two-stage UC model can be constructed by creating an instance of the deterministic UC model for each scenario $s \in S$. To enforce non-anticipativity for the first-stage variables, we then impose equality constraints among the instances of the corresponding variables in all scenarios. The resulting model is known as an explicit *extensive form* of the corresponding two-stage stochastic program, in which the first-stage decision variables (for reasons that are discussed in Section III) are replicated for each scenario instance. We take minimization of the sum of first stage cost (e.g., startup, no-load, and shutdown costs) plus expected second stage cost (e.g., production cost) as the optimization objective. However, we note that our solver extends trivially to risk-oriented optimization metrics such as Conditional Value-at-Risk (CVaR). For further details regarding the structure and properties of two-stage stochastic programs, we refer to [12].

Building on the Pyomo implementation of our core deterministic UC model, we have implemented our two-stage stochastic UC model in the open-source PySP package for stochastic programming [13]. Both Pyomo and PySP are distributed as part of the Coopr optimization software package (https://software.sandia.gov/trac/coopr).

As discussed in the companion paper [8], we restrict our focus to uncertainty in the day-ahead load. Our goal is to demonstrate scalability of a SUC solver on a realistic test case, using carefully constructed and accurate scenarios. While the broader project is developing analogous models for wind power, simultaneous consideration of stochastic wind power and load is beyond the scope of the present contribution.

We have chosen to consider two-stage stochastic UC models in this paper, in contrast to their more complex multi-stage variants, for three primary reasons. First, our scenario generation processes are not presently multi-stage, and instead mirror the structure of two-stage UC models. Second, two-stage models provide a lower bound on the cost savings of more general multi-stage models, and for that reason are typically analyzed first. Third, two-stage solutions are more straightforward to interpret than multi-stage solutions, simplifying presentation of results. Despite these limitations, we observe that the solution algorithms we describe below are extensible to the multi-stage case. However, the difficulty in constructing multi-stage scenario trees is a significant barrier to investigation, let alone adoption.

III. Solution Approach

Following a brief survey of prior efforts involving the solution of stochastic UC models in Section III-A, we describe our basic solution algorithm in Section III-B. We then discuss specializations of the core algorithm to two-stage stochastic UC in Section III-C. Issues related to the deployment and parallelization of our algorithm are detailed in Section III-D. Computation of lower bounds on solution quality is briefly discussed in Section III-E.

A. Background

As we report below in Section V-C, the extensive form of the two-stage SUC is practically insoluble via direct methods such as commercially available MIP solvers. Similar findings are reported throughout the SUC literature [6]. To achieve tractable run-times for two-stage SUC, decomposition techniques must be leveraged. Two dominant classes of decomposition techniques for general two-stage stochastic programs are time stage-based and scenario-based. The exemplar stage-based technique is the L-shaped method (Benders decomposition) [14]. Exemplars of scenario-based decomposition include progressive hedging (PH) [15] and dual decomposition [16].

One advantage of scenario-based decomposition techniques over their stage-based counterparts is a more uniform distribution of sub-problem difficulty. In particular, the computational difficulty of the master problem in the L-shaped method can grow significantly as the number of iterations increases, while the sub-problems are typically comparatively easy. Another advantage is that they are easily implemented in situations where software for solving the deterministic version of the problem already exists, and may have been highly customized for efficient solution – as is the case for unit commitment.

As discussed in [6, p.17], most prior analyses of stochastic unit commitment consider direct solution of the extensive form. Even those studies that use decomposition schemes are limited in the sizes of the test cases considered – typically no more than 100 generators, with 24 time periods, and fewer than 50 scenarios. Further, the reported run-times for the largest of these cases exceeds 30 minutes, and more typically an hour. Scalability to larger numbers of scenarios and time periods is thus a major and open concern.

B. Progressive Hedging

To describe the PH algorithm, we consider an abstract class of optimization problems that includes the unit commitment problem. Uncertainty in future system inputs is captured by a finite set of scenarios $\mathcal{S}$. Each $s \in \mathcal{S}$ provides a full realization of problem data, and has an associated probability of realization p_s. If the future were known with certainty, the resulting optimization problem could be written as: min $f(x) + g_s(x,y) \mid (x,y) \in Q_s$ where x and y represent vectors of first stage (e.g., unit commitments) and second stage (e.g., generator output levels) decisions, respectively. The functions f and g_s respectively compute the first (e.g., startup) and second stage (e.g., production) costs. The notation $(x,y) \in Q_s$ abstractly captures the requirement that any combination of feasible first and second stage decision vectors must satisfy all constraints imposed by the laws of physics and system operating policies under scenario s. In the case of SUC, each scenario corresponds to the deterministic, day-ahead unit commitment problem that is presently solved in operations.

Of course, the future cannot be known with certainty, and we must determine a non-anticipative x and corresponding scenario-specific y_s such that (1) the sum of first stage costs plus the expected second stage costs is minimized and (2) $(x,y_s) \in Q_s$ for all $s \in \mathcal{S}$. PH achieves this objective by decomposing the extensive form of a stochastic program by scenarios, initially relaxing the non-anticipativity constraints on first-stage decision variables. Non-anticipativity is restored via an iterative multiplier update scheme, as follows:

```
1: Initialization: $\nu \leftarrow 0$ and $w_s^\nu \leftarrow 0$ , $\forall s \in \mathcal{S}$
2: Iteration 0: $x_s^\nu = argmin_{x,y} f(x) + g_s(x,y) | (x,y) \in Q_s$, $\forall s \in \mathcal{S}$
3: Aggregation: $\overline{x}^\nu = \sum_{s \in \mathcal{S}} p_s x_s^{\nu-1}$
4: Iteration Update: $\nu \leftarrow \nu + 1$
5: Multiplier Update: $w_s^\nu \leftarrow w_s^{\nu-1} + \rho(x_s^{\nu-1} - \overline{x}^{\nu-1})$, $\forall s \in \mathcal{S}$
6: Iteration $\nu$: $x_s^\nu = argmin_{x,y} f(x) + g_s(x,y) + w_s^\nu x_s^\nu + \frac{\rho}{2}||x_s^\nu - \overline{x}^{\nu-1}||^2 \mid (x,y) \in Q_s$, $\forall s \in \mathcal{S}$
7: Convergence Check: if all solutions $x_s^\nu$ are identical, halt. Otherwise, go to Step 3.
```

In the PH pseudocode, we superscript the multipliers w, the first-stage scenario solutions x, and the first-stage averages $\overline{x}$ by the iteration counter ν; the w and x are additionally subscripted by the scenario $s \in \mathcal{S}$. Following initialization, PH solves the scenario sub-problems, in order to form an initial "best guess" at a solution that is non-anticipative. PH then updates the estimates of the multipliers w_s^ν required to enforce non-anticipativity, using a penalty parameter ρ. We observe that while ρ is a scalar in the pseudocode shown above, in general it can be variable-specific. Following the multiplier update, PH solves variants of the scenario sub-problems that are augmented with a linear term in x proportional to the multiplier w_s^ν and a quadratic proximal term penalizing deviation of x_s^ν from $\overline{x}^{\nu-1}$. These additional terms, in conjunction with the multiplier updates, are designed to gradually reduce the differences in x_s as PH progresses, eventually identifying a non-anticipative solution x. PH can be accelerated by executing the sub-problem solves in Steps 2 and 6 in parallel, with a barrier synchronization immediately following each step. While the pseudocode provided above is specific to two-stage stochastic programs, the algorithm generalizes to multi-stage contexts.

While PH is provably convergent in the case of non-linear (and therefore linear) stochastic programs, this is not the case for mixed-integer stochastic programs. In particular, the presence of integer decision variables can induce cycling behavior. However, effective techniques for detecting and breaking cycles have been recently introduced [17]. Further, accelerators are typically necessary to improve PH convergence, in order to achieve practical run-times: variable fixing (freezing the values of variables that have converged for the past k PH iterations) and slamming (forcing early convergence of specific variables that have minimal impact on the objective). Both of these techniques are described fully in [17].

C. Specialization to Stochastic UC

Progressive hedging was first applied to the SUC by Takriti et al. [1], who examined a variant designed to address the possibility of non-convergence in the mixed-integer case. [18] also examine SUC, and compare it with alternative heuristics. While both report promising results, the test case sizes considered are small, and in the case of [18] no run-times are reported; no configuration or tuning experiments are reported.

PH performance is known to be critically dependent upon the value of the ρ parameter. Poor choices can lead to non-convergence, or extremely slow convergence times. In our PH configuration, we use variable-specific ρ values for generation unit on/off variables. For a given thermal generator g, we compute the production cost $\overline{p}_g$ associated with the average power output level. We then introduce a global scaling factor α, and compute generator-specific penalties $\rho_g = \alpha \overline{p}_g$. The penalty factor is clearly independent of the time period. This strategy for setting ρ is known as "cost-proportional" ρ, shown to be an effective technique for PH parameterization [17], [13].

Other techniques we use to improve the PH performance for stochastic UC include the use of approximate solutions in early PH iterations, and the fixing of variables that have converged to non-anticipative values. The cost of obtaining optimal solutions to scenario sub-problems is prohibitive in early iterations, and is further not needed – precise estimates of the penalty terms w_s^ν are not necessary. Consequently, for PH iterations 0 and 1, we set the optimality tolerance (i.e., the "mipgap") for scenario sub-problem solves to a value γ_{01}. For PH iterations ≥ 2, we then linearly scale γ_{01} as a function of the current value of PH convergence metric δ (described in

[17]; as PH converges, the mipgap approaches 0. In practice, we threshold γ to the default migap tolerance of the solver employed, e.g., 1e-5 in the case of CPLEX. With variable fixing, we fix variables that have converged to a consistent value for the past μ PH iterations. The intuition here is that if a particular variable has converged for a fixed number of PH iterations, it is likely to remain fixed in subsequent iterations. The technique, while heuristic, has the benefit of reducing the size of the scenario sub-problems, in turn reducing solve times. We use the implementation of both techniques available in the "Watson-Woodruff" extensions available in the PySP software.

D. Parallelization and Deployment

Parallelization of PH is conceptually straightforward – the sub-problem solves at Steps 2 and 6 are independent, and can execute on distinct processing elements. In a parallel PH environment, a client process is responsible for initiating the request for sub-problem solves, computing the solution averages $\overline{x}^{\nu}$, and updating the multipliers w_s^{ν}. Relative to sub-problem solves, these actions consume a fraction of the overall run time. Parallel efficiency is limited by the difference between the average and maximum sub-problem solve time, which in practice can be significant. Because our performance metric is wall clock time, we ignore issues relating to parallel efficiency in our experiments. We use the parallel PH execution capabilities in the PySP library [13], which are in turn built on the Python Remote Objects library (http://pypi.python.org/pypi/Pyro. Pyro provides for parallel execution on distributed memory clusters and multi-core workstations.

E. Computation of Lower Bounds

Mirroring the case for deterministic unit commitment, a goal of stochastic unit commitment solvers is to provide both an implementable solution and some quantification of its optimality. Recently, [19] showed that a valid lower bound in the stochastic mixed-integer case can be obtained in any iteration μ of PH, simply by solving the optimization problems of the form min $f(x)+g_s(x,y)+w^{\nu}x_s$ for each scenario $s \in \mathcal{S}$ and forming the probability-weighted average of the resulting costs. We report these bounds in our computational experiments, considering only the bound associated with the final PH iteration. Mirroring the case of the basic PH algorithm, the lower bound computation is straightforward to parallelize.

IV. WECC-240 Case Study

As a basis for a test case, we choose the WECC-240 instance introduced in [7], which provides a simplified description of the western US interconnection. This instance consists of 85 thermal generators. Because it was originally introduced to assess market design alternatives, we have modified this instance to capture characteristics more relevant to reliability assessment, which were absent or incomplete in the original case. These include startup, shutdown, and nominal ramping limits, startup cost curves, and minimum up and down times. A full description of the modifications, and the case itself, can be obtained by contacting the authors. Our choice of WECC-240 was driven by the desire to develop a publicly releasable test case; at present, our ISO-NE test case (corresponding to the load scenario generation process described in the companion [8]) contains proprietary data.

We consider three SUC test instances in our experiments, constructed by scaling ISO-NE load to match WECC-240 system characteristics. Additional cases, one for each day in 2011, are available from the authors. Scenarios in the base case, denoted *WECC-240-r1*, are generated by randomly perturbing load from the original WECC-240 case (for a chosen day) by +- 10%. The base case is used for tuning PH parameters. Out-of-sample testing is then performed on two cases using scenarios constructed via the process described in the companion paper. These cases are denoted *WECC-240-r2* and *WECC-240-r3*, and respectively represent low-variance and high-variance scenario sets for ISO-NE (corresponding to Figures 6 and 7 in [8]). For each case, we consider instances with 3, 5, 10, 25, 50, and 100 scenarios.

V. Empirical Results

We now analyze the performance of our PH algorithm for two-stage SUC. We initially consider the *WECC-240-r1* case, performing parameter tuning and analysis. We then fix PH configuration and examine performance on the out-of-sample and more realistic *WECC-240-r2* and *WECC-240-r3* cases.

A. Computational Platforms

Our experiments are executed on two distinct platforms. The first represents a commodity high-end workstation, and consists of eight 8-core AMD Opteron 6278 2.4GHz processors with 512GB of RAM. Such a workstation is representative of the type of resource that is likely to be currently available or available in the near term to typical utilities and ISOs, and can be purchased for less than \$20K USD. The platform allows for modest-scale parallelism. The second platform, used exclusively for the larger 100-scenario instances, is Sandia National Laboratories Red Sky cluster, whose individual blades consist of two quad-core 2.3GHz Intel X5570 processors and 12GB of RAM. We observe that the dependency of our results on this particular architecture is negligible, in that a small-scale cluster with similar processors could achieve identical performance.

All parallel PH jobs are allocated a number of processes equal to the number of scenarios, plus additional processes for executing the PH master algorithm and coordinating communication among the sub-problem solution processes. We use CPLEX 12.5 as our extensive form and scenario sub-problem solver, with default parameter settings unless otherwise noted.

B. Individual Scenario Sub-Problem Difficulty

The overall run-time of PH is strongly a function of the difficulty of individual scenarios, both with and without the augmented objective terms. Thus, we begin our empirical analysis of PH performance considering CPLEX run-times on scenario sub-problems. Specifically, we consider our 100-scenario *WECC-240-r1* instance, executing PH in serial for one iteration using a ρ scaling factor equal to 1 and no variable

TABLE I
SOLVE TIME AND SOLUTION QUALITY STATISTICS FOR *WECC-240-r1* SCENARIO SUB-PROBLEMS, 100-SCENARIO INSTANCE.

MIP Gap	Solve Time (Avg. / Max.)	
	PH Iteration 0	PH Iteration 1
0.03	34.44 / 53.81	5.12 / 7.70
0.025	61.99 / 123.53	6.11 / 9.87
0.02	205.75 / 604.86	9.42 / 25.74

fixing. Warm-starting between iterations 0 and 1 is enabled, to reflect the actual PH configuration used in subsequent experiments. We vary the mipgap termination threshold γ from 0.02 to 0.03 in increments of 0.005. As the results below indicate, solution times with smaller γ are prohibitive in the context of PH. For each invocation of CPLEX, we allocate 2 threads; larger thread counts are not realistic in operational environments, where the number of scenarios is likely to exceed the number of available compute cores.

In preliminary experimentation, we observed that CPLEX 12.5 performance is significantly improved on a range of deterministic UC instances when the following two options are employed, as we do in all subsequent experiments. First, we enable the relaxation induced neighborhood search (RINS) heuristic [20], to be applied every 100 nodes in the branch-and-cut tree. Second, we set the search emphasis to "moving best bound" (option 3). Lacking either of these options, the run-times reported below are significantly inflated.

Statistics for the scenario sub-problem solve times (in seconds) are reported in Table I. We report average and maximum statistics, particularly as the latter is a key driver in parallel PH performance. The results immediately highlight the absolute difficulty of the deterministic single-scenario instances associated with the *WECC-240-r1* case, which is consistent with results reported for similarly sized instances [9], [10]. Given target run-times on the order of 30 minutes for SUC solvers to be viable in deployment contexts, it is clear that $\gamma \leq 0.02$ can not be considered; individual scenario solve times necessarily bound the time of individual PH iterations. In auxiliary experimentation, we observe the allocation of additional threads does not drop the solve times appreciably, such that the additional cores can be allocated to disparate scenario sub-problem solves.

Based on the results presented above, we limit scenario solves times in any PH iteration to 2 minutes in all experiments described below, and focus on cases when γ equals either 0.025 or 0.03. Values of γ significantly larger than 0.03 – even in early PH iterations – yield very poor PH behavior, for the following reason. In the *WECC-240-r1* instance, feasible solutions are quickly found – leveraging the characteristic that the majority of generators can be on for all time periods, while maintaining a low power output level. Taken across all scenarios, this can result in premature convergence of PH, to this trivial and highly sub-optimal solution.

Finally, we note that the use of the CPLEX RINS and moving-best-bound options empirically result in rapid reductions in the upper bound, relative to the default settings that more rapidly increase the lower bound. This difference is key, in that it moves us away from the trivial solutions described above. There are two non-exclusive strategies for achieving a target γ, and we have focused on those that reduce the optimality gap through identification of high-quality incumbent solutions.

C. Solving the Extensive Form

Next, we analyze the computational difficulty of the extensive form of the *WECC-240-r1* instance, as a function of the number of scenarios considered. The results serve as a performance baseline for the PH algorithm, and additionally provide an indication of absolute instance difficulty.

We execute all experiments associated with extensive form solves on our 64-core workstation, allocating the maximum possible number of (64) threads to each CPLEX run. Mirroring the case for individual scenario solves, enabling the RINS heuristic and moving best-bound emphasis yields significant improvements in solution quality relative to the default parameter settings. Consequently, we employ identical settings to those described in Section V-B. For each scenario count, we perform runs with a limit of 2 and 4 hours of wall clock time. For each run, we record the optimality gap reported at termination, the incumbent solution cost, the final lower bound, and the observed wall clock time. The latter can differ from the allocated time limit due to the granularity with which CPLEX checks the overall run time against the allocated limit. The results are reported in Table II.

We immediately observe the absolute difficulty of the *WECC-240-r1* extensive forms. In no case was an optimality gap less than 0.75% observed, and for the larger instances - despite the overall run-time - the gaps are significant. For the 50 and 100 scenario instances, processing had not progressed beyond the root node of the branch-and-cut tree, and the root relaxation (LP solve) time consumed a significant proportion of the run-time (e.g., 6084 seconds in the 100-scenario case). We highlight such behavior to illustrate that parallelism opportunities for a direct solve of the stochastic UC extensive form are limited, given current mixed-integer solver technology. As reported for individual scenario instances, identification of a feasible solution is relatively straightforward in the case of *WECC-240-r1*, specifically a trivial solution in which the majority of generators are on at low output levels for all time periods. Improvement of this trivial solution often does not occur until beyond an hour of wall clock time, particularly for instances with 25 or greater scenarios. Clearly, the difficulty of the root linear programming relaxation solve alone precludes direct solution of the stochastic UC in an operational context. However, the results do provide a performance baseline.

D. Parameter Tuning for PH

As discussed in Section III, there are three key parameters underlying our PH algorithm for stochastic UC: the scale factor ρ used to compute the ρ values, the choice of initial sub-problem mipgap γ, and the discrete variable fix lag μ. We now analyze the performance of our PH algorithm for various choices of these parameters, to illustrate their influence on performance in terms of both final solution quality and

TABLE II
SOLVE TIME AND SOLUTION QUALITY STATISTICS FOR THE EXTENSIVE FORM OF THE *WECC-240-r1* INSTANCE

	Two Hour Time Limit				Four Hour Time Limit			
# Scenarios	Solution Cost	Lower Bound	Gap %	Run Time (s)	Solution Cost	Lower Bound	Gap %	Run Time (s)
3	64279.18	63708.67	0.89	7291	64278.203	63797.72	0.75	14491
5	62857.52	62052.75	1.26	7309	62740.667	62180.86	0.89	14723
10	61873.01	60769.78	1.77	7444	61563.097	60835.45	1.18	14630
25	61496.24	59900.40	2.59	7739	61455.551	59963.78	2.36	14960
50	61911.74	59432.08	4.01	8279	61911.740	59540.87	3.83	15480
100	62388.85	3500.70	94.39	9379	62388.851	59548.23	4.51	16562

overall run-time. For brevity, we do not perform a fully crossed experiment, but rather explore a subset of parameter settings based on our experience with tuning PH in other domains and the sub-problem solve time statistics reported in Section V-B.

Our initial experiments consist of the following PH configuration: $\alpha = 1.0$, $\mu = 3$ for PH iterations ≥ 1, immediate fixing of variables at PH iteration 0 with converged value equal to 0, and an initial mipgap $\gamma = 0.03$. We limit the total number of PH iterations to 30, and record the terminating value of the convergence metric, the final expected solution cost, the total number of variables fixed, and the overall wall clock time. All runs with fewer than 100 scenarios are performed on our 64-core workstation, while 100 scenario runs are performed on the Red Sky cluster. Note that unless full convergence is achieved, the solution costs correspond to a (partially) anticipative solution. Our objective in this experiment is to examine the overall nature of PH convergence on stochastic UC. Convergence acceleration mechanisms are discussed subsequently in Section V-E.

The results of this first experiment are reported in Table III. We observe that in four of the instances, PH converges to a non-anticipative solution in at most PH iterations and no longer than approximately 16 minutes of wall clock time. In cases where PH did not converge within the 30 iterations allocated, the value of the convergence metric is very small, and only a small fraction of variables remain free of the total 4080. All run times are within the range required for operational deployment. With the exception of the smaller 3 and 5-scenario instances, the PH solutions are significantly better than the EF solutions reported in Table II after 4 hours of wall clock time. Further, the lower bounds are competitive, matching those of the extensive form on the larger instances. While the comparison in the case of anticipative PH solutions is not strictly fair, the small number of non-converged variables and the value of the convergence metric is strongly suggestive of ultimate PH performance, as demonstrated subsequently.

Next, in an effort to improve solution quality, we replicate the prior experiment with two exceptions. First, we decrease the ρ scale factor from $\alpha = 1.0$ to $\alpha = 0.5$. As reported in [17], lower values of ρ can improve solution quality, albeit possibly at the expense of increased run-times. Second, we increase the limit on the number of PH iterations from 30 to 80; decreased values of ρ are generally known to delay the rate of PH convergence. The results are shown in Table IV. Relative to the results obtained using $\alpha = 1.0$, we achieve fully non-anticipative solutions in an equal number of instances, in approximately the same run-times. In cases where non-anticipative solutions are achieved with both $\alpha = 1.0$ and $\alpha = 0.5$, lower cost solutions are obtained with $\alpha = 0.5$. The relative consistency of wall clock times despite the increased number of PH iterations is due to the fact that lower α values yield smaller iteration-to-iteration perturbations to the scenario sub-problems, which in turn increase the effectiveness of warm-start solutions and consequently reduce the sub-problem solve times. In contrast, larger values of α can induce large sub-problem perturbations, such that sub-problem solve times can often significantly exceed (if a limit were not imposed) the values reported in Table I.

In our next experiment, we reduce γ from 0.03 to 0.025. Lower γ results in improved sub-problem solutions, albeit at increased run-time costs. We also increase the limit on the number of PH iterations to 100. Intuitively, we also expect increases in PH run-times, but with generally improved solutions. The results, shown in Table V, confirm this expectation. We observe that PH converges naturally, with no acceleration techniques beyond variable fixing, in all instances. Despite the increases in run time, the absolute values do not exceed approximately 25 minutes. Further, the lower bounds indicate that the achieved solutions are within 1-2% of optimality in all cases. We conjecture reduced γ also yields more stable sub-problem solutions, which in turn accelerates PH convergence. SUC is known to be highly degenerate, in that there are numerous high-quality sub-optimal solutions within a gap γ. Reducing γ reduces the degeneracy, which in turns improves PH stability by improving the likelihood that sub-problem solves from iteration to iteration yield structurally more similar solutions.

Finally, we replicate the previous experiment, with the exception that we increase the PH variable iteration fix lag μ from 3 to 6. Increased fix lags should lead to better solutions, as aggressive fixing runs the risk of premature convergence of particular generator commitments. The results, reported in Table VI, confirm our intuition: $\mu = 6$ results in consistently improved solutions and lower bounds, at the expense of slight increases in solve times (which still remain $\leq$ 30 minutes).

E. Convergence Accelerators for PH

Our results demonstrate that careful tuning of PH configuration can yield "natural" convergence to a fully non-anticipative solution. However, in general, additional mechanisms may be employed in cases where this does not occur. One example, variable slamming – discussed in Section III-B, forces non-anticipativity for non-converged variables if the convergence metric associated with PH stops decreasing. Another option

TABLE III
SOLVE AND SOLUTION QUALITY STATISTICS FOR PH EXECUTING ON THE *WECC-240-r1* INSTANCE, WITH $\alpha = 1.0$, $\mu = 3$, AND $\gamma = 0.03$

# Scenarios	Convergence Metric	Solution Cost	Lower Bound	# Variables Fixed	Run Time
3	0.0016	64396.589	62579.50	4070	640
5	0.0 (in 17 iters)	63068.793	61611.26	4078	295
10	0.0 (in 18 iters)	61449.576	60257.18	4080	427
25	0.0001 (in 27 iters)	61024.444	59811.41	4079	676
50	0.0005	60721.109	59426.69	4065	1251
100	0.0 (in 30 iters)	61202.06	59925.20	4080	1067

TABLE IV
SOLVE AND SOLUTION QUALITY STATISTICS FOR PH EXECUTING ON THE *WECC-240-r1* INSTANCE, WITH $\alpha = 0.5$, $\mu = 3$, AND $\gamma = 0.03$

# Scenarios	Convergence Metric	Solution Cost	Lower Bound	# Variables Fixed	Run Time
3	0.0 (in 32 iters)	64390.298	63075.85	4080	382
5	0.0016	62913.423	61492.04	4076	855
10	0.0 (in 38 iters)	61447.679	60293.42	4077	527
25	0.0 (in 31 iters)	60986.051	59838.03	4080	544
50	0.0060	60729.513	59401.30	4048	1367
100	0.0 (in 27 iters)	61200.52	60053.06	4079	1095

TABLE V
SOLVE AND SOLUTION QUALITY STATISTICS FOR PH EXECUTING ON THE *WECC-240-r1* INSTANCE, WITH $\alpha = 0.5$, $\mu = 3$, AND $\gamma = 0.025$

# Scenarios	Convergence Metric	Solution Cost	Lower bound	# Variables Fixed	Run Time
3	0.0 (in 39 iters)	64236.26	63080.57	4078	578
5	0.0 (in 20 iters)	62686.50	61687.98	4080	485
10	0.0 (in 60 iters)	61463.57	60367.25	4075	701
25	0.0 (in 55 iters)	61035.12	59873.22	4075	881
50	0.0 (in 74 iters)	60726.20	59477.17	4080	1511
100	0.0 (in 12 iters)	61201.48	60059.46	4069	870

TABLE VI
SOLVE AND SOLUTION QUALITY STATISTICS FOR PH EXECUTING ON THE *WECC-240-r1* INSTANCE, WITH $\alpha = 0.5$, $\mu = 6$, AND $\gamma = 0.025$

# Scenarios	Convergence Metric	Solution Cost	Lower Bound	Variables Fixed	Run Time
3	0.0 (15 iters)	64215.98	63168.71	4060	774
5	0.0 (in 7 iters)	62672.15	61723.03	3868	673
10	0.0 (in 23 iters)	61395.66	60414.61	4066	885
25	0.0 (in 18 iters)	60934.60	59932.87	4066	957
50	0.0 (in 72 iters)	60623.67	59512.28	4058	1826
100	0.0 (in 18 iters)	61200.63	60053.41	4025	1671

is to terminate PH once a sufficient number of first-stage variables have been fixed, and solve a restricted extensive form with the remaining variables free. While detailed discussion of these accelerators is beyond the present scope, we note that we do use these accelerators in more extensive testing environments than those reported above.

F. Out-of-Sample Testing

To demonstrate that our PH performance is not due to specialized tuning on the specific *WECC-240-r1* case, we now fix the configuration with $\alpha = 0.5$, $\nu = 3$, and $\gamma = 0.025$, and execute the resulting algorithm on the *WECC-240-r2* and *WECC-240-r2* cases. With one key exception noted below, the performance reported above is sustained - convergence is achieved in less than 30 minutes of wall clock time in the 100-scenario case, on the Red Sky cluster. Similarly, the lower bound quality is maintained, indicating PH is obtaining solutions within 1-2% of optimal. The noted exception in these experiments relates to modifications of the PH configuration. Specifically, it was necessary establish the initial mipgap γ dynamically, based on the average gap associated with solutions obtained after 2 minutes of wall clock time following the iteration 0 solves. In practice, the mipgap is sensitive to the scenario set, such that *a priori* fixed values can lead to either very poor initial solutions.

VI. CONCLUSIONS

Driven by the desire to directly incorporate representations of uncertainty in load and renewables output, researchers have conducted a significant amount of research into the development of algorithms for solving the stochastic unit commitment problem. Yet, these advances have not yet impacted practice, primarily due to the computational challenge of the problem. In this paper, we propose a decomposition-based strategy for solving the stochastic unit commitment problem, based on the progressive hedging algorithm of Rockafellar and Wets. Leveraging various advances over the past decade in the configuration, tuning, and lower bounding of progressive hedging in the mixed-integer case, we demonstrate tractable ($\leq$ 30 minute) solve times on the WECC-240 test case with a reasonable number of scenarios. This performance can be achieved with both small-scale multi-core workstations and commodity distributed memory clusters. Both platforms represent computing capabilities either currently deployed at ISOs and utilities, or are likely to be deployed in the near future. We are presently engaged in efforts to increase the scalability of our approach, focusing on larger-scale ISO test cases and scenarios concurrently considering uncertainty in load and renewables output.

ACKNOWLEDGMENTS

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94-AL85000. This work was funded by the Department of Energy's Advanced Research Projects Agency - Energy, under the Green Energy Network Integration (GENI) project portfolio, and by Sandia's Laboratory Directed Research and Development program.

An earlier version of some of the work presented here appeared in [21].

REFERENCES

[1] S. Takriti, J. Birge, and E. Long, "A stochastic model for the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 11, no. 3, pp. 1497–1508, 1996.

[2] Q. Zheng, J. Wang, P. Pardalos, and Y. Guan, "A decomposition approach to the two-stage stochastic unit commitment problem," *Annals of Operations Research*, To Appear.

[3] P. Ruiz, C. Philbrick, and P. Sauer, "Modeling approaches for computational cost reduction in stochastic unit commitment formulations," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 588–589, 2010.

[4] A. Papavasiliou and S. Oren, "A stochastic unit commitment model for integrating renewable supply and demand response," in *Proceedings of the 2012 IEEE Power and Energy Society Meeting*, 2012.

[5] P. Ruiz, R. Philbrick, E. Zack, K. Cheung, and P. Sauer, "Uncertainty management in the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 642–651, 2009.

[6] A. Papavasiliou, "Coupling renewable energy supply with deferrable demand," Ph.D. dissertation, University of California Berkeley, 2011.

[7] J. Price, "Reduced network modeling of WECC as a market design prototype," in *Proceedings of the 2011 IEEE Power and Energy Society General Meeting*, 2011.

[8] Y. Feng, I. Rios, S. M. Ryan, K. Spurkel, J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff, "Scalable stochastic unit commitment, part 1: Load scenario generation," *Submitted to IEEE Transactions on Power Systems*.

[9] M. Carrion and J. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem," *IEEE Transactions on Power Systems*, vol. 21, no. 3, 2006.

[10] J. Ostrowski, M. Anjos, and A. Vanneli, "Tight mixed integer linear programming formulations for the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, 2012.

[11] W. Hart, J. Watson, and D. Woodruff, "Pyomo: Modeling and solving mathematical programs in python," *Mathematical Programming Compuation*, vol. 3, no. 3, 2011.

[12] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics (SIAM), 2009.

[13] J.-P. Watson, D. Woodruff, and W. Hart, "Pysp: Modeling and solving stochastic programs in python," *Mathematical Programming Computation*, vol. 4, no. 2, 2012.

[14] R. V. Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal of Applied Mathematics*, vol. 17, 1969.

[15] R. Rockafellar and R. J.-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty," *Mathematics of Operations Research*, pp. 119–147, 1991.

[16] C. Caroe and R. Schultz, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 24, no. 1–2, 1999.

[17] J. Watson and D. Woodruff, "Progressive hedging innovations for a class of stochastic mixed-in teger resource allocation problems," *Computational Management Science*, vol. 8, no. 4, 2011.

[18] J.Goez, J. Luedtke, D. Rajan, and J. Kalagnanam, "Stochastic unit commitment problem," IBM, Tech. Rep., 2008.

[19] D. Gade, G. Hackebeil, S. Ryan, J. Watson, R. Wets, and D. Woodruff, "Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs," *Under Review*, 2013.

[20] E. Danna, E. Rothberg, and C. L. Pape, "Exploring relaxation induced neighborhoods to improve mip solutions," *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.

[21] S. Ryan, R. Wets, D. Woodruff, C. Silva-Monroy, and J. Watson, "Toward scalable, parallel progressive hedging for stochastic unit commitment," in *Proceedings of the 2013 IEEE Power and Energy Society General Meeting*, 2013.

Kwok Cheung (S87-M91-SM98) received his Ph.D. in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY in 1991. He joined Alstom Grid Inc. (formerly ESCA) in 1991. He is currently the Director of R&D worldwide market management systems at Alstom. His interests include deregulation applications and power system stability. Dr. Cheung is a registered Professional Engineer of the State of Washington since 1994.

Dinakar Gade received the B.E.(Hons) degree in mechanical engineering from BITS, Pilani, India in 2005, the M.S. degree in industrial engineering from the University of Arkansas in 2007 and the PhD degree in industrial and systems engineering from The Ohio State University in 2012. He is currently an Operations Research Contributor at Sabre Holdings in Southlake, TX.

César Silva-Monroy (M'05) works for Sandia National Laboratories in the electric power systems research group in Albuquerque, NM. He obtained his BS in electrical engineering from Universidad Industrial de Santander in Colombia, and his MS and PhD in electrical engineering from the University of Washington in Seattle. His research interests include power system operations, energy storage systems and renewable energy integration.

Sarah M. Ryan (M09) received her BS in systems engineering from the University of Virginia, and MSE and Ph.D. degrees in industrial and operations engineering from the University of Michigan. She is currently Professor in the Department of Industrial and Manufacturing Systems Engineering at Iowa State University. Her research applies stochastic modeling and optimization to the planning and operation of energy and manufacturing systems.

Jean-Paul Watson (M'10) received his Ph.D. in computer science from Colorado State University. He is currently a Principal Member of Technical Staff in the Discrete Math and Complex Systems Department at Sandia National Laboratories, in Albuquerque, New Mexico. He leads projects involving optimization under uncertainty and general analytics for US government agencies, including the Department of Energy and Defense.

Roger J.-B. Wets is a Distinguished Research Professor of Mathematics at the University of California, Davis. His research interests include stochastic optimization, variational analysis, equilibrium problems in stochastic environments, and the fusion of hard and soft information in statistical estimation. His awards include Guggenheim and Erskine Fellowships, the SIAM-MPS Dantzig Prize, and the INFORMS Lanchester prize.

David L. Woodruff is Professor at the Graduate School of Management at the University of California at Davis. He received his PhD in industrial engineering from Northwestern University. His research concerns computational aspects of multi-stage optimization under uncertainty, which includes solution algorithms, problem representation and modeling language support. He has worked on applications in a variety of areas and has been involved recently in a number of applications in power systems.