Multistage Stochastic Unit Commitment Using Stochastic Dual Dynamic Integer Programming

Jikai Zou∗ Shabbir Ahmed∗ Xu Andy Sun∗

May 14, 2017

Abstract

Unit commitment (UC) is a key operational problem in power systems used to determine an optimal daily or weekly generation commitment schedule. Incorporating uncertainty in this already difficult mixed-integer optimization problem introduces significant computational challenges. Most existing stochastic UC models consider either a two-stage decision structure, where the commitment schedule for the entire planning horizon is decided before the uncertainty is realized, or a multistage stochastic programming model with simplistic stochastic processes to ensure tractability. We propose a new type of decomposition algorithm based on Stochastic Dual Dynamic Integer Programming (SDDiP) to solve a dynamic programming formulation of a multistage stochastic unit commitment (MSUC) problem. We propose a variety of computational enhancements to adapt SDDiP to MSUC, and conduct extensive computational experiments to demonstrate that the proposed method is able to handle elaborate stochastic processes and can solve MSUCs with a huge number of scenarios that are impossible to handle by existing methods.

Keywords: stochastic unit commitment, multistage stochastic integer programming, stochastic dual dynamic integer programming

1 Introduction

1.1 Motivation

Unit commitment (UC) is one of the key problems in power system operations. It is used by system operator to decide a commitment schedule of generation units for the next day or week, under which the forecast demand can be met in the most cost efficient way. Besides satisfying the load requirements, the commitment decisions also need to satisfy certain physical constraints, e.g., generation capacity, minimum up/down time, ramping limit, as well as the flow limit of each transmission line.

In recent years, an increasing penetration of renewable energy has cast another layer of complexity to the UC problem. Due to the intermittent nature renewable energy, the grid system needs to be more flexible when dealing with uncertainty. Stochastic optimization approaches have been utilized in UC problems to achieve this goal. There are typically two modeling approaches, namely two-stage and multistage. In a two-stage model, the commitment decisions are determined day-ahead thus the generator commitment schedule is fixed regardless what happens in real time. In contrast, a multistage model handles uncertainty more dynamically. A solution to the multistage model is referred to as a policy, which system operator can use in real time to adjust the generator status according to actual load and renewable output. There has been a great amount of work to solve the UC problem in both two-stage and multistage settings. We refer

∗H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. E-mail: jikai.zou@gatech.edu, shabbir.ahmed@isye.gatech.edu, andy.sun@isye.gatech.edu
the reader to several comprehensive surveys [20; 26] for the progress in the two-stage setting, and we will summarize the existing work in the multistage setting in the next section.

1.2 Related Work

A major stream of efforts for MSUC is the development of advanced decomposition algorithms. Two different approaches have been proposed, namely unit decomposition and scenario decomposition.

In unit decomposition, the demand and spinning reserve constraints are relaxed so that each subproblem corresponds to a single generation unit. Such a decomposition scheme was first studied in [5] and has been extended by incorporating electricity contracts and spot-market prices [22]. Proximal bundle method (cf. [8]) is used in solving the Lagrangian dual problem [11; 2]. A Dantzig-Wolfe decomposition approach is studied in [19], where the single-unit subproblems are solved by dynamic programming and their schedules are added back to the restricted master problem.

Alternatively, the scenario decomposition approach attempts to relax the coupling constraints among scenarios, usually referred to as non-anticipativity constraints. The resulting subproblems then become deterministic, each of which corresponds to a single scenario. Different methods have been used to solve the relaxed problem, such as Progressive Hedging algorithm [21] and Dantzig-Wolfe decomposition [24; 17].

Besides the development of decomposition techniques, there have been efforts on strengthening the formulation using effective cutting planes. A majority of these works focus on the two-stage setting (e.g., [15; 12; 10], etc.). Recently, cutting planes have been studied for multistage models [13; 7]. Most of the works above has demonstrated the benefit (profit gain/cost reduction) of using the multistage approach compared to the deterministic or two-stage setting. However, the solution methods are tailored for relatively small scenario trees considered in the formulation thus may not be implementable to other possible scenarios.

Pereira and Pinto [14] propose a stochastic variant of nested Benders decomposition, known as Stochastic Dual Dynamic Programming (SDDP) to solve multistage stochastic linear programs. It has been successfully applied to multistage stochastic hydro-thermal scheduling problems since then (e.g., [16; 1; 18]). SDDP is a sampling-based algorithm and benefits from stage-wise independence of the underlying stochastic process. Due to the presence of integer recourse decisions in MSUC, such Benders-type decomposition algorithm has its own limitation, thus has only been used in a two-stage setting (see e.g., [6; 23; 25]). An extension of SDDP to solve multistage stochastic integer programming (MSIP) problems, called Stochastic Dual Dynamic integer Programming (SDDiP), is proposed in [27]. The algorithm is designed for MSIP with binary state variables, and can also accommodate general state variables under mild conditions. In SDDiP algorithm, state variables refer to those whose values will be passed to the next stage problem as inputs, and the rest are referred to as local variables. A new family of cuts, termed Lagrangian cuts, is proposed in this work and is shown to be the key to close optimality gap and to guarantee almost sure convergence of SDDiP.

1.3 Contributions

In this paper, we consider an MSUC problem with uncertain net load, i.e., the difference between the total demand and the renewable output. It captures the uncertainty from both the demand and supply sides. We solve the MSUC problem with an objective to minimize the total expected cost of start-up, shut-down, generation, and possible penalties. The key contributions are summarized below.

1. We propose a new type of decomposition algorithm based on the SDDiP algorithm to solve large-scale MSUC problems. To the best of our knowledge, this is the first attempt to tackle the MSUC problem using a stage-wise decomposition framework. Upon termination of the algorithm, a multistage, implementable policy is returned. Operators can use such a policy in real time to deal with system uncertainties.

2. We propose several enhancement of the SDDiP algorithm to improve running time, including using the Level Method to compute the Lagrangian cuts, a “hybrid” mixed-integer and linear modeling approach with the notion of “breakstage”, and a parallel implementation of the backward step in SDDiP.
3. Extensive computational experiments are conducted on the IEEE 14-bus and 118-bus systems. We study the effectiveness of three types of cuts for solving the MSUC problem and the impact of breakstage. Our experiments show that the proposed method can handle MSUCs with a huge number of scenarios that were considered impossible before.

The remainder of the paper is organized as follows. In Section 2, we describe the SDDiP algorithm and different cut families. In Section 3, we present the mathematical formulation for MSUC. In Section 4, we describe various computational enhancements of SDDiP for solving MSUC. Sections 5-6 discuss experiment settings and detailed computational results. Finally, we provide some concluding remarks in Section 7.

2 Stochastic Dual Dynamic Integer Programming

We start with a scenario tree formulation of an MSIP problem with binary state variables.

\[
\begin{align*}
\min_{x_n, y_n, z_n} & \quad \sum_{n \in T} p_n g_n(x_n, y_n) \\
\text{s.t.} & \quad \forall n \in T \\
& \quad (z_n, x_n, y_n) \in X_n \\
& \quad z_n = x_{a(n)}, z_n \in [0, 1]^d \\
& \quad x_n \in \{0, 1\}^d.
\end{align*}
\]

In the above formulation, \( x_n \in \{0, 1\}^d \) is the state variable of node \( n \) in the scenario tree \( T \) and \( y_n \) is the local variable. In addition, \( z_n \in [0, 1]^d \) is another continuous local variable which is a copy of the state variable from previous stage. Note that in this formulation only successive stages are linked together. This can be always be ensured by a proper reformulation of the problem. We assume that (2.1) has complete continuous recourse, i.e., given any value of the state and local integer variables, there exist a value of continuous local variables such that (2.1) is feasible. If the state variables are not binary, we can use binary expansion/approximation to transform them into binary. Suppose \( x \in [0, U] \) is a continuous state variables, we substitute it by \( \sum_{i=0}^{\kappa} 2^{i-1} \epsilon \lambda_i \), where \( \lambda_i \in \{0, 1\} \), \( \kappa = \lfloor \log_2(U/\epsilon) \rfloor + 1 \), and \( \epsilon \) is the approximation accuracy. In this way \( |x - \sum_{i=0}^{\kappa} 2^{i-1} \epsilon \lambda_i| \leq \epsilon \). For general integer variables, setting \( \epsilon = 1 \) results in an exact representation.

Now we can write down the DP equations for the optimal value function of the multistage problem (2.1) as follows:

\[
(P_1) \quad \min_{x_1} g_1(x_1, y_1) + \sum_{m \in C(1)} q_{1m} Q_m(x_1)
\]
\[
\text{s.t. (2.1a) - (2.1c) (for } n = 1)\]

and for each node \( n \in T \setminus \{1\} \),

\[
(P_n) \quad Q_n(x_{a(n)}) := \min_{x_n, y_n} g_n(x_n, y_n) + \sum_{m \in C(n)} q_{nm} Q_m(x_n)
\]
\[
\text{s.t. (2.1a) - (2.1c),}
\]

where \( q_{nm} \) is the conditional probability from \( n \) to its children node \( m \). We will refer to \( Q_n(\cdot) \) as the optimal value function (of \( x_{a(n)} \)) at node \( n \) and denote the function \( Q_n(\cdot) := \sum_{m \in C(n)} q_{nm} Q_m(\cdot) \) as the expected cost-to-go (ECTG) function at node \( n \).

In the SDDiP algorithm, the scenario tree satisfies stage-wise independence, i.e., for any two nodes \( n \) and \( n' \) in stage \( t \), the set of their children nodes \( C(n) \) and \( C(n') \) are defined by identical data and conditional probabilities. In this case, the ECTG functions depend only on the stage rather than the nodes, i.e., we have \( Q_n(\cdot) \equiv Q_t(\cdot) \) for all \( n \) in stage \( t \). More specifically, let \( N_t \) be the number of possible data realizations at stage \( t = 2, \ldots, T \), each outcome has an equal probability of \( 1/N_t \). Then total number of scenarios is \( N = \prod_{t=2}^{T} N_t \).
By exploiting the stage-wise independence of the underlying stochastic process, SDDiP algorithm proceeds stage-wise from \( t = 1 \) to \( T \) by solving a dynamic programming equation with an approximated ECTG function at each stage. The ECTG function at each stage is approximated from below by a convex piece-wise linear function, and these approximations are improved through iterations. In particular, the stage problem in the \( i \)-th iteration \( P_i^t(x_{i-1}^t, \psi_i^t, \xi_i^t) \) is of the following form:

\[
Q^t_i(x_{i-1}^t, \psi_i^t, \xi_i^t) := \min_{x_t, y_t, z_t} g_t(x_t, y_t, \xi_i^t) + \psi_i^t(x_t)
\]

\[
\text{s.t. } (x_t, y_t, z_t) \in X_t(\xi_i^t)
\]

\[
z_t = x_{i-1}^t
\]

\[
x_t \in \{0, 1\}^d, \quad z_t \in [0, 1]^d,
\]

where \( \psi_i^t(\cdot) \) is defined as:

\[
\psi_i^t(x_t) := \min \{ \theta_t : \quad \theta_t \geq L_t, \quad \theta_t \geq \frac{1}{N_{t+1}} \sum_{j=1}^{N_{t+1}} (v^f_{ij} + (\pi_i^j x_t)^\top), \forall \ell = 1, \ldots, i-1 \}.
\]

The function \( \psi_i^t(x_t) \) is the current convex lower-approximation of the true ECTG function \( Q_t(x_t) \).

Given a solution \( x_{i-1}^t \) from the previous stage, the current approximation \( \psi_i^t(\cdot) \), and a particular uncertainty realization \( \xi_i^k(1 \leq k \leq N_i) \) at stage \( t \), the forward problem in iteration \( i \) is fully characterized. We assume that a lower bound exists for the ECTG function to avoid unboundedness. Once a forward iteration is completed, we have obtained a feasible solution \( \{(x_t^i, y_t^i)\}_{i=1}^T \) for the corresponding scenario.

The backward step starts from the last stage \( T \). Given the solution \( x_{T-1}^t \) from iteration \( i \) and a particular uncertainty realization \( \xi_T^j(1 \leq j \leq N_T) \), let \( R_T^i \) be a relaxation of the forward problem \( P_T(x_{T-1}^t, \psi_T^{i+1}, \xi_T^j) \). Note that \( \psi_T = 0 \) for all \( i \geq 0 \). Solving \( R_T^i \) for each \( j \) produces a linear inequality defined by \((v^f_{ij}, \pi_i^j x_t)\) and it is valid for the value function \( Q_T(x_{T-1}^t, \xi_T^j) \). Then the inequalities are aggregated to obtain one of form (2.3b), which is valid for ECTG function \( Q_{T-1}(x_{T-1}^t) \). The lower approximation of the ECTG function is updated from \( \psi^{i-1}_{T-1}(\cdot) \) to \( \psi^{i+1}_{T-1}(\cdot) \). The backward step then proceeds to stage \( T - 1 \). When the first stage is completed, since we have solved a lower approximation of the original problem, the optimal value of the first stage problem is an exact lower bound of the original problem.

One can also generate more than one scenario in the forward step. These single scenario cost can be used to compute a confidence interval of their mean value. The mean value of these costs is usually referred as a statistical upper bound for the original problem. In fact, common stopping criteria for SDDP-type algorithm is often based on this statistical upper bound and the exact lower bound obtained from the backward step. A complete description of the SDDiP algorithm and the proof of almost surely convergence can be found in [27].

2.1 Cut Families in Backward Step

Depending on the relaxation problems \( R_T^i \) solved in the backward step, different families of linear inequalities can be obtained. In this paper, we implement the SDDiP algorithm with standard Benders’ cuts, a type of Lagrangian cuts obtained from a particular reformulation, and strengthened Benders’ cuts, which are a byproduct of the Lagrangian cuts.

2.1.1 Benders’ cut[3]

These cuts are derived from the LP relaxation of \( P_i^t(x_{i-1}^t, \psi_i^t, \xi_i^t) \), and the cut coefficients \( (v^{ij}_i, \pi_i^{ij}) \) correspond to the optimal value of the LP relaxation and a basic feasible dual solution. To be precise, the cut
added to $\psi^{i}_{t-1}$ takes the form:

$$\theta_{t-1} \geq \frac{1}{N_t} \sum_{j=1}^{N_t} Q_{i,t}^{ij} + \frac{1}{N_t} \sum_{j=1}^{N_t} (\pi_{LP,t}^{ij})^\top (x_{t-1} - x_{i-1}^i),$$  \hspace{1cm} (2.4)$$

where $Q_{i,t}^{ij}$ is the optimal LP relaxation objective function value of problem $P_i^t(x_{t-1}^i, \psi_{t+1}^{i+1}, \xi_{t}^{i})$, and $\pi_{LP,t}^{ij}$ is a basic optimal dual solution in connection with the constraints $z_t = x_{i,t-1}$. Benders’ cut are in general not tight when integer variables are present. Therefore almost sure convergence is not guaranteed if only these cuts are used.

2.1.2 Lagrangian cut

This family of cuts is based on solving a Lagrangian dual of $P_i^t(x_{t-1}^i, \psi_{t+1}^{i+1}, \xi_{t}^{i})$. In particular, the relaxation problem $R_{i,t}^{ij}$ is

$$(R_{i,t}^{ij}) : \max_{\pi_t} \left\{ L_{i,t}^{ij}(\pi_t) + \pi_t^\top x_{i-1}^i \right\} \hspace{1cm} (2.5)$$

where

$$L_{i,t}^{ij}(\pi_t) = \min g_t(x_t, y_t, z_t) + \psi_t^i - \pi_t^\top z_t$$

\hspace{1cm} s.t. \hspace{0.3cm} (x_t, y_t, \xi_{t}^{i}) \in X_t(\xi_{t}^{i})$$

$$x_t \in \{0, 1\}^d, \hspace{0.2cm} z_t \in [0, 1]^d$$  \hspace{1cm} (2.6)$$

The cut coefficients $(v_{i,t}^{ij}, \pi_{i,t}^{ij})$ are then equal to $L_{i,t}^{ij}(\pi_{LG,t}^{ij})$ and $\pi_{LG,t}^{ij}$, where $\pi_{LG,t}^{ij}$ is an optimal solution of the Lagrangian dual problem $R_{i,t}^{ij}$. It can be proven that the Lagrangian dual problem has zero duality gap and almost sure convergence is guaranteed if these cuts are used in the backward step [27].

2.1.3 Strengthened Benders’ cut.

Instead of solving the Lagrangian dual problem to optimality, we can solve (2.6) with $\pi_t = \pi_{LP,t}^{ij}$. The cut then takes the form

$$\theta_{t-1} \geq \frac{1}{N_t} \sum_{j=1}^{N_t} L_{i,t}^{ij}(\pi_{LP,t}^{ij}) + \frac{1}{N_t} \sum_{j=1}^{N_t} (\pi_{LP,t}^{ij})^\top (x_{t-1} - x_{i-1}^i).$$

The cut is valid because $\pi_{LP,t}^{ij}$ is feasible to (2.5). It is parallel to and at least as tight as standard Benders’ cuts.

3 Multistage Stochastic UC

3.1 Problem Formulation

We present an extensive formulation for MSUC where the uncertain net load is modeled by a scenario tree. Before presenting the mathematical formulation, we summarize the notation in Table 1.

<table>
<thead>
<tr>
<th>Sets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{T}$</td>
<td>Scenario tree</td>
</tr>
<tr>
<td>$\mathcal{B}$</td>
<td>Set of all buses</td>
</tr>
<tr>
<td>$\mathcal{G}$</td>
<td>Set of generation units</td>
</tr>
</tbody>
</table>
\[ \mathcal{G}_b \] Set of generation units at bus \( b \)
\[ \mathcal{D} \] Set of demand bus
\[ \mathcal{L} \] Set of all transmission lines
\[ \mathcal{P}(n, t) \] Path from the \( t \)-th ancestor node of \( n \) to \( n \)

**Indices**

- \( n, m \): Node in the scenario tree
- \( i \): Generation unit
- \( b \): Load bus
- \( \ell \): Transmission line
- \( t \): Decision stage
- \( t(n) \): Decision stage of node \( n \)

**Parameters**

- \( p_n \): Probability associated with node \( n \)
- \( S_i, S_i \): Start-up/shut-down cost of generation unit \( i \)
- \( C_p \): Penalty cost for unsatisfied demand and over generation
- \( D_{nb} \): Load demand of bus \( b \) at node \( n \)
- \( F_\ell \): Maximum flow capacity of line \( \ell \)
- \( K_\ell \): Vector of the shift vectors of transmission line \( \ell \)
- \( P_i, P_i \): Maximum/minimum output of generation unit \( i \)
- \( R_t \): Reserve requirement in period \( t \)
- \( \Delta_{SU}^i, \Delta_{SD}^i \): Regular ramp up/down rate for generation unit \( i \)
- \( UT_i, DT_i \): Minimum up/down time for generation unit \( i \)

**Decision variables**

- \( x_{ni} \): State of unit \( i \) at node \( n \), equals 1 if on, 0 otherwise
- \( y_{ni} \): Generation by unit \( i \) at node \( n \)
- \( u_{ni} \): Start-up decision for unit \( i \), equals 1 if it is turned on at node \( n \), 0 otherwise
- \( v_{ni} \): Shut-down decision for unit \( i \), equals 1 if it is turned off at node \( n \), 0 otherwise
- \( r_{ni} \): Reserved spinning capacity from unit \( i \) at node \( n \)
- \( \delta_n^+, \delta_n^- \): Total unsatisfied demand at node \( n \)
- \( \delta_n^- \): Total over-generation at node \( n \)

A multistage stochastic programming formulation can be written as follows.

\[
\text{min} \quad \sum_{n \in \mathcal{T}} p_n \left[ \sum_{i \in \mathcal{G}} \left(S_i u_{ni} + S_i v_{ni} + f_i(y_{ni}) + C_p(\delta_n^+ + \delta_n^-) \right) \right]
\]

s.t. \( \forall n \in \mathcal{T}, \)

\[
\sum_{i \in \mathcal{G}} y_{ni} + \delta_n^+ - \delta_n^- = \sum_{b \in \mathcal{D}} D_{nb}
\]

\[
\sum_{i \in \mathcal{G}} r_{ni} \geq R_t(n)
\]

\[
\left| \sum_{b \in \mathcal{B}} K_{ib} \left( \sum_{i \in \mathcal{G}_b} y_{ni} - D_{nb} \right) \right| \leq F_\ell, \forall \ell \in \mathcal{L}
\]

\[
y_{ni} + r_{ni} \leq P_i x_{ni}, \quad y_{ni} \geq P_i x_{ni}, \forall i \in \mathcal{G}
\]

\[
y_{ni} - y_{a(n), i} \leq \Delta_{SU}^i u_{ni} + \Delta_i x_{a(n), i}, \forall i \in \mathcal{G}
\]

\[
y_{a(n), i} - y_{ni} \leq \Delta_{SD}^i v_{ni} + \Delta_i x_{ni}, \forall i \in \mathcal{G}
\]
where $\xi$ (3.1e). Constraints (3.1f) specify the output capacities for each generator. Ramping constraints are enforced by (3.1k)-(3.1l). These constraints are proposed in [15] for the two-stage setting and the authors show that such a formulation describes the convex hull of the minimum up and down time polytope. It is shown in [7] that such a formulation is also tight in the stochastic setting.

Lastly, (3.1m) contains the binary and non-negativity constraints for the decision variables. Any initial state of the system can be accommodated by constraints (3.1g)-(3.1l). In our experiments, we assume that each generator is off and has met the minimum downtime requirement thus can be turned on immediately.

### 3.2 Stage-wise Independence in Net Load

The demand process and renewable output are usually correlated across different hours. However, they exhibit certain patterns throughout a day. We assume that a base net load profile (24-hour resolution) is given and the true net load deviates from this profile according to a given distribution. The deviations at each hour are assumed to be independent. Specifically, let $\{D_t\}_{t=1}^{T}$ be the nominal hourly net load profile. For each $t = 1, \ldots, T$, the true net load $\tilde{D}_t$ is generated as follows:

$$\tilde{D}_t = D_t \cdot \xi_t,$$

where $\xi_t \sim \Xi_t$, and $\Xi_t$ is the inferred distribution from historical data. The total net load across the network is then allocated to each load bus according to the ratio implied from the base load profile, i.e., the proportion of the net load at each bus is the same for all realizations within the same stage.

### 3.3 State Variables

In formulation (3.1), decisions at node $n$ are dependent on information passed from node $n$’s ancestors. This information includes the generator state $x_{a(n)}$ of the last stage, the generation level $y_{a(n)}$ at the previous stage, and a sequence of commitment decisions from earlier stages, i.e., $\{y_{m}, m \in \mathcal{P}(n, UT - 1)\}$, and $\{v_{m}, m \in \mathcal{P}(n, DT - 1)\}$. They are the state variables, and have a dimension of $\sum_{i \in \mathcal{G}}(UT_i + DT_i)$.

One obstacle of applying SDDiP directly to MSUC is the minimum up and down time constraints. They link variables from more than two stages, while SDDiP requires that the current stage problem depend only on the previous stage. To resolve this problem, we reformulate these constraints by making copies of decisions from nodes in $\mathcal{P}(n, UT - 1)$ and $\mathcal{P}(n, DT - 1)$. More specifically, we create two sets of new variables at each node $n$: $\{u_{ni}^{(k)}, k = 0, \ldots, UT - 1\}$ and $\{v_{ni}^{(k)}, k = 0, \ldots, DT - 1\}$. Constraints (3.1k)-(3.1l) are then equivalent to the following set of inequalities:

$$\sum_{k=0}^{UT-1} u_{ni}^{(k)} \leq x_{ni}, \quad \sum_{k=0}^{DT-1} v_{ni}^{(k)} \leq 1 - x_{ni},$$

$$u_{ni}^{(0)} - u_{ni} = 0, \quad v_{ni}^{(0)} - v_{ni} = 0,$$

$$u_{ni} \geq 0, v_{ni} \geq 0, \quad \forall i \in \mathcal{G}.$$
u_{ni}^{(k)} = u_{a(n),i}^{(k-1)}, \forall k = 1, \ldots, UT_i - 1 \quad (3.3c)
\nu_{ni}^{(k)} = v_{a(n),i}^{(k-1)}, \forall k = 1, \ldots, DT_i - 1 \quad (3.3d)

As a result, it is sufficient to pass $x_{a(n),i}, y_{a(n),i}, \{u_{a(n),i}^{(k)}\}_{k=0}^{UT_i-2}$, and $\{v_{a(n),i}^{(k)}\}_{k=0}^{DT_i-2}$ to node $n$, all of which comes from the parent node $a(n)$.

Moreover, SDDiP requires all state variables to be binary. In MSUC, generator states $(x)$ and commitment decisions $(u,v)$ are already binary, however, the dispatch decision $(y)$ is continuous and requires a binary approximation.

With the above two modeling treatment and the stage-wise independence assumption, the MSUC problem (3.1) can be reformulated as a DP equation ready for SDDiP, and the state space has a dimension of $\sum_{i \in G} (UT_i + DT_i + \lceil \log_2(P_i/\epsilon) \rceil)$.

4 SDDiP Enhancements

In this section, we describe several enhancements to the basic SDDiP method described previously.

4.1 Level Method for Lagrangian Cut

To obtain the cut coefficients of the Lagrangian cuts, one needs to solve the Lagrangian dual problem (2.5). This is a non-smooth convex optimization problem often solved by a subgradient method (see e.g.,[4]). We propose to use the Level Method due to [9]. This method is similar to a cutting plane method that proceeds by considering an approximation or model of the objective function constructed from subgradients evaluated at preceding iterates. The next iterate is obtained by projecting a minimizer of the model function to an appropriate level set so that its objective value lies in some neighborhood of the objective of the current iterate. In this way the iterates are regularized and the method achieves a theoretical optimal convergence rate. It has also been proven to be very effective in practice. In Section 6, we compare the performance of the Level Method with a basic subgradient algorithm for obtaining Lagrangian cut coefficients.

4.2 Hybrid Model using “Breakstage”

The quality of a policy obtained by SDDiP depends on the quality of the approximation of the ECTG in each stage. Intuitively, the stages further in the future has less influence on the current stage. Motivated by this, we propose a hybrid modeling approach, which allows us to improve the solution time while not compromising its quality to a large extent. This approximation relies on a prescribed stage $t_b$, which we will refer to as the breakstage hereafter. More specifically, in any decision stage before $t_b$, we solve the formulation $P_t^i (x_{t-1}^i, \psi_{t+1}^i, \xi_{t}^i, \zeta_{t}^i)$, where state variables are binary. From stage $t_b$ onward, we change the state variables back into their original space. It is still valid to add all three types of cuts at every stage, except that the Lagrangian cuts after $t_b$ are not guaranteed to be tight.

In our experiments, we further relax all integrality constraints after $t_b$ to improve solution time. As a result, only Benders’ cuts are added for stage problems after $t_b$. If $t_b = 0$, the method reduces to SDDP applied to the LP relaxation of the original formulation; if $t_b = T + 1$, the fully discretized problem is solved by SDDiP.

The breakstage gives us the flexibility to evaluate the trade-off between solution time and solution quality. Solving the LP relaxation and using a state space of smaller dimension both contribute to the reduction of solution time. In addition, one can always adjust the policy if new information, e.g., a more accurate renewable output forecast, becomes available.
4.3 Backward Parallelization

In the backward step of SDDiP, multiple scenario problems are solved, then the cut coefficients returned by each of them are aggregated to produce a cut for its previous stage problem. Since these scenario problems are independent from each other, we implement a simple parallelization scheme using OpenMP to speed up the backward step.

5 Experimental Settings

In this section, we discuss our experimental settings. The 14-bus system has 5 generators, 20 transmission lines, and 11 demand buses; the 118-bus system includes 54 generators, 186 transmission lines, and 91 demand buses. Most data about the physical electrical network is from MatPower 6.0. Ramping limit is set to be 80% of the maximum generation capacity or specified otherwise. Minimum up and down times vary from 1 to 10 hours. To avoid infeasibility, slack variables are added to the load balance constraints and penalized with a large cost in the objective function. All penalty costs are assumed to be $5000 per MW.

5.1 Stage Problem Size

Deriving strengthened Benders’ cuts and Lagrangian cuts require solving MIPs in each stage. Therefore, the size of the stage problem greatly affects the solution time. In the 14-bus system, the numbers of (binary) state variables, integer local variables, and continuous local variables are 127, 10, and 174, respectively. For the 118-bus system, the corresponding numbers are 1086, 108, and 1514.

5.2 Scenario Tree Generation

To generate a recombining scenario tree, we start with a given net load in the first stage (12am, \( t = 1 \)). At each following hour (stage), realizations are independently generated according to (3.2). For the 14-bus system, we assume \( \xi_t \sim U(1 - \alpha, 1 + \alpha) \) for all \( t > 1 \), and \( \alpha \in [0.1, 0.3] \). Six types of scenarios trees are generated, each of them is characterized by net load variation (\( \alpha \)) and the number of outcomes at each stage (\( \beta \)). The corresponding tree is denoted by \( T_{14}^{\alpha, \beta} \). In our experiments, we consider \( \alpha = 0.1, 0.2, 0.3 \) and \( \beta = 10, 20 \).

For the 118-bus system, we use a truncated normal distribution, which is estimated based on data from California ISO website. We used the hourly net load forecast and the actual net load data across the entire California network in February 2017. The forecast is generated day ahead. For each hour, the distribution of forecast-to-actual ratio is approximated by a normal distribution. We assume \( \xi_t \sim TN(\mu_t, k^2\sigma_t^2) \), where \( TN(\mu_t, k^2\sigma_t^2) \) is the normal distribution \( N(\mu_t, k^2\sigma_t^2) \) truncated between \( \mu_t \pm 3k\sigma_t \), and \( \mu_t, \sigma_t \) are estimated from historical data. The scenario tree, denoted by \( T_{118}^{k, \beta} \), is then characterized by \( k \) and the number of
outcomes at each stage ($\beta$). In our experiments, we fix $\beta = 20$ and consider $k$ varying from 1.0 to 1.3. Figure 1 is an illustration of 50 independent scenarios from scenario tree $T_{14}^{0.2,20}$ (left) and $T_{118}^{1.3,20}$ (right).

5.3 Other Implementation Details

In the forward step of the SDDiP algorithm, we generate candidate solutions for five independent sample paths, and in the backward step, we evaluate two of them which result in the highest cost. The Lagrangian dual problem is solved to optimality using a basic subgradient algorithm and the Level Method with an optimality tolerance of $10^{-4}$ for the 14-bus system and $5 \times 10^{-4}$ for the 118-bus system, respectively. Other relative MIP tolerance is set to be the same as above for each system. The SDDiP algorithm is implemented in C++ with CPLEX 12.7.0 to solve the MIP and LP subproblems. All experiments are performed on a 16-core machine with Intel Xeon E5-2630 v3 @2.40GHz CPUs and 128GB of main memory. Reported solution times are wall clock times.

6 Computational Results

6.1 14-bus Results

We generate six different instances for the 14-bus system: $T_{14}^{\alpha,\beta}$, $\alpha = 0.1, 0.2, 0.3$ and $\beta = 10, 20$. An instance with $\beta = 10$ involves, $10^{23}$ scenarios, and its extensive form has over $2.5 \times 10^{24}$ variables, motivating the need for a sampling based decomposition method such as SDDiP. For each instance, the designed experiment consists of two phases: Phase I, Run SDDiP and obtain a policy; Phase II, Evaluate the policy with restored integrality constraints. For each $(\alpha, \beta)$ pair, we generate two scenario trees, the first one is used in Phase I to obtain policies, and the other is used for evaluation in Phase II.

In Phase I, we solve SDDiP with different breakstages ($t_b$). As mentioned earlier, when $t_b = 0$, SDDiP reduces to standard SDDP. If $t_b = 1$, nothing changes except that the first-stage problem becomes a MIP. When $t_b > 1$, other types of cuts may be added to the stage problems before $t_b - 1$. In particular, we consider five different cut combinations: Benders’ cut only (B), strengthened Benders’ cut only (SB), Lagrangian cut obtained by subgradient method (Sub), Lagrangian cut obtained by the Level Method (Level), and strengthened Benders’ cut plus Lagrangian cut obtained by the Level Method (SB + Level).

Once $t_b$ and cut families are determined, SDDiP starts. In the first half of iterations, we ignore any integrality constraints and only turn on Benders’ cuts to get a rough estimation of the ECTG functions. In the second half, we restore these integrality constraints and add other types of cuts to improve the estimation. The final statistical upper bound is evaluated based on a set of 800 independent forward sample paths. SDDiP terminates after a fixed number of iterations.

In Phase II, we reinsert the integrality constraints in stage problems after $t_b$. A set of 800 scenarios is sampled independently from the second scenario tree, forward problems are solved with the policy obtained in Phase I, and the cost associated with each scenario is recorded. The performance of the policy is evaluated by comparing the lower bound returned by SDDiP in Phase I, with the right endpoint of 95%-CI for the sample mean of scenario costs obtained from Phase II. All results in this section are averaged over 3 independent runs.

We discuss our findings with respect to following three aspects:

1. Which cut combination(s) perform the best in SDDiP?
2. What is the effect of different choices of breakstage?
3. What is the speed-up ratio and parallel efficiency from the backward parallelization?
6.1.1 Cut Combinations

To test the power of different families of cuts, we solve each instance with breakstage $t_b = 25$, i.e., the fully discretized problem. In the forward step, we solve MIPs to obtain binary candidate solutions, and in the backward step, different cuts are generated by evaluating these solutions. The power of each cut family is assessed based on SDDiP gap, solution time, and evaluation gap. The number of iterations in SDDiP is fixed at 150 for instances with $\alpha = 0.1, 0.2$ and 500 for instances with $\alpha = 0.3$.

Figure 2 shows the SDDiP results of the six instances with different cut combinations. The figure on the left presents the gap between the final lower bound and statistical upper bound. The one on the right contains the solution time of the SDDiP algorithm. The horizontal axis indicates the instances indexed by the $(\alpha, \beta)$ pair.

![SDDiP results with different cut combinations](image)

Figure 2: SDDiP results with different cut combinations. Horizontal axis indicates instance label $(\alpha, \beta)$, where $\alpha$ represents the demand variation ($\mathcal{U}(1 - \alpha, 1 + \alpha)$), $\beta$ represents the number of branches in the scenario tree. SDDiP gap and time are evaluated upon termination: 150 iterations for Instance 1–4, and 500 for Instance 5 and 6.

Clearly, $\text{SB+Level}$ and $\text{Level}$ yield the smallest gap with a reasonable solution time among all. When the net load variation is small, using any type of these cuts is sufficient. When the variation becomes bigger, however, at least one family of tight cuts is needed to close the gap. Strengthened Benders’ cut slightly improves the SDDiP gap of only using Benders’ cut. Even though Lagrangian cuts and strengthened Benders’ cuts are not dominated by each other, there is a significant improvement in SDDiP gap when Lagrangian cuts are used. In addition, it is evident that the Level Method performs better than the subgradient method.

The Phase II evaluation results are summarized in Figure 3. $\text{SB+Level}$ and $\text{Level}$ produce the most stable policies and yield the tightest statistic upper bound estimation. The policy approximated by Lagrangian cuts using subgradient method is again shown to be inferior to the one with the Level Method. In addition, we can observe a large evaluation gap for the policy characterized by the strengthened Benders’ cut in the instance $(0.3, 10)$. A possible reason is that 10 realizations per stage is not enough to represent the uncertainty with such big variation, the scenario tree used in the evaluation phase has some extreme scenarios not assessed in Phase I.

In summary, $\text{SB+Level}$ or $\text{Level}$ is the best cut combination for SDDiP, and solving the Lagrangian dual problem using the Level Method is more efficient and stable.
6.1.2 Effect of Breakstage

We next study the hybrid modeling approach proposed in Section 4.2. In particular, we choose 6 values for \( t_b \), ranging from 0 to 25. When \( t_b = 0 \), the standard SDDP algorithm is used to solve the LP relaxation of the original problem. When \( t_b > 1 \), both strengthened Benders’ cuts and Lagrangian cuts (using the Level Method) are used in the backward step for stage problems before \( t_b \). The number of iterations in SDDiP is fixed at 150 for instances with \( \alpha = 0.1, 0.2 \) and 500 for instances with \( \alpha = 0.3 \).

Figure 4: Effect of breakstage. SDDiP gap and time are evaluated upon termination: 150 iterations for Instance 1–4, and 500 for Instance 5 and 6.

Figure 4 summarizes the effect of different breakstage values. The solution time for the SDDiP algorithm increases as the breakstage increases. This is simply because more MIPs are solved as \( t_b \) increases. On average, when the breakstage increases from 0 (LP case) to 25 (fully discretized problem), the solution time increases by a factor of 4. SDDiP gap is not reported here since the algorithm terminates with a gap smaller than 0.6% for all these instances. The right figure in Figure 4 summarizes the evaluation results. The evaluation gap tends to decrease as breakstage increases. For instances with smaller net load variation, a policy obtained by solving an approximation model with small breakstage is sufficiently good, i.e., evaluation gap is small.
When the uncertainty variation is high (e.g., $\alpha = 0.3$), such a policy results in too much penalty. Therefore, solving an approximation model to optimality does not necessarily imply that SDDiP has produced a good policy, sometimes the effort of recovering the true ECTG function at each stage is necessary.

### 6.1.3 Backward Parallelization

Let $T(k)$ be the solution time when $k$ threads are used. We define speed-up ratio by $\frac{T(1)}{T(k)}$, and efficiency by $k \frac{T(1)}{T(k)}$. Figure 5 depicts an average speed-up ratio and efficiency graph with respect to the number of threads for a particular instance. We use 32 threads in all of our computation experiments. On average, the maximum speed-up ratio is 4.8 with an efficiency of 15%.

![Speed-up ratio and efficiency](image)

Figure 5: Parallelization speed-up ratio & efficiency ($T_{14}^{0.2, 20}$ instance)

### 6.2 118-bus Results

Similar to the 14-bus system, the experiments for the 118-bus system also consists of two phases: SDDiP and policy evaluation. We fix $\beta = 20$ in each scenario tree tested. An instance involves, $2 \times 10^{24}$ scenarios, and its extensive form has over $5.4 \times 10^{26}$ variables. Each instance is indexed by a pair $(r, k)$, where $r$ is ramping ratio with respect to the output capacity, and $k$ is the parameter in the truncated normal distribution. We consider twelve instances with $r = 0.9, 0.8, 0.7$ and $k = 1.0, 1.1, 1.2, 1.3^1$. A smaller $r$ indicates more restricted ramping constraints, while a larger $k$ value suggests a more volatile scenario tree.

We combine SDDiP with sample average approximation (SAA) to evaluate the quality of returned policy. For each pair of $(r, k)$, we generate six scenario trees independently. We solve SDDiP on the first five trees (the algorithm is terminated after 500 iterations), and test each returned policies on the sixth tree. SDDiP results are taken as average over the five runs on the first five trees, and the final evaluation results are the 95%-CIs calculated based on the five policy assessments on the sixth tree.

Table 2 contains the SDDiP computation time and evaluation results for the 118-bus system. The results indicate using SDDiP with Benders’ cut only is sufficient to produce an accurate and reliable policy for all 12 instances. This could be due to the tight formulation of a single scenario deterministic UC problem. To verify the tightness of the LP relaxation gap, we independently generate 100 scenarios from the most volatile load distribution ($k = 1.3$), and solve a deterministic 24-hour UC problem and its LP relaxation for each of the scenarios. The ramping limit is set to be 70% of the maximum generation capacity. Indeed, the average LP gap over these 100 instances is only 0.254%. Given that our uncertainty variation is based on real data, such a

---

^1We do not consider value bigger than 1.3 because a larger value incurs a net load which exceeds the system’s total generation capacity.
Table 2: Computational results for 118-bus system

<table>
<thead>
<tr>
<th>Instance $(r, k)$</th>
<th>Time (sec.)</th>
<th>Eval. Gap $(%)$</th>
<th>Instance $(r, k)$</th>
<th>Time (sec.)</th>
<th>Eval. Gap $(%)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.9, 1.0)</td>
<td>4389</td>
<td>[0.47, 0.68]</td>
<td>(0.8, 1.2)</td>
<td>4424</td>
<td>[0.51, 0.75]</td>
</tr>
<tr>
<td>(0.9, 1.1)</td>
<td>4387</td>
<td>[0.51, 0.59]</td>
<td>(0.8, 1.3)</td>
<td>4455</td>
<td>[0.55, 0.96]</td>
</tr>
<tr>
<td>(0.9, 1.2)</td>
<td>4394</td>
<td>[0.50, 0.77]</td>
<td>(0.7, 1.0)</td>
<td>4389</td>
<td>[0.37, 0.63]</td>
</tr>
<tr>
<td>(0.9, 1.3)</td>
<td>4405</td>
<td>[0.55, 0.69]</td>
<td>(0.7, 1.1)</td>
<td>4427</td>
<td>[0.58, 0.84]</td>
</tr>
<tr>
<td>(0.8, 1.0)</td>
<td>4333</td>
<td>[0.48, 0.63]</td>
<td>(0.7, 1.2)</td>
<td>4455</td>
<td>[0.50, 1.12]</td>
</tr>
<tr>
<td>(0.8, 1.1)</td>
<td>4362</td>
<td>[0.48, 0.58]</td>
<td>(0.7, 1.3)</td>
<td>4521</td>
<td>[0.67, 1.28]</td>
</tr>
</tbody>
</table>

small LP relaxation gap suggests that the SDDiP with standard Benders’ cut is good enough to solve this large-scale MSUC instance.

7 Conclusion

In this paper, we propose a stagewise-decomposition algorithm based on SDDiP with various algorithmic enhancements to solve the MSUC problem. Extensive numerical experiments demonstrate that the proposed algorithm can successfully handle MSUC problems with a huge number of scenarios that were impossible before. It is also verified that Lagrangian cuts are indispensable in achieving exact solution and convergence. Our experiments show that when solving the Lagrangian relaxation of the stage problem, the Level Method performs superior to the standard subgradient method. We also observe that for the 118-bus system, it suffices to use SDDiP with only standard Benders’ cuts to obtain a good policy.

There are several interesting future research questions related to MSUC. In this paper, we decompose the 24-hour MSUC problem on an hourly base. An alternative is to consolidate several consecutive hours into one stage. Such a formulation increases the size of a stage problem but reduces the total number of decision stages. It would be interesting to investigate how such an aggregated model performs compares to the hourly based multistage model. Another direction is to study the MSUC problem under a risk-averse setting, as system reliability is of the utmost importance in practice.

Acknowledgement

This research has been supported in part by the National Science Foundation Grant 1633196.
References


