Linearizing Bilinear Products of Shadow Prices and Dispatch Variables in Bilevel Problems for Optimal Power System Planning

Nicholas D. Laws, *Student Member, IEEE,* and Grani A. Hanasusanto

***Abstract*—This work presents a general method for linearizing bilinear terms in the upper level of bilevel optimization problems when the bilinear terms are products of the primal and dual variables of the lower level. Bilinear terms of this form often appear in energy market optimization models where the dual variable represents the market price of energy and the primal variable represents a generator dispatch decision. Prior works have linearized such bilinear terms for specific problems. This work is the first to demonstrate how to linearize these terms in the most general case.**

***Index Terms*—Duality, Optimization methods, Power system economics, Power system planning.**

I. Introduction and Background

Since the restructuring of electricity markets began in the early 1980's [1] and the introduction of locational marginal pricing into large scale power markets in the 1990's researchers have been investigating electricity market design optimization problems. From a market participant point-of-view one of the most critical terms in a problem is the price signal (typically in \$/MWh) from the market operator multiplied by the energy delivered (MWh) by the participant, which together represent the participant's income. When both the price signal and energy delivered are decision variables in a mathematical program then the problem becomes bilinear.

In many electricity markets the price signal to market participants (or generators) is determined as the marginal price of the load balance constraint at any given network node at any given time step. The objective of the market model is to minimize the total cost of energy, or as it is commonly known: maximizing the social welfare. The constraints of the model typically represent an approximation to the power flow equations and take participant cost functions as input.

Equilibrium models allow modeling both the electricity market and participant behavior by including the power flow constraints and multiple, competing objective functions. Bilevel or Stackelberg Game formulations are common in electricity market models that include participant objectives, which are typically to maximize profits.

This work was funded in part by the National Renewable Energy Laboratory's Laboratory Directed Research and Development program.

N. D. Laws is with the National Renewable Energy Laboratory, Golden, CO, USA and the Walker Department of Mechanical Engineering, The University of Texas at Austin, TX, USA (e-mail: nick.laws@nrel.gov).

G. A. Hanasusanto is with the Walker Department of Mechanical Engineering, The University of Texas at Austin, TX, USA (e-mail: grani.hanasusanto@utexas.edu).

Manuscript received August 24, 2021.

Ruiz *et al*. 2009 [2] is the earliest known example to demonstrate that bilinear terms for market price and participant dispatch can be linearized. Their model places the market participant in the upper level, which chooses its offer curve for energy generation, while the lower level models the electricity market given the other participants' offer curves. Interestingly, Fernandez-Blanco *et al*. 2016 [3] claims to be the first to find a linearization for the same bilinear terms (products of lower level primal and dual variables) in the upper level of a bilevel program. Additional problem specific examples of the linearization technique can be found in [4] and [5].

This paper presents a generic algorithm for linearizing the bilinear terms of interest, starting with integer dispatch decisions in the lower level problem multiplied by shadow or market price decisions. Then, we show what conditions are required to linearize the same bilinear terms when the dispatch decisions are continuous. The linearization technique relies primarily on Strong Duality Theorem [6]. The technique works when the upper level and/or the lower level problems are non-linear in constraints or objectives. However, the lower level constraints that include the lower level variables from the upper level bilinear terms must be linear to get an exact linearization of the upper level bilinear terms.

II. Linearization Method with Integer Upper Level Variables

We begin by assuming that the upper level variables $\boldsymbol{x}$ are integer such that any product of integer $\boldsymbol{x}$ and the continuous variables $\boldsymbol{y}$ or $\boldsymbol{\lambda}$ can be made linear using binary expansion [7]. The bilievel problem with bilinear terms in the upper level objective and a linear lower level is

$$\min_{\boldsymbol{x}\in\mathcal{Z}^M,\boldsymbol{y}\in\mathcal{R}^N} \quad f(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{\lambda}^\intercal \boldsymbol{A}\boldsymbol{y} \tag{1a}$$

$$\text{s.t.} \quad g(\boldsymbol{x},\boldsymbol{y}) \leq 0 \tag{1b}$$

$$\boldsymbol{y} \in \arg\min_{\boldsymbol{y}\in\mathcal{R}^N} \quad \boldsymbol{c}^\intercal\boldsymbol{y} + \boldsymbol{x}^\intercal\boldsymbol{B}\boldsymbol{y} \tag{1c}$$

$$\text{s.t.} \quad \underline{\boldsymbol{y}} \leq \boldsymbol{y} \quad (\underline{\boldsymbol{\mu}}) \tag{1d}$$

$$\boldsymbol{y} \leq \overline{\boldsymbol{y}} \quad (\overline{\boldsymbol{\mu}}) \tag{1e}$$

$$\boldsymbol{U}\boldsymbol{x} + \boldsymbol{V}\boldsymbol{y} = \boldsymbol{w} \quad (\boldsymbol{\lambda}). \tag{1f}$$

Table I summarizes the terms in Equation 1. Note that the method is also valid for bilinear terms of $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ in the upper level constraints, but they are not shown for clarity.

TABLE I
SETS, INDICES, PARAMETERS, AND DECISION VARIABLES.

Decision Variables	
$\boldsymbol{x} \in \mathcal{R}^M$	upper level, primal decision variables
$\boldsymbol{y} \in \mathcal{R}^N$	lower level, primal decision variables
$\boldsymbol{\lambda} \in \mathcal{R}^J$	lower level, dual variables for equality constraints
$\overline{\boldsymbol{\mu}} \in \mathcal{R}_+^N$	lower level, non-negative, dual variables for upper bounds
$\underline{\boldsymbol{\mu}} \in \mathcal{R}_+^N$	lower level, non-negative, dual variables for lower bounds
Parameters	
$\boldsymbol{c} \in \mathcal{R}^N$	lower level cost coefficients for lower level decisions $\boldsymbol{y}$
$\boldsymbol{U} \in \mathcal{R}^{J \times M}$	lower level equality constraint coefficients for upper level decisions x
$\boldsymbol{V} \in \mathcal{R}^{J \times N}$	lower level equality constraint coefficients for lower level decisions $\boldsymbol{y}$
$\boldsymbol{w} \in \mathcal{R}^J$	lower level equality constraints right-hand-side
$\overline{\boldsymbol{y}} \in \mathcal{R}^N$	upper bounds for lower level, primal decision variables
$\underline{\boldsymbol{y}} \in \mathcal{R}^N$	lower bounds for lower level, primal decision variables
$\boldsymbol{A} \in \mathcal{R}^{J \times N}$	upper level coefficients for bilinear terms of lower level primal and dual variables
$\boldsymbol{B} \in \mathcal{R}^{M \times N}$	lower level coefficients for bilinear terms of lower level primal and upper level primal variables
Sets and Indices	
$\mathcal{A}$	$\{(j,n) \in \mathcal{J} \times \mathcal{N} : A_{jn} \neq 0\}$
$\mathcal{A}_\mathcal{J}$	$\{j \in \mathcal{J} : \exists n \in \mathcal{N}$ such that $A_{jn} \neq 0\}$
$\mathcal{A}_\mathcal{N}$	$\{n \in \mathcal{N} : \exists j \in \mathcal{J}$ such that $A_{jn} \neq 0\}$
$\mathcal{J}$	$1, 2, \dots, J$, $\lvert\mathcal{J}\rvert$ = number of lower level equality constraints
$\mathcal{J}_j \subseteq \mathcal{J}$	indices of lower level equality constraints connected to constraint j via non-zero values of $\boldsymbol{V}$, i.e. the constraints that share variables with constraint j and the constraints that share variables with those constraints (and so on recursively as described in Algorithm 1).
$\mathcal{J}_\cup$	$\bigcup_{j \in \mathcal{A}_\mathcal{J}} \mathcal{J}_j$
$\mathcal{M}$	$1, 2, \dots, M,\ \lvert\mathcal{M}\rvert$ = number of upper level variables
$\mathcal{N}$	$1, 2, \dots, N,\ \lvert\mathcal{N}\rvert$ = number of lower level variables
$\mathcal{N}_n \subseteq \mathcal{N}$	indices of lower level variables connected to variable y_n via non-zero values of $\boldsymbol{V}$
$\mathcal{N}_\cup$	$\bigcup_{n \in \mathcal{A}_\mathcal{N}} \mathcal{N}_n$
$\mathcal{AB}_\mathcal{N}$	$\{n \in \mathcal{A}_\mathcal{N} : \exists m \in \mathcal{M}$ such that $B_{mn} \neq 0\}$
$\mathcal{AB}$	$\{(j,n) \in \mathcal{A} : \exists m \in \mathcal{M}$ such that $B_{mn} \neq 0\}$
$\emptyset$	The empty set
$\mathcal{Z}$	The set of integers

To linearize any $\lambda_j y_n$ term one must combine the lower level primal and dual constraints. The first step is to multiply the lower level primal constraints (1f) by $\boldsymbol{\lambda}$ component-wise:

$$\boldsymbol{V}\boldsymbol{y} \circ \boldsymbol{\lambda} = \boldsymbol{w} \circ \boldsymbol{\lambda} - \boldsymbol{U}\boldsymbol{x} \circ \boldsymbol{\lambda} \tag{2}$$

where $\circ$ denotes the Hadamard product. Note that one can also multiply each of the primal constraints by each of the components of $\boldsymbol{\lambda}$ to get J^2 equations. However, in practice the bilinear terms that appear in the upper level problem are bilinear in y_n and λ_j, where λ_j is the Lagrange multiplier of the constraint that involves y_n.

Second, the dual constraints of $\boldsymbol{y}$ are multiplied with $\boldsymbol{y}$ as follows:

$$(\boldsymbol{V}^\intercal \boldsymbol{\lambda}) \circ \boldsymbol{y} = \boldsymbol{c} \circ \boldsymbol{y} + \overline{\boldsymbol{\mu}} \circ \boldsymbol{y} - \underline{\boldsymbol{\mu}} \circ \boldsymbol{y} + (\boldsymbol{B}^\intercal \boldsymbol{x}) \circ \boldsymbol{y}. \tag{3}$$

Note that any $\overline{\mu}_n y_n$ can be linearized because of the upper bound

$$y_n \leq \overline{y}_n. \tag{4}$$

The complementary slackness condition for (4) allows one to linearize $\overline{\mu}_n y_n$:

$$\overline{\mu}_n y_n = \overline{\mu}_n \overline{y}_n. \tag{5}$$

A similar result follows for any $\underline{\mu}_n y_n$. Combining the last result with the complementary slackness conditions gives:

$$(\boldsymbol{V}^\intercal \boldsymbol{\lambda}) \circ \boldsymbol{y} = \boldsymbol{c} \circ \boldsymbol{y} + \overline{\boldsymbol{\mu}} \circ \overline{\boldsymbol{y}} - \underline{\boldsymbol{\mu}} \circ \underline{\boldsymbol{y}} + (\boldsymbol{B}^\intercal \boldsymbol{x}) \circ \boldsymbol{y}. \tag{6}$$

Equations (2) and (6) are then combined to produce a system of equations with the bilinear products of $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ as the unknowns. In the following we show how to solve for a specific $\lambda_j V_{jn} y_n$.

Let the i^{th} row of (2) be defined as (P_i), which can be written:

$$(P_i) : \lambda_i V_{in} y_n = w_i \lambda_i - \lambda_i \sum_{k \in \mathcal{N} \setminus \{n\}} V_{ik} y_k - \lambda_i \sum_{m \in \mathcal{M}} U_{im} x_m \tag{7}$$

And, let the k^{th} row of (6) be defined as (D_k), which can be written:

$$(D_k) : y_k \sum_{i \in \mathcal{J}} V_{ik} \lambda_i = c_k y_k + \overline{\mu}_k \overline{y}_k - \underline{\mu}_k \underline{y}_k + y_k \sum_{m \in \mathcal{M}} B_{mk} x_m \tag{8}$$

Note that the choice of P for (P_i) and D for (D_k) are intentional: P is for *primal* constraints and D is for *dual* constraints.

Algorithm 1 outlines the procedure for determining the minimum set of the (P_i) and (D_k) equations needed to linearize a given $\lambda_j y_n$ term. Note that the algorithm refers to the indices of (P_i) as rows and (D_k) as columns because the sums over V_{jk} in (7) and (8) are over the rows and columns of $\boldsymbol{V}$ respectively.

The first step of Algorithm 1 is to check if V_{jn} is the only non-zero value in the n^{th} column of $\boldsymbol{V}$: in this case (D_n) provides the exact linearization of $\lambda_j y_n$ (and (P_i) is unnecessary):

$$y_n \lambda_j = \frac{1}{V_{jn}} \left(c_n y_n + \overline{\mu}_n \overline{y}_n - \underline{\mu}_n \underline{y}_n + y_n \sum_{m \in \mathcal{M}} B_{mn} x_m \right) \tag{9}$$

Note that (9) only applies under the condition that y_n is in a single lower level primal constraint.

In the second step of Algorithm 1 the first primal equation (P_j) is added to the set of row indices that will be returned at the end of the algorithm (where j is an input). Additionally, for all the non-zero values in the j^{th} row of $\boldsymbol{V}$, except V_{jn}, the indices of the dual equations (D_k) are added to the set of column indices. In mathematical terms, this step is taking (P_j):

$$\lambda_j V_{jn} y_n = w_j \lambda_j - \lambda_j \sum_{k \in \mathcal{N} \setminus \{n\}} V_{jk} y_k - \lambda_j \sum_{m \in \mathcal{M}} U_{jm} x_m \tag{10}$$

and all of the (D_k) equations for $k \in \mathcal{N} \setminus \{n\}$ in order to replace the bilinear terms of λ_j and y_k on the right-hand-side of (10). Each (D_k) equation can add more bilinear terms of $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ and so step three of Algorithm 1 adds additional equations if necessary.

In the third and final step of Algorithm 1 a recursive function, Algorithm 2, is used to search the array $\boldsymbol{V}$ for non-zero, "connected" values. We use the term "connected" to indicate that one could draw horizontal and vertical paths through $\boldsymbol{V}$ to connect non-zero entries to the first entry of interest V_{jn}, starting with a horizontal line each time. A horizontal line adds a (P_i) equation and a vertical line adds a (D_k) equation. The indices of the rows and columns are collected until a sufficient amount of equations are obtained to linearize the $\lambda_j y_n$ term in the upper level objective.

Note that Algorithm 2 is similar to - but not the same as - finding the blocks of a block-diagonal matrix. The difference is that Algorithm 2 does not necessarily find *all* of the non-zero values in a block. In other words, one does not need all of the (P_i) and (D_k) equations that may be available; one only needs as many equations as unknowns (where the unknowns are products of $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ entries). Algorithm 2 has some conditions under which it returns as error: these errors occur when the search has indicated that redundant row or column indices should be appended to the final vectors. Mathematically, these errors indicate that there are more unknowns than equations and thus the system of equations is underdetermined.

Algorithm 1: Minimum set of equations to linearize $\lambda_j y_n$

---

input : The 2D array V; and the integers (j, n) of non-zero V_{jn}.
output: Indices of (P_i) and (D_k) necessary to linearize a $\lambda_j y_n$ term.

1. **if** $V_{j'n} = 0 \; \forall j' \in \mathcal{J} \setminus \{j\}$ **then**
    - **return** {},$\{n\}$ (only need D_n)

end

2. Initialize arrays of integers for the rows and columns:
    - $\mathcal{J}_j$ = {j}
    - cols_to_check = $\{k \in \mathcal{N} \setminus \{n\} : V_{jk} \neq 0\}$
    - $\mathcal{N}_n$ = copy(cols_to_check)
3. Recursive search to find all connections

foreach k *in cols_to_check* **do**
- *rows*, *cols* = recursive_array_search(V, *j*, *k*, {}, {})
- $\mathcal{J}_j \leftarrow \mathcal{J}_j \cup$ *rows*
- $\mathcal{N}_n \leftarrow \mathcal{N}_n \cup$ *cols*

end
return $\mathcal{J}_j$, $\mathcal{N}_n$

Algorithm 2: recursive_array_search

---

input : The 2D array V; integers row j and column k; and two vectors of integers to append to: *rows* and *cols*.
output: Two vectors of integers for the non-zero entries of V connected to row j and column k.

rs = $\{ j' \in \mathcal{J} \setminus \{j\} : V_{ik} \neq 0 \}$
if *rs* $\cap$ *rows* $\neq \emptyset$ **then**
- **return** error: redundant row

end
rows $\leftarrow$ *rows* $\cup$ *rs*
foreach $r \in$ *rs* **do**
- *cs* = $\{ k' \in \mathcal{N} \setminus \{k\} : V_{rk'} \neq 0 \}$
- **if** $\{$*cs* $\cap$ *cols*$\} \neq \emptyset$ **then**
    - **return** error: redundant column
- **end**
- *cols* $\leftarrow$ *cols* $\cup$ *cs*
- **foreach** $c \in$ *cs* **do**
    - recursive_array_search(V, *r*, *c*, *rows*, *cols*)
- **end**

end
return *rows*, *cols*

Let the indices of (P_i) and (D_k) returned from Algorithm 1 for a given $(j, n) \in \mathcal{A}$ pair be defined as $\mathcal{J}_j$ and $\mathcal{N}_n$ respectively. The exact linearization of $\lambda_j y_n$ is:

$$\begin{aligned} \lambda_j y_n = \frac{1}{V_{jn}} \Bigg[& \sum_{j' \in \mathcal{J}_j} \Big(w_{j'} \lambda_{j'} - \lambda_{j'} \sum_{m \in \mathcal{M}} U_{j'm} x_m \Big) \\ & - \sum_{n' \in \mathcal{N}_n} \Big(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \\ & + y_{n'} \sum_{m \in \mathcal{M}} B_{mn'} x_m \Big) \Bigg], \end{aligned} \tag{11}$$

which is simply a combination of (7) and (8) for all of the non-zero values of $\boldsymbol{V}$ connected to $\lambda_j y_n$, as demonstrated with the following examples.

Example 1. The simplest case for linearizing a certain $\lambda_j y_n$ term occurs when y_n is in only one lower level constraint, which is when step 1 of Algorithm 1 returns and (9) provides the solution. Note that (9) is a particular instance of (8). In this example we present the *next* simplest case, which is when y_n is in more than one constraint but the other $\boldsymbol{y}$ variables in constraint j are in no other lower level constraints.

In Step 2 of Algorithm 1 the indices of (D_k) in the set $\{k \in \mathcal{N} \setminus \{n\} : V_{jk} \neq 0\}$ are added to the set of column indices to check using the recursive Algorithm 2. And the set $\mathcal{J}_j$ is initialized with $\{j\}$. Algorithm 2 then checks for non-zero values of $\boldsymbol{V}$ above and below each V_{jk} entry for each of the column indices. If no non-zero values are found then Algorithm 2 returns the same sets that were passed to it, meaning that no more equation indices are needed to linearize the $\lambda_j y_n$ term.

Take one particular k' in $\mathcal{N} \setminus \{n\}$ for example. The special case that $V_{ik'} = 0 \ \forall i \in \mathcal{J} \setminus \{j\}$ is illustrated as follows:

$$\boldsymbol{V} = \begin{array}{c} \\ \\ \\ j\text{-th row} \\ \\ \\ \\ \end{array} \begin{bmatrix} & & 0 & \\ & & \vdots & \\ & & 0 & \\ \dots V_{jn} \dots & & V_{jk'} & \dots \\ & & 0 & \\ & & \vdots & \\ & & 0 & \end{bmatrix} \quad (k'\text{-th col.}). \tag{12}$$

When $y_{k'}$ is only in constraint j then $(D_{k'})$ is:

$$\lambda_j V_{jk'} y_{k'} = c_{k'} y_{k'} + \overline{\mu}_{k'} \overline{y}_{k'} - \underline{\mu}_{k'} \underline{y}_{k'} + y_{k'} \sum_{m \in \mathcal{M}} B_{mk'} x_m. \tag{13}$$

Equation (13) can then be substituted into (P_j), shown in (10), to eliminate the bilinear term of λ_j and $y_{k'}$ in the sum over $k \in \mathcal{N} \setminus \{n\}$. A similar result follows for eliminating all of the $\lambda_j y_k$ terms on the right hand side of (10). ■

Example 2. Continuing from our previous example, now let us assume that the k'-th column of $\boldsymbol{V}$ has one other non-zero entry. Now, additional combinations of the (P_i) and (D_k) equations are necessary to eliminate the $\lambda_j y_{k'}$ term in (10). This is where step three of the Algorithm comes in.

For this example let $V_{i'k'} \neq 0$ for a particular i' in $\mathcal{J} \setminus \{j\}$, and let $V_{ik'} = 0 \ \forall i \in \mathcal{J} \setminus \{j, i'\}$. Also, let the i'-th row of $\boldsymbol{V}$ contain one other non-zero value $V_{i'\ell}$, and let $V_{i\ell} = 0 \ \forall i \in \mathcal{J} \setminus \{i'\}$. This case is illustrated in (14).

$$V = \begin{array}{r} \\ \\ \\ j\text{-th row} \\ i'\text{-th row} \\ \\ \\ \\ \end{array} \begin{bmatrix} & & 0 & 0 & & \\ & & \vdots & \vdots & & \\ & & \vdots & 0 & & \\ & \dots V_{jn} & 0 & V_{jk'} & \dots & \\ 0 \dots & 0 & V_{i'\ell} & V_{i'k'} & 0 & \dots 0 \\ & & 0 & 0 & & \\ & & \vdots & \vdots & & \\ & & 0 & 0 & & \end{bmatrix} \quad (\ell\text{-th col.}, \ k'\text{-th col.}) \tag{14}$$

Now $(D_{k'})$ gives:

$$\begin{aligned} \lambda_j V_{jk'} y_{k'} + \lambda_{i'} V_{i'k'} y_{k'} &= c_{k'} y_{k'} + \overline{\mu}_{k'} \overline{y}_{k'} - \underline{\mu}_{k'} \underline{y}_{k'} \\ &+ y_{k'} \sum_{m \in \mathcal{M}} B_{mk'} x_m. \end{aligned} \tag{15}$$

Since the i'-th row of $\boldsymbol{V}$ contains only one other non-zero value $V_{i'\ell}$, and the other values in column ℓ of $\boldsymbol{V}$ are zero as illustrated in (14), then adding equations $(P_{i'})$ and (D_ℓ) allows one to linearize the $\lambda_{i'} V_{i'k'} y_{k'}$ term in (15) in a similar fashion to the previous example. ■

These examples demonstrate that the task of linearizing any $\lambda_j y_n$ term is to find the minimum set of (P_i) and (D_k) equations in order to eliminate the other bilinear terms of $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ entries.

Finally, using the result (11) the mixed integer linear form of (1) is:

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}, \overline{\boldsymbol{\mu}}, \underline{\boldsymbol{\mu}}} \ & f(\boldsymbol{x}, \boldsymbol{y}) \\ & + \sum_{(j,n) \in \mathcal{A}} \frac{A_{jn}}{V_{jn}} \Bigg[\sum_{j' \in \mathcal{J}_j} \Big(w_{j'} \lambda_{j'} - \lambda_{j'} \sum_{m \in \mathcal{M}} U_{j'm} x_m \Big) \\ & - \sum_{n' \in \mathcal{N}_n} \Big(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \\ & + y_{n'} \sum_{m \in \mathcal{M}} B_{mn'} x_m \Big) \Bigg] \end{aligned} \tag{16a}$$

$$\text{s.t. } g(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \tag{16b}$$
$$\boldsymbol{c} + \boldsymbol{B}^\intercal \boldsymbol{x} + \boldsymbol{V}^\intercal \boldsymbol{\lambda} + \overline{\boldsymbol{\mu}} - \underline{\boldsymbol{\mu}} = \boldsymbol{0} \tag{16c}$$
$$\underline{\boldsymbol{y}} \leq \boldsymbol{y} \tag{16d}$$
$$\boldsymbol{y} \leq \overline{\boldsymbol{y}} \tag{16e}$$
$$\boldsymbol{U}\boldsymbol{x} + \boldsymbol{V}\boldsymbol{y} = \boldsymbol{w} \tag{16f}$$
$$\overline{\boldsymbol{\mu}} \perp (\boldsymbol{y} - \overline{\boldsymbol{y}}) \tag{16g}$$
$$\underline{\boldsymbol{\mu}} \perp (\underline{\boldsymbol{y}} - \boldsymbol{y}) \tag{16h}$$

where the lower level problem has been replaced with the Karush Kuhn Tucker (KKT) conditions and the complementary constraints can be modeled as special order sets or using the "big M" method from [8].

III. Linearization Method with Continuous Upper Level Variables

In Section II we assumed that $\boldsymbol{x}$ are integer such that all of the products of x_m and y_n or products of x_m and λ_j can be linearized using binary expansion [7]. Here we show the conditions under which a $\lambda_j y_n$ term can be linearized when the upper level variables $\boldsymbol{x}$ are continuous.

The conditions are divided into two groups with one group less restrictive than the other. The first group of conditions is less restrictive but does not allow lower level variables $\boldsymbol{y}$ to be bilinear in both the upper level problem with $\boldsymbol{\lambda}$ and the lower objective with $\boldsymbol{x}$. In mathematical terms this is when $\mathcal{AB} = \emptyset$, where $\mathcal{AB} \triangleq \{(j,n) \in \mathcal{A} : \exists m \in \mathcal{M} \text{ such that } B_{mn} \neq 0\}$.

The second group of conditions allows a problem to be linearized when bilinear products of $\lambda_j y_n$ are in the upper level and bilinear products of $x_m y_n$ are in the lower level objective for a given n. These types of problems are particularly relevant to energy system market models as is demonstrated with examples.

A. Conditions when $\mathcal{AB} = \emptyset$

Recall that the Algorithms 1 and 2 provide the sets $\mathcal{J}_j$ for each λ_j in the upper level objective. Let $\mathcal{J}_\cup \triangleq \bigcup_{j \in \mathcal{A}_\mathcal{J}} \mathcal{J}_j$, which includes the indices of all the lower level constraints that are connected (via non-zero values of $\boldsymbol{V}$) to the λ_j terms in the upper level objective. Therefore, in order to eliminate all bilinear terms of the form $\lambda_j U_{jm} x_m$ in (16a) the following condition must be met:

Condition 1. $U_{jm} = 0 \; \forall j \in \mathcal{J}_\cup, \; \forall m \in \mathcal{M}$

Similar to Condition 1, let $\mathcal{N}_\cup \triangleq \bigcup_{n \in \mathcal{A}_\mathcal{N}} \mathcal{N}_n$, then one could assume that

Condition 2. $B_{mn} = 0 \; \forall m \in \mathcal{M}, \; \forall n \in \mathcal{N}_\cup$

to eliminate all bilinear terms of the form $x_m B_{mn} y_n$ from (16a). Under Conditions 1 and 2 the mixed integer result for (1) is

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda},\overline{\boldsymbol{\mu}},\underline{\boldsymbol{\mu}}} f(\boldsymbol{x},\boldsymbol{y}) + \sum_{(j,n)\in\mathcal{A}} \frac{A_{jn}}{V_{jn}} \left[\sum_{j'\in\mathcal{J}_j} (w_{j'}\lambda_{j'}) - \sum_{n'\in\mathcal{N}_n} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \right) \right] \tag{17a}$$

$$\text{s.t. } g(\boldsymbol{x},\boldsymbol{y}) \le 0, \tag{17b}$$

$$\boldsymbol{c} + \boldsymbol{B}^\mathsf{T}\boldsymbol{x} + \boldsymbol{V}^\mathsf{T}\boldsymbol{\lambda} + \overline{\boldsymbol{\mu}} - \underline{\boldsymbol{\mu}} = \boldsymbol{0} \tag{17c}$$

$$\underline{\boldsymbol{y}} \le \boldsymbol{y} \tag{17d}$$

$$\boldsymbol{y} \le \overline{\boldsymbol{y}} \tag{17e}$$

$$\boldsymbol{U}\boldsymbol{x} + \boldsymbol{V}\boldsymbol{y} = \boldsymbol{w} \tag{17f}$$

$$\overline{\boldsymbol{\mu}} \perp (\boldsymbol{y} - \overline{\boldsymbol{y}}) \tag{17g}$$

$$\underline{\boldsymbol{\mu}} \perp (\underline{\boldsymbol{y}} - \boldsymbol{y}) \tag{17h}$$

B. Conditions when $\mathcal{AB} \neq \emptyset$

The case when Condition 2 is violated and Problem (1) has bilinear terms in the upper and lower level objectives of the form $\lambda_j A_{jn} y_n$ and $x_m B_{mn} y_n$, (where $A_{jn} \neq 0$ and $B_{mn} \neq 0$), for some n is particularly relevant to energy system market models. For example, take the case where $A_{jn} = 1$ and $B_{mn} = -1$ for some particular m, j, and n. Let y_n represent a lower level generator dispatch decision. Then λ_j represents the marginal cost of the dispatch decision y_n as well as the upper level's cost of purchasing power from the lower level. And $-x_m y_n$ in the lower level objective is the lower level's income for the generation y_n using the price signal x_m. Appendix A provides a simple example of such a scenario. Thus it is useful to investigate the linearization of problems when $\mathcal{AB} \neq \emptyset$ (i.e. when Condition 2 is violated).

The problem of interest has the following structure:

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) + \sum_{(j,n)\in\mathcal{A}} \lambda_j A_{jn} y_n \tag{18a}$$

$$\text{s.t. } g(\boldsymbol{x},\boldsymbol{y}) \le 0 \tag{18b}$$

$$\boldsymbol{y} \in \arg\min_{\boldsymbol{y}} \; \boldsymbol{c}^\mathsf{T}\boldsymbol{y} + \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{A}_\mathcal{N}} x_m B_{mn} y_n + \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}\setminus(\mathcal{A}_\mathcal{N}\cup\mathcal{N}_\cup)} x_m B_{mn} y_n \tag{18c}$$

$$\text{s.t.} \quad \underline{\boldsymbol{y}} \le \boldsymbol{y} \quad (\underline{\boldsymbol{\mu}}) \tag{18d}$$

$$\boldsymbol{y} \le \overline{\boldsymbol{y}} \quad (\overline{\boldsymbol{\mu}}) \tag{18e}$$

$$\sum_{n\in\mathcal{N}} V_{jn} y_n = w_j \quad (\lambda_j), \quad \forall j \in \mathcal{J}_\cup \tag{18f}$$

$$\sum_{m\in\mathcal{M}} U_{jm} x_m + \sum_{n\in\mathcal{N}} V_{jn} y_n = w_j, \quad \forall j \in \mathcal{J} \setminus \mathcal{J}_\cup. \tag{18g}$$

Note that the products of $\boldsymbol{x}$ and $\boldsymbol{y}$ in the lower level objective (18c) are linearized when the lower level problem is replaced with the KKT conditions. And the set of y_n for all $n \in \mathcal{N} \setminus (\mathcal{A}_\mathcal{N} \cup \mathcal{N}_\cup)$ in the last sum of (18c) are the values of $\boldsymbol{y}$ that are *not* in the upper level objective nor connected to the $y_n, n \in \mathcal{A}_\mathcal{N}$, in the upper level objective. Recall that the connected indices are provided by Algorithm 1 and captured in $\mathcal{N}_\cup$. We will show shortly that the connected y_n values must not be in the lower level objective with $\boldsymbol{x}$ terms to prevent $y_n x_m$ terms from showing up in the (D_k) equations needed to linearize the $\lambda_j y_n$ in the upper level objective. Also, Condition 1 is reflected in (18f).

Now, applying Condition 1 to (11) gives

$$\lambda_j y_n = \frac{1}{V_{jn}} \left[\sum_{j'\in\mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n'\in\mathcal{N}_n} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} + y_{n'} \sum_{m\in\mathcal{M}} B_{mn'} x_m \right) \right], \; \forall (j,n) \in \mathcal{A}. \tag{19}$$

We wish to eliminate the $y_{n'} B_{mn'} x_m$ terms when $\mathcal{AB} \neq \emptyset$. Recall that the $y_{n'} B_{mn'} x_m$ terms in (11) and (19) come from the (D_k) equations with $B_{mk} \neq 0$, and that the (D_k) equations for all $k \in \mathcal{N}_\cup$ are necessary to linearize the upper level $\lambda_j A_{jk} y_k$ terms.

Let us assume that a less restrictive version of Condition 2 holds:

Condition 2′. $B_{mn} = 0 \; \forall m \in \mathcal{M}, \; \forall n \in \mathcal{N}_\cup \setminus \mathcal{A}_\mathcal{N}$

Condition 2′ implies that none of the lower level variables *connected* to the $\lambda_j A_{jn} y_n$ terms (provided by Algorithm 1) are in the lower level objective with $y_n B_{mn} x_m$ terms, except the lower level variables in the upper level objective ($y_n \; \forall n \in$

$\mathcal{A}_{\mathcal{N}}$). Applying Condition 2$'$ to (19) gives:

$$\lambda_j y_n = \frac{1}{V_{jn}} \Bigg[\sum_{j' \in \mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n' \in \mathcal{N}_n \setminus \mathcal{A}_{\mathcal{N}}} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \right) - \sum_{n' \in \mathcal{N}_n \cap \mathcal{A}_{\mathcal{N}}} \bigg(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} + y_{n'} \sum_{m \in \mathcal{M}} B_{mn'} x_m \bigg) \Bigg], \ \forall (j,n) \in \mathcal{A}. \quad (20)$$

Let us also assume that Condition 3 holds:

Condition 3. $\mathcal{A}_{\mathcal{N}} \setminus \{n\} \subseteq \mathcal{N}_n \ \forall n \in \mathcal{A}_{\mathcal{N}}$

$$\Rightarrow \mathcal{N}_n \cap \mathcal{A}_{\mathcal{N}} = \mathcal{A}_{\mathcal{N}} \setminus \{n\} \ \forall n \in \mathcal{A}_{\mathcal{N}}$$

Condition 3 implies that the $y_n \ \forall n \in \mathcal{A}_{\mathcal{N}}$ variables of interest are connected to each other via non-zero values of $\boldsymbol{V}$. Appendix B provides an example problem, in which the $y_n \ \forall n \in \mathcal{A}_{\mathcal{N}}$ are indexed on time and connected to each other via another time-indexed variable in each equality constraint that is restricted to be no more than a certain value across all time.

Condition 3 allow us to rewrite (20) as

$$\lambda_j y_n = \frac{1}{V_{jn}} \Bigg[\sum_{j' \in \mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n' \in \mathcal{N}_n \setminus \mathcal{A}_{\mathcal{N}}} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \right) - \sum_{n' \in \mathcal{A}_{\mathcal{N}} \setminus \{n\}} \bigg(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} + y_{n'} \sum_{m \in \mathcal{M}} B_{mn'} x_m \bigg) \Bigg], \ \forall (j,n) \in \mathcal{A}. \quad (21)$$

Applying (8) to the last summation in (21) gives:

$$\lambda_j y_n = \frac{1}{V_{jn}} \Bigg[\sum_{j' \in \mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n' \in \mathcal{N}_n \setminus \mathcal{A}_{\mathcal{N}}} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \right) - \sum_{n' \in \mathcal{A}_{\mathcal{N}} \setminus \{n\}} \bigg(y_{n'} \sum_{j' \in \mathcal{J}} V_{j'n'} \lambda_{j'} \bigg) \Bigg], \ \forall (j,n) \in \mathcal{A}, \quad (22)$$

This step is key to eliminating the bilinear $y_{n'} B_{mn'} x_m$ terms when $\mathcal{AB} \neq \emptyset$. The next steps are to impose conditions that allow us to move the last summation in (22) to the left hand side to get a single sum of terms over the set $\mathcal{A}$.

Let us assume that Condition 4 holds:

Condition 4. $V_{j'n} = 0 \ \forall j' \in \mathcal{J} \setminus \{j\}, \ \forall (j,n) \in \mathcal{A}$.

Condition 4 is equivalent to each y_n for all $n \in \mathcal{A}_{\mathcal{N}}$ being in only one lower level constraint. Condition 4 implies that

$$y_k \sum_{j' \in \mathcal{J}} V_{j'k} \lambda_{j'} = \lambda_j V_{jk} y_k, \ \forall (j,k) \in \mathcal{A} \quad (23)$$

Note that Condition 4 requires that Step 1 of the Algorithm be skipped. The revised algorithm for Conditions 1, 2$'$, 3, and 4 is shown in Algorithm 3.

Algorithm 3: Minimum set of equations to linearize $\lambda_j y_n$ under Conditions 1, 2$'$, 3, and 4

input : The 2D array V; and the integers (j, n) of non-zero V_{jn}.
output: Indices of (P_i) and (D_k) necessary to linearize a $\lambda_j y_n$ term.

1. Initialize arrays of integers for the rows and columns:
   $\mathcal{J}_j$ = $\{j\}$
   cols_to_check = $\{k \in \mathcal{N} \setminus \{n\} : V_{jk} \neq 0\}$
   $\mathcal{N}_n$ = copy(cols_to_check)
2. Recursive search to find all connections

foreach k *in cols_to_check* **do**
    rows, *cols* = recursive_array_search(V, *j*, *k*, {}, {})
    $\mathcal{J}_j \leftarrow \mathcal{J}_j \cup$ *rows*
    $\mathcal{N}_n \leftarrow \mathcal{N}_n \cup$ *cols*
end
return $\mathcal{J}_j$, $\mathcal{N}_n$

Condition 4 allows us to write (22) as

$$\lambda_j y_n = \frac{1}{V_{jn}} \Bigg[\sum_{j' \in \mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n' \in \mathcal{N}_n \setminus \mathcal{A}_{\mathcal{N}}} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \right) - \sum_{(j',n') \in \mathcal{A} \setminus \{(j,n)\}} \left(y_{n'} V_{j'n'} \lambda_{j'} \right) \Bigg], \ \forall (j,n) \in \mathcal{A}, \quad (24)$$

Rearranging (24) gives:

$$\sum_{(j',n') \in \mathcal{A}} \lambda_{j'} V_{j'n'} y_{n'} = \sum_{j' \in \mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n' \in \mathcal{N}_n \setminus \mathcal{A}_{\mathcal{N}}} \left(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \right), \ \forall (j,n) \in \mathcal{A}. \quad (25)$$

Note that since (25) is valid for all $(j,n) \in \mathcal{A}$ it implies that $\mathcal{N}_n$ are equal for all $n \in \mathcal{A}_{\mathcal{N}}$ and that $\mathcal{J}_j$ are equal for all $j \in \mathcal{A}_{\mathcal{J}}$.

Lastly, we see that to replace $\sum_{(j,n) \in \mathcal{A}} \lambda_j A_{jn} y_n$ with the last result for $\sum_{(j,n) \in \mathcal{A}} \lambda_j V_{jn} y_n$ we must require that the two sums are equal to a proportionality constant p:

Condition 5. $A_{jn} = pV_{jn} \ \forall (j,n) \in \mathcal{A}$

$$\Rightarrow \sum_{(j,n)\in\mathcal{A}} \lambda_j A_{jn} y_n = p \sum_{(j,n)\in\mathcal{A}} \lambda_j V_{jn} y_n.$$

(In the Examples of Appendices (A) and (B) $p = -1$.)

With Condition 5 we can write (25) as:

$$\sum_{(j,n)\in\mathcal{A}} \lambda_j A_{jn} y_n = p \Bigg[\sum_{j'\in\mathcal{J}_j} w_{j'} \lambda_{j'} - \sum_{n'\in\mathcal{N}_n\setminus\mathcal{A}_\mathcal{N}} \Big(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \Big) \Bigg] \ \forall (j,n) \in \mathcal{A}. \tag{26}$$

Substituting (26) into (1) and replacing the lower level with the KKT conditions, under Conditions 1, 2′, 3, 4 and 5 the mixed integer result is

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda},\overline{\boldsymbol{\mu}},\underline{\boldsymbol{\mu}}} f(\boldsymbol{x},\boldsymbol{y}) + p \Bigg[\sum_{j'\in\mathcal{J}_j} (w_{j'} \lambda_{j'}) - \sum_{n'\in\mathcal{N}_n\setminus\mathcal{A}_\mathcal{N}} \Big(c_{n'} y_{n'} + \overline{\mu}_{n'} \overline{y}_{n'} - \underline{\mu}_{n'} \underline{y}_{n'} \Big) \Bigg] \tag{27a}$$

$$\text{s.t. } g(\boldsymbol{x},\boldsymbol{y}) \leq 0, \tag{27b}$$

$$\boldsymbol{c} + \boldsymbol{B}^\intercal \boldsymbol{x} + \boldsymbol{V}^\intercal \boldsymbol{\lambda} + \overline{\boldsymbol{\mu}} - \underline{\boldsymbol{\mu}} = \boldsymbol{0} \tag{27c}$$

$$\underline{\boldsymbol{y}} \leq \boldsymbol{y} \tag{27d}$$

$$\boldsymbol{y} \leq \overline{\boldsymbol{y}} \tag{27e}$$

$$\boldsymbol{U}\boldsymbol{x} + \boldsymbol{V}\boldsymbol{y} = \boldsymbol{w} \tag{27f}$$

$$\overline{\boldsymbol{\mu}} \perp (\boldsymbol{y} - \overline{\boldsymbol{y}}) \tag{27g}$$

$$\underline{\boldsymbol{\mu}} \perp (\underline{\boldsymbol{y}} - \boldsymbol{y}) \tag{27h}$$

Note that any $(j,n) \in \mathcal{A}$ can be used in (27a) to define the sets $\mathcal{J}_j$ and $\mathcal{N}_n$.

To summarize all of the conditions under which (27) is valid:

- Condition 1: $U_{jm} = 0 \ \forall j \in \mathcal{J}_\cup, \ \forall m \in \mathcal{M}$
  - None of the connected constraints contain $\boldsymbol{x}$ terms.
- Condition 2′: $B_{mn} = 0 \ \forall m \in \mathcal{M}, \ \forall n \in \mathcal{N}_\cup \setminus \mathcal{A}_\mathcal{N}$
  - None of the connected variables are multiplied with $\boldsymbol{x}$ in the lower level objective, except the $\boldsymbol{y}$ in the upper level objective that are multiplied with $\boldsymbol{\lambda}$.
- Condition 3: $\mathcal{A}_\mathcal{N} \setminus \{n\} \subseteq \mathcal{N}_n \ \forall n \in \mathcal{A}_\mathcal{N}$
  - Each of the y_n in the upper level objective are connected to each other via non-zero values of $\boldsymbol{V}$.
- Condition 4 $V_{j'n} = 0 \ \forall j' \in \mathcal{J} \setminus \{j\}, \ \forall j \in \mathcal{A}_\mathcal{J}$
  - Each of the y_n in the upper level objective are in only one lower level equality constraint.
- Condition 5 $A_{jn} = pV_{jn} \ \forall (j,n) \in \mathcal{A}$
  - All of the coefficients of the upper level $\lambda_j y_n$ terms are proportional to the corresponding coefficients in the lower level constraints to the same constant p.

The Examples in Appendices (A) and (B) both meet the Conditions 1, 2′, 3, 4 and 5. The theme in both problems is an energy market model with a load balance constraint in the lower level, bilinear products in the upper level objective of lower level dispatch variables and the load balance dual variables, and bilinear products in the lower level objective of upper level price signal variables and the same lower level dispatch variables as in the upper level objective.

In Appendix (C) we present a solution time comparison for the problem in Appendix (B) solved in the linearized form and the bilinear form. The linear problem solves in less than four seconds with a tolerance of 0.01%. The bilinear problem takes 380 seconds to get to a less than 2% gap and never gets below a 1% gap. See Appendix (C) for more details.

IV. Conclusion

This work presents a general method for linearizing bilinear products of lower level primal and dual variables in the upper level of bilevel optimization problems. The linearization method is especially relevant for modeling large scale energy distribution systems with many stakeholders and is therefore applicable to a growing number of problems as energy markets expand and adapt to new regulations such as FERC Order 2222 [9] and the increasing adoption of distributed energy resources [10]. By publishing this method we hope that more use cases will be discovered for the linearization technique.

For future work we intend to include the the method in an open source mathematical programming package such as BilevelJuMP.jl [11], [12]. We also plan on using the technique to convert large scale bilinear bilevel problems to mixed integer linear programs for studying compensation mechanisms of distributed energy resources serving as power system upgrade deferrals (c.f. [13]). Another future research direction involves using the optimal price signals from one level to the other as a transactive control mechanism.

Appendix A
Simple Energy Market Example

We present a simple example to demonstrate the linearization method. The example is also used to show the impact on solution times when modeling the original bilinear problem versus the linearized problem in Appendix C. We start without a time index for simplicity and then add a time index in Appendix B to create a larger problem for comparing solution times. The example with a time index will also allow us to demonstrate how to handle the case when $\mathcal{AB} \neq \emptyset$.

$$\min_{\boldsymbol{x}} \; x_0 q_0 + \lambda y_e \tag{28a}$$
$$\text{s.t. } x_0 \geq 0, \;\; x_e \geq 0 \tag{28b}$$
$$x_0 \geq x_i \tag{28c}$$
$$x_i - y_i + y_e = d \tag{28d}$$
$$y_e \perp y_i \tag{28e}$$
$$\boldsymbol{y} \in \arg\min_{\boldsymbol{y}} \;\; c_i y_i - x_e y_e + c_g y_g \tag{28f}$$
$$\text{s.t.} \qquad \boldsymbol{y} \geq \boldsymbol{0} \quad (\underline{\boldsymbol{\mu}}) \tag{28g}$$
$$\boldsymbol{y} \leq \overline{\boldsymbol{y}} \quad (\overline{\boldsymbol{\mu}}) \tag{28h}$$
$$-y_e + y_i + y_g = w \quad (\lambda) \tag{28i}$$

In (28) the upper level represents a power system operator that can import positive energy x_0 from the wholesale market at the price q_0. The net power imported from the wider grid is x_i. The system operator is not compensated for exported power, or negative values of x_i. The operator can also purchase power from node 1 for the marginal cost of energy λ determined by the load balance at node 1 (28i). The upper level load balance is given in (28d), where a second node is represented by the given demand d. The upper level also constrains node 1 from importing and exporting energy simultaneously (28e).

The lower level represents the decisions at node 1, which are how to meet the demand w for the least cost. The lower level can purchase retail energy y_i at the cost of c_i and/or purchase generation capacity y_g at cost c_g to meet the load. Node 1 can also choose to export excess energy production via y_e, which is compensated by x_e (chosen by the upper level).

Let $\boldsymbol{y} \triangleq [y_e, y_i, y_g]^\intercal$, thus y_e is in the first index of $\boldsymbol{y}$. Also note that the lower level has just one equality constraint, making $\mathcal{J} = \{1\}$. Using Algorithm 1 for Problem (28) yields $\mathcal{J}_1 = \{1\}$ and $\mathcal{N}_1 = \{2, 3\}$ (where the indices in $\mathcal{N}_1$ correspond with the indices in $\boldsymbol{y} = [y_e, y_i, y_g]^\intercal$). From (11):

$$\begin{aligned} \lambda_e(-1)y_e &= w\lambda - \left(c_g y_g + \overline{y}_g \overline{\mu}_g + c_i y_i + \overline{y}_i \overline{\mu}_i\right) \\ \Rightarrow \lambda y_e &= c_g y_g + \overline{y}_g \overline{\mu}_g + c_i y_i + \overline{y}_i \overline{\mu}_i - w\lambda. \end{aligned} \tag{29}$$

Substituting (29) into (28) and replacing the lower level with the KKT conditions yields the mixed integer-linear problem:

$$\min_{\boldsymbol{x}} \; x_0 q_0 + c_g y_g + \overline{y}_g \overline{\mu}_g + c_i y_i + \overline{y}_i \overline{\mu}_i - w\lambda \tag{30a}$$
$$\text{s.t. } x_0 \geq 0, \;\; x_e \geq 0 \tag{30b}$$
$$x_0 \geq x_i \tag{30c}$$
$$x_i - y_i + y_e = d \tag{30d}$$
$$y_e \perp y_i \tag{30e}$$
$$y_e \perp \overline{\mu}_e, \;\; y_i \perp \overline{\mu}_i \;\; y_g \perp \overline{\mu}_g \tag{30f}$$
$$y_e \perp \underline{\mu}_e, \;\; y_i \perp \underline{\mu}_i \;\; y_g \perp \underline{\mu}_g \tag{30g}$$
$$-y_e + y_i + y_g = w \tag{30h}$$
$$-x_e + \lambda + \overline{\mu}_e - \underline{\mu}_e = 0 \tag{30i}$$
$$c_i - \lambda + \overline{\mu}_i - \underline{\mu}_i = 0 \tag{30j}$$
$$c_g - \lambda + \overline{\mu}_g - \underline{\mu}_g = 0 \tag{30k}$$

where the complementary constraints can be modeled as special order sets or using the "big M" method from [8].

Appendix B
Energy Market Example that meets Conditions 1, 2′, 3, and 4

Let $\boldsymbol{y} \triangleq [y_{e,1}, \ldots, y_{e,\mathcal{T}}, \; y_{i,1}, \ldots, y_{i,\mathcal{T}}, \; y_{p,1}, \ldots, y_{p,\mathcal{T}}, \; y_g]^\intercal$ collect the lower level decision variables, where the second index represents integer time steps from $\{1, \ldots, T\} \triangleq \mathcal{T}$. Similarly, let $\boldsymbol{x} \triangleq [x_{0,1}, \ldots, x_{0,\mathcal{T}}, \; x_{i,1}, \ldots, x_{i,\mathcal{T}}, \; x_{e,1}, \ldots, x_{e,\mathcal{T}}]^\intercal$ collect the upper level decision variables. For the time-indexed problem we add an additional lower level decision $y_{p,t}$, which represents how much energy is produced by the generator with capacity y_g in time step t.

$$\min_{\boldsymbol{x}} \; \sum_{t \in \mathcal{T}} (x_{0,t} q_{0,t} + \lambda_t y_{e,t}) \tag{31a}$$
$$\text{s.t. } x_{0,t} \geq 0, \;\; x_{e,t} \geq 0, \qquad \forall t \in \mathcal{T} \tag{31b}$$
$$x_{0,t} \geq x_{i,t}, \qquad \forall t \in \mathcal{T} \tag{31c}$$
$$x_{i,t} - y_{i,t} + y_{e,t} = d_t, \qquad \forall t \in \mathcal{T} \tag{31d}$$
$$y_{e,t} \perp y_{i,t}, \qquad \forall t \in \mathcal{T} \tag{31e}$$
$$\boldsymbol{y} \in \arg\min_{\boldsymbol{y}} \; \sum_{t \in \mathcal{T}} (c_{i,t} y_{i,t} - x_{e,t} y_{e,t}) + c_g y_g \tag{31f}$$
$$\text{s.t.} \qquad \boldsymbol{y} \geq \boldsymbol{0} \quad (\underline{\boldsymbol{\mu}}) \tag{31g}$$
$$\boldsymbol{y} \leq \overline{\boldsymbol{y}} \quad (\overline{\boldsymbol{\mu}}) \tag{31h}$$
$$-y_{e,t} + y_{i,t} + y_{p,t} = w_t \quad (\lambda_t), \quad \forall t \in \mathcal{T} \tag{31i}$$
$$y_{p,t} \leq h_t y_g \quad (\mu_{h,t}), \qquad \forall t \in \mathcal{T} \tag{31j}$$

Problem (31) shows the example with a time index. The major change from (28) is that when $y_g > 0$ the lower level can produce energy $y_{p,t}$ based on a given, time-varying production factor $0 \leq h_t \leq 1$ (31j).

Problem (31) has a special structure that allows eliminating the $x_{e,t} y_{e,t}$ that come from th

First, we convert Problem (31) to standard form by adding the slack variable $y_{s,t}$ to the inequality (31j), as well as rename $\mu_{h,t}$ to $\lambda_{h,t}$:

$$(31j) \rightarrow y_{p,t} - h_t y_g + y_{s,t} = 0 \quad (\lambda_{h,t}), \;\; \forall t \in \mathcal{T} \tag{32}$$

and append $y_{s,t}$ to $\boldsymbol{y}$:

$$\boldsymbol{y} \triangleq [y_{e,1}, \ldots, y_{e,T}, \; y_{i,1}, \ldots, y_{i,T}, \; y_{p,1}, \ldots, y_{p,T}, \; y_g, y_{s,1}, \ldots, y_{s,T}]^\intercal \tag{33}$$

.

The structure of $\boldsymbol{V} \in \mathcal{R}^{2T \times 4T+1}$ for Problem (31) is:

$$\boldsymbol{V} = \left[\begin{array}{ccc:ccc:ccc:c:ccc} -1 & & & 1 & & & 1 & & & 0 & & & \\ & \ddots & & & \ddots & & & \ddots & & \vdots & & 0 & \\ & & -1 & & & 1 & & & 1 & 0 & & & \\ \hdashline & & & & & & 1 & & & -h_1 & 1 & & \\ & 0 & & & 0 & & & \ddots & & \vdots & & \ddots & \\ & & & & & & & & 1 & -h_T & & & 1 \end{array}\right] \tag{34}$$

Applying Algorithm 1 to $\lambda_1 y_{e,1}$ yields $\mathcal{J}_1 = \{1, \ldots, 2T\}$ and $\mathcal{N}_1 = \{2, 3, \ldots, 4T+1\}$. For illustrative purposes let us set $T = 3$, which implies $\mathcal{J}_1 = \{1, \ldots, 6\}$ and $\mathcal{N}_1 = \{2, 3, \ldots, 13\}$. Using (11):

$$\lambda_1 V_{11} y_{e,1} = \sum_{t\in\mathcal{T}} w_t\lambda_t - \sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) - c_g y_g - \overline{\mu}_g\overline{y}_g - \sum_{t\in\mathcal{T}\setminus\{1\}} \left(-x_{e,t}y_{e,t} + \overline{\mu}_{e,t}\overline{y}_{e,t}\right) \quad (35)$$

where the last sum comes from $n' = 2$ and $n' = 3$. Note that the last sum in (35) comes from (8), i.e. equations D_2 and D_3 for this problem. However, D_2 and D_3 introduce undesirable bilinear terms $x_{e,t}y_{e,t}$. This problem has an (intentionally) interesting structure: recall that we wish to linearize the sum $\sum_{t\in\mathcal{T}} \lambda_t y_{e,t}$. Therefore, allow us to use the opposite sides of D_2 and D_3, which makes (35):

$$\lambda_1 V_{11} y_{e,1} = \sum_{t\in\mathcal{T}} w_t\lambda_t - \sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) - c_g y_g - \overline{\mu}_g\overline{y}_g - \sum_{t\in\mathcal{T}\setminus\{1\}} \lambda_t V_{tt} y_{e,t}. \quad (36)$$

Now, if we look at the other equations from Algorithm 1 we can see why it is advantageous to leave the $\lambda_t V_{tt} y_{e,t}$ terms.

Applying Algorithm 1 to $\lambda_2 y_{e,2}$ yields $\mathcal{J}_2 = \{1, \ldots, 6\}$ and $\mathcal{N}_2 = \{1, 3, 4, \ldots, 13\}$. Using (11):

$$\lambda_2 V_{22} y_{e,2} = \sum_{t\in\mathcal{T}} w_t\lambda_t - \sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) - c_g y_g - \overline{\mu}_g\overline{y}_g - \sum_{t\in\mathcal{T}\setminus\{2\}} \lambda_t V_{tt} y_{e,t}. \quad (37)$$

Similarly, Applying Algorithm 1 to $\lambda_3 y_{e,3}$ yields $\mathcal{J}_3 = \{1, \ldots, 6\}$ and $\mathcal{N}_3 = \{1, 2, 4, 5, \ldots, 13\}$. Using (11):

$$\lambda_3 V_{33} y_{e,3} = \sum_{t\in\mathcal{T}} w_t\lambda_t - \sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) - c_g y_g - \overline{\mu}_g\overline{y}_g - \sum_{t\in\mathcal{T}\setminus\{3\}} \lambda_t V_{tt} y_{e,t}. \quad (38)$$

Adding the last three results while moving the $\boldsymbol{V}$ terms to the right hand side gives:

$$\begin{aligned}\sum_{t\in\mathcal{T}} \lambda_t y_{e,t} = {} & \frac{1}{V_{11}}\sum_{t\in\mathcal{T}} w_t\lambda_t + \frac{1}{V_{22}}\sum_{t\in\mathcal{T}} w_t\lambda_t + \frac{1}{V_{33}}\sum_{t\in\mathcal{T}} w_t\lambda_t \\ & - \frac{1}{V_{11}}\sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) \\ & - \frac{1}{V_{22}}\sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) \\ & - \frac{1}{V_{33}}\sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) \\ & - \frac{1}{V_{11}}\left(c_g y_g + \overline{\mu}_g\overline{y}_g\right) - \frac{1}{V_{22}}\left(c_g y_g + \overline{\mu}_g\overline{y}_g\right) \\ & - \frac{1}{V_{33}}\left(c_g y_g + \overline{\mu}_g\overline{y}_g\right) \\ & - \frac{1}{V_{11}}\sum_{t\in\mathcal{T}\setminus\{1\}} \lambda_t V_{tt} y_{e,t} - \frac{1}{V_{22}}\sum_{t\in\mathcal{T}\setminus\{2\}} \lambda_t V_{tt} y_{e,t} \\ & - \frac{1}{V_{33}}\sum_{t\in\mathcal{T}\setminus\{3\}} \lambda_t V_{tt} y_{e,t}. \end{aligned} \quad (39)$$

Now, if we substitute in $V_{tt} = -1\ \forall t \in \mathcal{T}$ and rearrange we get:

$$T\sum_{t\in\mathcal{T}} \lambda_t y_{e,t} = -T\sum_{t\in\mathcal{T}} w_t\lambda_t + T\sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t}\right) + T\left(c_g y_g + \overline{\mu}_g\overline{y}_g\right). \quad (40)$$

Finally, we can cancel T throughout the last result to get:

$$\sum_{t\in\mathcal{T}} \lambda_t y_{e,t} = c_g y_g + \overline{\mu}_g\overline{y}_g + \sum_{t\in\mathcal{T}} \left(c_{i,t}y_{i,t} + \overline{\mu}_{i,t}\overline{y}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t} - w_t\lambda_t\right). \quad (41)$$

Similar to the result (30), the time-indexed problem with linearized bilinear terms is:

$$\min_{\boldsymbol{x}} \sum_{t\in\mathcal{T}} \Big(x_{0,t}q_{0,t} + c_{i,t}y_{i,t} + \overline{y}_{i,t}\overline{\mu}_{i,t} + \overline{\mu}_{e,t}\overline{y}_{e,t} + \overline{\mu}_{p,t}\overline{y}_{p,t} - w_t\lambda_t\Big) + c_g y_g + \overline{y}_g\overline{\mu}_g \tag{42a}$$

$$\text{s.t. } x_{0,t} \geq 0,\ x_{e,t} \geq 0, \qquad \forall t \in \mathcal{T} \tag{42b}$$

$$x_{0,t} \geq x_{i,t}, \qquad \forall t \in \mathcal{T} \tag{42c}$$

$$x_{i,t} - y_{i,t} + y_{e,t} = d_t,, \qquad \forall t \in \mathcal{T} \tag{42d}$$

$$y_{e,t} \perp y_{i,t}, \qquad \forall t \in \mathcal{T} \tag{42e}$$

$$y_{e,t} \perp \overline{\mu}_{e,t},\ y_{i,t} \perp \overline{\mu}_{i,t},\ y_{p,t} \perp \overline{\mu}_{p,t}, \qquad \forall t \in \mathcal{T} \tag{42f}$$

$$y_{e,t} \perp \underline{\mu}_{e,t},\ y_{i,t} \perp \underline{\mu}_{i,t},\ y_{p,t} \perp \underline{\mu}_{p,t}, \qquad \forall t \in \mathcal{T} \tag{42g}$$

$$y_{p,t} - h_t y_g \perp \mu_{h,t}, \qquad \forall t \in \mathcal{T} \tag{42h}$$

$$y_g \perp \overline{\mu}_g,\ y_g \perp \underline{\mu}_g \tag{42i}$$

$$-y_{e,t} + y_{i,t} + y_{p,t} = w_t, \qquad \forall t \in \mathcal{T} \tag{42j}$$

$$-x_{e,t} + \lambda_t + \overline{\mu}_{e,t} - \underline{\mu}_{e,t} = 0, \qquad \forall t \in \mathcal{T} \tag{42k}$$

$$c_{i,t} - \lambda + \overline{\mu}_{i,t} - \underline{\mu}_{i,t} = 0, \qquad \forall t \in \mathcal{T} \tag{42l}$$

$$-\lambda_t + \overline{\mu}_{p,t} - \underline{\mu}_{p,t} + \mu_{h,t} = 0, \qquad \forall t \in \mathcal{T} \tag{42m}$$

$$c_g + \overline{\mu}_g - \underline{\mu}_g - \sum_{t\in\mathcal{T}} (\mu_{h,t} h_t) = 0 \tag{42n}$$

Appendix C
Solution Time Comparison

Using Problem (42) we compare solution times with and without the upper level bilinear terms $\lambda_t y_{e,t}$ replaced with the linearization derived in Section III and shown in (42a). Both the mixed integer-linear and the mixed integer-bilinear problems were solved using Gurobi 9.1 on a Macbook Pro with 8GB of RAM and 8 Apple M1 cores using special order sets of type I (SOS1) for the complementary constraints.

An hourly time step was chosen and one year is modeled, making $\mathcal{T} = 8,760$. The model parameters are $c_g = 0.5, c_{i,t} = 1, w_t = 1, d_t = 1, q_{0,t} = 1,\ \forall t \in \mathcal{T}$, and h_t was set equal to a normalized solar PV production factor from [14].

Both the linearized and bilinear problems have 43,801 SOS1 constraints and 157,682 continuous variables. After 4 seconds in the presolve the linear problem has 4,119 SOS1 constraints, 17,861 continuous variables, and 998 binary variables. The linear problem solves in less than four seconds with a tolerance of 0.01%. After 0.5 seconds in the presolve the bilinear problem has 8,753 SOS constraints, 4,250 bilinear constraints, and 32,254 continuous variables. The bilinear model takes 380 seconds to get to a less than 2% gap and then gets stuck at a 1.55% gap after 459 seconds until Gurobi kills the problem at 5,600 seconds due to running out of RAM.

References

[1] R. Raineri, "Chile: Where it all started," *Electricity market reform: an international perspective*, pp. 77–108, 2006.

[2] C. Ruiz and A. J. Conejo, "Pool strategy of a producer with endogenous formation of locational marginal prices," *IEEE Transactions on Power Systems*, vol. 24, no. 4, pp. 1855–1866, 2009.

[3] R. Fernández-Blanco, J. M. Arroyo, and N. Alguacil, "On the solution of revenue-and network-constrained day-ahead market clearing under marginal pricing—part i: An exact bilevel programming approach," *IEEE Transactions on Power Systems*, vol. 32, no. 1, pp. 208–219, 2016.

[4] B. Cornélusse, I. Savelli, S. Paoletti, A. Giannitrapani, and A. Vicino, "A community microgrid architecture with an internal local market," *Applied Energy*, vol. 242, pp. 547–560, 2019.

[5] S. Wogrin, S. Pineda, and D. A. Tejada-Arango, "Applications of bilevel optimization in energy and electricity markets," in *Bilevel Optimization*. Springer, 2020, pp. 139–168.

[6] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[7] F. Glover, "Improved linear integer programming formulations of nonlinear integer problems," *Management Science*, vol. 22, no. 4, pp. 455–460, 1975.

[8] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," *Journal of the operational Research Society*, vol. 32, no. 9, pp. 783–792, 1981.

[9] (2020, September) FERC News Order 2222. [Online]. Available: https://www.ferc.gov/news-events/news/ferc-opens-wholesale-markets-distributed-resources-\landmark-action-breaks-down

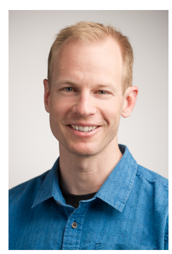
[10] G. Brinkman, D. Bain, G. Buster, C. Draxl, P. Das, J. Ho, E. Ibanez, R. Jones, S. Koebrich, S. Murphy *et al.*, "The north american renewable integration study (naris): A canadian perspective," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2021.

[11] Bileveljump: A bilevel optimization extension of the jump package. [Online]. Available: https://github.com/joaquimg/BilevelJuMP.jl

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

[13] P. Andrianesis, M. Caramanis, R. D. Masiello, R. D. Tabors, and S. Bahramirad, "Locational marginal value of distributed energy resources as non-wires alternatives," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 270–280, 2019.

[14] National Renewable Energy Laboratory, "PVWatts Calculator," 2020. [Online]. Available: https://developer.nrel.gov/docs/solar/pvwatts/



Nicholas D. Laws Nicholas D. Laws is a PhD student in the Webber Energy Group at The University of Texas at Austin and a Research Engineer at the National Renewable Energy Laboratory. His research focuses on the integration of clean distributed energy resources in ways that benefit all stakeholders. He holds a Bachelor of Science in Aerospace Engineering from Boston University and a Master of Science from Dartmouth College, Thayer School of Engineering.

Grani A. Hanasusanto Grani A. Hanasusanto is an Assistant Professor of Operations Research and Industrial Engineering at The University of Texas at Austin (UT). Before joining UT, he was a postdoctoral researcher at the College of Management of Technology at École Polytechnique Fédérale de Lausanne. He holds a PhD degree in Operations Research from Imperial College London and an MSc degree in Financial Engineering from the National University of Singapore. He is the recipient of the 2018 NSF CAREER Award. His research focuses on the design and analysis of tractable solution schemes for decision-making problems under uncertainty, with applications in operations management, energy systems, finance, machine learning and data analytics.