# Computational Approaches for Efficient Scheduling of Steel Plants as Demand Response Resource

Xiao Zhang*, Gabriela Hug†, Zico Kolter‡, and Iiro Harjunkoski§
* Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA
† Information Technology and Electrical Engineering, ETH Zurich, Switzerland
‡ Computer Science, Carnegie Mellon University, Pittsburgh, USA
§ ABB Corporate Research, Ladenburg, Germany

*Abstract*—**Demand response is seeing increased popularity worldwide and industrial loads are actively taking part in this trend. As a host of energy-intensive industrial processes, steel plants have both the motivation and potential to provide demand response. However, the scheduling of steel plants is very complex and the involved computations are intense. In this paper, we focus on these difficulties and propose methods such as adding cuts and implementing an application-specific branch and bound algorithm to make the computations more tractable.**

*Index Terms*—**Branch and bound algorithm, mixed integer programming, demand response, steel plant scheduling, resource task network.**

## I. Introduction

Demand response is becoming increasingly important in the electric power system to support the integration of variable renewable generation, thereby contributing to cutting down the carbon footprint of the electric power system. Load shifting allows for reducing peak load and adjusting the load to the level of available generation. Demand response can also help in balancing the intermittent wind and solar power by utilizing the flexibility provided by loads. With its great potential to enhance the power system's operational flexibility, demand response has gained significant attention in recent years [1]–[3]. Demand response can be provided by residential, commercial, and industrial loads. For example, there have been discussions and studies about demand response provided by buildings [4], storages [5], data centers [6], [7], residential areas [8], aluminum smelters [9], [10], air separation units [11], as well as steel plants [12].

However, for many potential players, especially the smaller loads in the residential and commercial areas, it may not be profitable to participate in demand response, as the payments are usually not high enough for them to justify the investment in implementing the platforms for participation. Meanwhile, industrial loads, given their large energy consumption, are ideal candidates for providing demand response as they have both the potential to provide demand response as well as the motivation to do so [13]–[15]: most industrial plants are already equipped with the infrastructures for control, measurement, and communications that are required for demand response providers; many industrial loads are able to offer large, fast, and accurate adjustments in their power consumption; the demand response programs are also financially appealing to the industrial plants, especially to those energy-intensive plants who treat demand response as an opportunity to increase their profits by making full use of their assets. The range of industrial loads that can support the operation of the electric power system include aluminum smelting pots, steel melting furnaces, fans, freezers, pumps, etc.

On the other hand, the computations associated with industrial loads demand response provision are often intense. The goal of the industrial plants is to optimize the schedule of production activities while fitting their power consumption to the need of power system operation. However, the industrial production process is usually very complex with multiple process stages, and the raw material needs to be processed by each of these stages following the correct sequence. There are many critical constraints involved in the production, e.g. the intermediate products need to be processed quickly to prevent expensive reheating and the industrial plants also need to satisfy the orders/needs from their customers with respect to the final products quantity and quality. Consequently, the industrial scheduling is usually a large scale optimization problem with a lot of integer variables that is difficult to solve. There has been a significant number of papers studying the efficient modeling and computing for industrial scheduling, with the intention to address the computational difficulties. For example, in [16], the long-term scheduling of steel plants is studied, and instead of formulating one large, intractable MIP problem, a decomposition strategy is proposed to generate smaller problems that can often be solved to optimality. In [17], an alternative continuous time formulation focused on the relative positions of tasks and time periods is presented, which improves significantly the computation time of the steel manufacturing short-term scheduling problem discussed in [18]. In [19], a resource-task network is used to provide a generic modeling framework for short-term production scheduling under energy constraints, and three alternative process models for steel plant scheduling are proposed, in which a trade-off is observed between accurate representation of steel manufacturing and computational performance.

In this paper, we focus on the optimal scheduling of steel plants to provide demand response and propose approaches to overcome the associated computational difficulties. The scheduling models proposed in [19] with a finer time grid resolution cannot be solved to optimality within hours by commercial solvers, as the number of binary variables and

The authors would like to acknowledge the financial support by ABB.

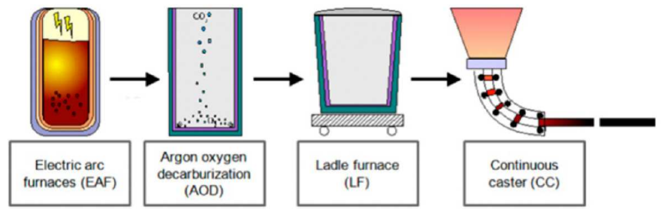

CO
Electric arc furnaces (EAF)
Argon oxygen decarburization (AOD)
Ladle furnace (LF)
Continuous caster (CC)

Figure 1. Production process of steel manufacturing [19]

constraints increase with the number of time slots. However, a finer time grid is desired to accurately represent the production activities. In this paper, we propose methods to make the relevant computations more tractable.

The remaining of the paper is organized as follows: Section II introduces the considered scheduling problem. Section III explains the resource task network modeling framework and introduces the mathematical formulations for the scheduling problem. In Section IV, the computational approaches are presented. The case study of a typical steel plant is discussed in Section V to demonstrate the effectiveness of the proposed approaches, based on which the conclusions are drawn in Section VI.

## II. Steel Plants Scheduling

Figure 1 displays the typical process of steel manufacturing. There are four process stages and each process stage is referred to by the operation equipment in this stage, i.e. electric arc furnaces (EAF), argon oxygen decarburization units (AOD), ladle furnaces (LF), and continuous casters (CC). The input to the process is the raw material, which is solid metal scrap recycled from abandoned steel products such as discarded cars. The output from the process is the final products, i.e. the slabs. The EAF melts the raw material into molten metal; the AOD removes the impurities and reduces the carbon content from the molten metal; the LF further refines the molten metal and transport the molten metal to the CC; the CC casts the refined molten metal into slabs with different shapes. The final products have different characteristics such as grade, slab width, thickness, etc. according to the needs of the customers. Different final products require different chemical ingredients and different processing times for each of the process stages.

The process equipment in the first three stages operate in batch mode, i.e. these equipment (e.g. furnace) can process a certain amount of metal at one time, and that quantity of metal remains consistent along different stages. Each such amount of metal is termed as a *heat*. By *heat*, we could quantify the throughput of the steel plant. For instance, a medium sized steel plant is able to deliver 20 heats of final products within one day. Unlike the first three stages, the casting stage operates continuously. However, due to the extreme conditions in the caster, it can only process a limited number of heats, after which it needs maintenance such as changing the caster mold and tundish before further operation, to ensure desired steel quality and dimension. Hence, the steel plant operators usually combine several heats sharing the same or very similar grade characteristics and shape requirements to form a campaign (a group of heats), and cast these heats together. The casters are maintained between casting two campaigns. The casting order for the heats within one campaign should follow certain rules and the casting sequence must not be interrupted.

The steel manufacturing is highly energy-intensive but it also has the flexibility in scheduling its production activities to follow a desired energy consumption curve over time. Besides, a steel plant can also adjust its loading level by switching the taps of the transformers that supply power for its equipment, which enables the provision of spinning reserve. However, steel manufacturing has been recognized as one of the most difficult industrial processes for scheduling, as the production of steel is a large-scale, multi-product, multi-stage batch process that involves parallel equipment and critical production-related constraints.

## III. Mathematical Modeling

The resource-task network (RTN) modeling framework has been widely adopted to model and optimize the scheduling of industrial plants. Same as in [12], the following mathematical formulations for steel plant scheduling are developed based on the RTN modeling framework proposed in [19].

### A. Resource-task Network Modeling

The RTN modeling framework is able to explicitly represent the complex chemical processes by mathematical formulations in a systematic way. The RTN of a steel plant is illustrated in Fig. 2, in which the resources are denoted by circles and the tasks are represented by rectangles. The resource is a very general concept and it includes all entities that are involved in the production such as equipment units, intermediate and final products, and utilities such as electricity. The set of resources is denoted by $\mathbb{S}$, i.e. $\mathbb{S} = \{\text{EAF, AOD, LF, CC}\} \cup \{\text{EA}_h^\text{s}, \text{EA}_h^\text{d}, \text{AL}_h^\text{s}, \text{AL}_h^\text{d}, \text{LC}_h^\text{s}, \text{LC}_h^\text{d}, \text{H}_h | h \in \mathbb{H}\} \cup \{\text{EL}\}$ with $\mathbb{H}$ as the set of heats to produce. Intermediate products at different locations (start or destination of the corresponding transfer) are treated as different resources and are specified by superscripts $s$ or $d$, respectively. For example, $\text{AL}_h^\text{d}$ represents the intermediate product between stage AOD and LF that has already been transferred to the LF stage and is waiting to be processed.

We use a discrete time grid with uniform slot width of $t_0$ and we use $T$ to denote the total number of time slots. A matrix $Y \in \mathbb{R}^{|\mathbb{S}| \times T}$ is used to denote the available amounts of these resources at all time slots, in which $|\mathbb{S}|$ is the size of $\mathbb{S}$. Each element in $Y$ is a continuous variable, $y_{s,t}$, which represents the available amount of resource $s$ at time $t$. For example, $y_{\text{EAF},t} = 3$ means there are three furnaces available at time slot $t$; $y_{\text{EL},t} = 50$ MWh means 50 MWh of electric energy is used by the steel plant during time slot $t$. Due to their physical meanings, most $y_{s,t}$ can actually only take discrete values such as 0, 1, or 2. However, they are modeled as continuous variables since a larger number of discrete variables generally leads to a problem that is more difficult to solve. As discussed later the constraints in the optimization model will enforce these variables to take discrete values.

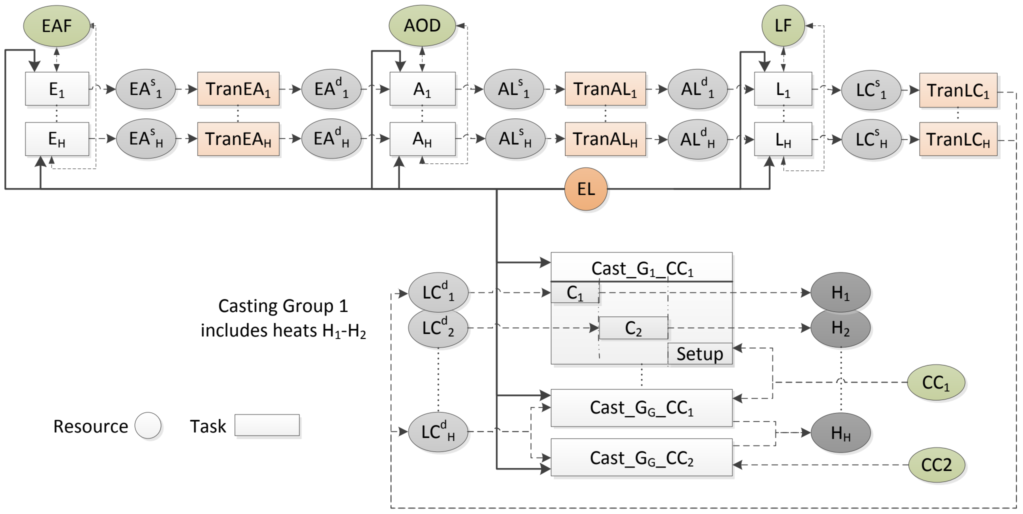
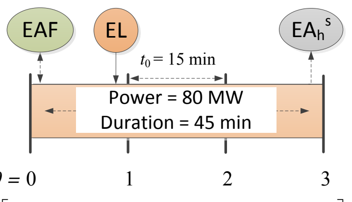


Figure 2. Resource task network for a steel plant.

There are seven kinds of tasks in total: four operational tasks at each of the four stages and three transfer tasks between the stages. For all kinds of tasks except for the casting task in the last stage, the number of tasks is equal to the number of heats to produce; these tasks are denoted by the task type with the corresponding heat as subscript, e.g. $\mathrm{E}_h$ stands for the melting of heat $h$ in the EAF stage, and $\mathrm{EA}_h$ (without superscript $s$ or $d$) denotes the transfer of heat $h$ between stage EAF and AOD. Meanwhile, the casting task is denoted as $\mathrm{C}_{g,u}$ which stands for the casting of group $g$ by caster unit $u$. As mentioned before, the tasks in the CC stage are executed by group instead of by heat. Besides, since generally different casters are designed for casting different slabs, we need to specify the caster for the casting task. In other words, $\mathrm{C}_{g_1,u_1}$ is different from $\mathrm{C}_{g_1,u_2}$, e.g. their processing durations might be different due to the different casters. We use $\mathbb{K}$ to denote the set of tasks, i.e. $\mathbb{K} = \{\mathrm{E}_h, \mathrm{EA}_h, \mathrm{A}_h, \mathrm{AL}_h, \mathrm{L}_h, \mathrm{LC}_h | h \in \mathbb{H}\} \cup \{\mathrm{C}_{g,u} | g \in \mathbb{G}, u \in \mathbb{CC}\}$, with $\mathbb{G}$ and $\mathbb{CC}$ as the set of casting campaign groups and available casters, respectively.

The starting times of all the tasks are denoted by a $|\mathbb{K}|$-by-$T$ binary matrix $X$, in which $|\mathbb{K}|$ is the size of $\mathbb{K}$. Each element of $X$ is a binary variable, $x_{k,t}$, which represents whether task $k$ starts at time slot $t$. For example, $x_{\mathrm{E}_h,t} = 1$ means the processing of heat $h$ in stage EAF starts at time slot $t$; only one out of $x_{E_h,t}, t = \{1, \ldots, T\}$ is non-zero since this task only takes place once.

In Fig. 2, the networks of how each task interacts with each resource are represented by arrows. For each task $k \in \mathbb{K}$, its interaction parameter $\Delta^k$ is a $|\mathbb{S}|$-by-$(\tau_k + 1)$ matrix that quantifies how much task $k$ consumes/generates of each of the resources as it proceeds, in which $\tau_k$ denotes its duration as a number of time slots. For example, its element $\Delta^k_{s,1}$ quantifies the interaction between task $k$ and resource $s$ at the beginning of the first time slot during this task, and $\Delta^k_{s,\tau_k+1}$ quantifies the interaction at the end of the last time slot. A zero element means that there is no interaction, and $\Delta^k$ is very sparse as a task typically only interacts with a few resources. To better understand the interaction parameters, an example for



$$\Delta^{E_h} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\left[\begin{array}{cccc} -1 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 20 & 20 & 20 & 0 \end{array}\right]\begin{array}{l} \text{EAF} \\ \\ \text{EA}_h^s \\ \\ \text{EL [MWh]} \end{array}$$

Figure 3. Illustration of interaction parameters for a melting task.

a melting task is given in Fig. 3: the duration of the melting task is 45 minutes and the time slot width is $t_0 = 15$ minutes. This task interacts with resources EAF, EL and $\mathrm{EA}^s_h$, hence its interaction parameter matrix are all zeros except for these three rows. At the beginning of the task, it uses one furnace so it reduces EAF by one. Meanwhile at the end of the task, it releases that furnace hence EAF is increased by one; $\mathrm{EA}^s_h$ is also increased by one as it has just been generated. Besides, the melting task consumes electric energy at every time slot within its duration.

## *B. Mathematical Formulations*

For the presentation of the following computational approaches, here we summarize the relevant formulations given in [12].

*1) Resource Balance:* The resource balance equation describes the interaction between each resource and its relevant tasks, as in

$$y_{s,t} = y_{s,t-1} + \sum_{k\in\mathbb{K}} \sum_{\theta=0}^{\tau_k} \Delta^k_{s,\theta} \cdot x_{k,t-\theta} \;\; \forall s \in \mathbb{S}_{\neg\{\mathrm{EL}\}}, \forall t \quad (1)$$

in which the value of resource $s$ at time slot $t$ is equal to its previous value at $t-1$ adjusted by the amounts generated/consumed by all the tasks, and $\mathbb{S}_{\neg\{\mathrm{EL}\}}$ stands for the set of all the resources except EL. Only nonzero $\Delta^k_{s,\theta}$ implies actual interaction. Besides, the interaction occurs at time slot $t$ only if task $k$ starts $\theta$ earlier than $t$ ($x_{k,t-\theta}=1$), with $\theta \leq \tau_k$. Equation (1) enforces the continuous variable $y_{s,t}$ to only take integer values, because: (1) the interaction parameters $\Delta^k_{s,\theta}$ for these resources are integers, (2) $x_{k,t-\theta}$ are binary variables, and (3) the initial values for the resources are integers (zero or the number of available equipment).

Similarly, the electric energy usage of the steel plant is calculated as

$$y_{\mathrm{EL},t} = \sum_{k\in\mathbb{K}} \sum_{\theta=0}^{\tau_k} \Delta^k_{\mathrm{EL},\theta} \cdot x_{k,t-\theta} \qquad \forall t \tag{2}$$

where $\Delta^k_{\mathrm{EL},\theta}$ is the electricity used by task $k$ at the $\theta$-th time slot within its execution.

*2) Task Execution:* The following constraints make sure that each heat is processed exactly once by all types of tasks within the scheduling horizon, as in

$$\begin{aligned} &X_{\mathbb{K}_{\neg\mathbb{CC}}} \mathbb{1}_T = \mathbb{1} \\ &\mathbb{1}' X_{\mathbb{C}_u} \mathbb{1}_T = 1 \quad \forall u \in \mathbb{CC} \end{aligned} \tag{3}$$

in which $X_{\mathbb{K}_{\neg\mathbb{CC}}}$ denotes $X$ without the rows involving the casting tasks; $X_{\mathbb{C}_u}$ denotes the rows of $X$ corresponding to casting tasks by caster $u$; $\mathbb{1}_T$ means a vector of 1s with length $T$ and $\mathbb{1}$ stand for vectors of 1s with appropriate lengths.

Of course, only one of $\mathrm{C}_{g_1,u_1}$ and $\mathrm{C}_{g_1,u_2}$ will take place as group $g_1$ should be casted exactly once.

*3) Waiting Time:* In steel plant operations, the transfer tasks are usually enforced to be executed immediately after the completion of its preceding processing task. This can be enforced by setting

$$Y_{\mathrm{EA^s}} = \mathbf{0} \tag{4}$$

in which $Y_{\mathrm{EA^s}}$ stands for the rows of $Y$ corresponding to the intermediate products $\mathrm{EA^s}$; $\mathbf{0}$ is a zero matrix with the same dimensions as $Y_{\mathrm{EA^s}}$. The above constraint implies that all intermediate products $\mathrm{EA^s}$ do not stay at any time slot. Similar constraints apply for the intermediate products $\mathrm{AL^s}$ and $\mathrm{LC^s}$.

To simplify the problem, we assume the transfer tasks are independent of the specific heats and the exact locations of the units. The transfer times are denoted as $w_{\mathrm{EA}}$, $w_{\mathrm{AL}}$, and $w_{\mathrm{LC}}$, and the maximum allowable transportation times which prevent adverse cooling effects are denoted as $\bar{w}_{\mathrm{EA}}$, $\bar{w}_{\mathrm{AL}}$, and $\bar{w}_{\mathrm{LC}}$. The maximum waiting time constraint for intermediate products should be satisfied, in order to avoid the expensive reheating:

$$Y_{\mathrm{EA}^{\mathrm{d}}_h} \mathbb{1}_T \leq \frac{(\bar{w}_{\mathrm{EA}} - w_{\mathrm{EA}})}{t_0} \mathbb{1} \tag{5}$$

in which $Y_{\mathrm{EA^d}}$ stands for the rows of $Y$ corresponding to intermediate products $\mathrm{EA^d}$ before the transfer. The left side of the constraint corresponds to the number of time slots during which the intermediate product is waiting before being processed. Similar constraints apply for intermediate products $\mathrm{AL^d}$ and $\mathrm{LC^d}$.

*4) Product Delivery:* The final products should be available at the end of the time horizon, which is enforced by setting

$$y_{\mathrm{H}_h,T} = 1 \qquad \forall h \tag{6}$$

in which $y_{\mathrm{H}_h,T}$ stands for the availability of the final product for heat $h$ at the end of the scheduling horizon.

*5) Objective Function:* The objective of the scheduling is to minimize the total electricity cost of the steel production. Given the energy price vector $\lambda_{\mathrm{EL}} \in \mathbb{R}_T$, the overall optimization problem is formulated as

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & Y_{\mathrm{EL}} \lambda_{\mathrm{EL}} \\ \text{subject to} \quad & (1) - (6) \\ & x_{s,t} \in \{0,1\},\ \ y_{s,t} \in [0, \bar{y}_s],\ \ \forall s, \forall t \end{aligned}$$

in which $\bar{y}_s$ is the upper bound for the available amount of resource $s$. For example, $\bar{y}_{\mathrm{EAF}}$ equals to the number of EAF furnaces and $\bar{y}_{\mathrm{EL}}$ equals to the summation of energy usage by all the equipment units in one time slot.

## IV. Computations of Steel Plants Scheduling

In the RTN-based scheduling model described above, the computation difficulty largely depends on the problem scale and the time resolution. A 5-minute resolution is desired to accurately model the production activities, but will result in a MIP problem with thousands of binary variables which cannot be solved to optimality even within hours by commercial MIP solvers. Meanwhile, steel plant operators cannot afford such a long computation time, as they constantly need to make fast decisions to adapt the production plans. We try to improve the computations in the following two ways: from the modeling aspect, we add additional constraints as cuts to reduce the search space for the MIP problem, and from the algorithmic aspect, we design a tailored branch and bound algorithm that utilizes the knowledge from steel manufacturing to speed up the algorithm.

### *A. Additional constraints as cuts*

In steel manufacturing, there are many tasks that are equivalent to each other, e.g. the decarburization of molten metal for two similar batches of products. We can impose an enforced processing order for these tasks, thus the search space of the MIP problem is reduced. Several other types of constraints can also be imposed which reduce the feasible region but with potential sacrifice to the degree of optimality, an example is to restrict the start time of EAF tasks to time intervals that have lower energy price expectations. Here we consider imposing processing orders for the tasks.

In steel manufacturing, the casting sequence for heats belonging to the same casting group are pre-specified - this pre-specified processing sequence results from expert experiences or casting optimization. Intuitively, that processing sequence should also apply to the other three stages. For instance, suppose the casting group G1 consists of heats 1, 2, and 3, and they will be casted one by one sequentially, which means that the intermediate product $\mathrm{LC}_1$ will be casted first. Consequently, $\mathrm{LC}_1$ should be generated first, i.e. heat 1 should be processed before the other two heats in the 3rd stage. Hence, we can define a set of ordered tasks, denoted by $\mathbb{O}$, whose processing sequences are pre-specified. For instance, $\mathbb{O} = \{(\mathrm{E}_1, \mathrm{E}_2), (\mathrm{E}_2, \mathrm{E}_3), (\mathrm{A}_1, \mathrm{A}_2), (\mathrm{A}_2, \mathrm{A}_3), (\mathrm{L}_1, \mathrm{L}_2), (\mathrm{L}_2, \mathrm{L}_3)\}$ for scheduling G1 of heats 1, 2 and 3. The imposed additional constraints (cuts) on processing order enforcement can be written as follows:

$$\sum_{t' \leq t} (x_{k_1,t'} - x_{k_2,t'}) \geq 0 \quad \forall t, (k_1, k_2) \in \mathbb{O} \tag{7}$$

in which the ordered tasks set $\mathbb{O}$ considers the processing of heats belonging to the same group for each of the first three stages. In steel manufacturing practice, the above additional constraint is also beneficial as it follows the first-in-first-out principle and reduces the chance of over-waiting and reheating for the intermediate products.

### B. Tailored branch and bound algorithm

The MIP problems in industrial scheduling are usually solved by commercial solvers [14], [20]. These commercial solvers are powerful, but are designed to handle general optimization problems. We develop a tailored branch and bound algorithm to utilize the special features in steel manufacturing: the heats belonging to the same campaign group are generally processed close to each other.

The branch and bound algorithm is presented in Fig. 4 in which $q$ and $q2$ are priority queues that store the relaxation solutions at each iteration and the achieved feasible integer solutions, respectively; $q$ provides the lower bounds for the mixed-integer minimization problem, while $q2$ provides the upper bounds. SolveRelaxation($C$) is a function that takes input $C$ and solves the relaxation of the original problem plus constraints in $C$; the relaxation is a linear programming problem and we solve it by using CPLEX's LP solver; the function returns $(f, x^*, y^*, C)$, i.e. the optimal objective value, the optimal values of relaxed integer variables and continuous variables, as well as the corresponding constraints $C$. The function input $C$ is actually defined as $C = [(a_{k1}, b_{k1}), (a_{k2}, b_{k2}), \ldots, (a_K, b_K)]$, which specifies the start times for each task. For instance, $(a_{k1}, b_{k1})$ restricts the start time of task $k1$ to be between $a_{k1}$ and $b_{k1}$ - the constraint is actually implemented by setting the upper bounds for $x_{k1,t}$ as zero for $t$ outside of $(a_{k1}, b_{k1})$ while leaving the upper bounds for other $t$ as one, i.e.

$$\overline{x}_{k1,t} = \begin{cases} 1, & \text{if } a_{k1} \leq t < b_{k1} \\ 0, & \text{otherwise} \end{cases}$$

GetIntegerSolutionHeuristics() is a heuristics method that packs all the tasks to the earliest available equipment units to get a feasible integer solution, which serves as the initial upper bound for the algorithm. Rounding() tries to round the relaxation solution to be integer, and returns the integer solution if it is feasible.

```
1: procedure TailoredBranchBound
2:     $q \leftarrow$ Priority-Queue()            ▷ pops largest objective first
3:     $q2 \leftarrow$ Priority-Queue()           ▷ pops smallest objective first
4:     $q$.push(SolveRelaxation({ }))
5:     $q2$.push(FindIntegerSolutionHeuristics())
6:     while $q$ not empty do
7:         $(f, x, y, C) \leftarrow q$.pop()
8:         $q2$.push(Rounding($(f, x, y, C)$))
9:         if $q2$.first - $f \leq \epsilon$ then
10:            return $q2$.pop()
11:        else
12:            for $C_i$ in BranchNodes($C$) do
13:                $q$.push(SolveRelaxation($C_i$))
14:            end for
15:        end if
16:    end while
17: end procedure
```

Figure 4. Tailored branch and bound algorithm

```
1: function BranchNodes($C$)
2:     if C == { } then
3:         return $[(0, T), \ldots, (0, T)]$
4:     else
5:         $k^* \leftarrow \arg\max_{k \in L.\text{keys}}(b_k - a_k)$
6:         if $b_{k^*} - a_{k^*} > \epsilon_d$ then
7:             $m^* \leftarrow \text{int}(\frac{b_{k^*} - a_{k^*}}{2})$
8:             $\{k : (d_a, d_b)\} \leftarrow L[k^*]$
9:             $C_1 \leftarrow [\ldots, (a_{k^*}, m^*), (a_{k^*} + d_a, m^* + d_b), \ldots]$
10:            $C_2 \leftarrow [\ldots, (m^*, b_{k^*}), (m^* + d_a, b_{k^*} + d_b), \ldots]$
11:            return $\{C_1, C_2\}$
12:        else
13:            $k^* \leftarrow \arg\max_{k \in \mathbb{K}}(b_k - a_k)$
14:            $m^* \leftarrow \text{int}(\frac{b_{k^*} - a_{k^*}}{2})$
15:            $C_1 \leftarrow [\ldots, (a_{k^*}, m^*), \ldots]$
16:            $C_2 \leftarrow [\ldots, (m^*, b_{k^*}), \ldots]$
17:            return $\{C_1, C_2\}$
18:        end if
19:    end if
20: end function
```

Figure 5. Branch by leader heats

In order to enforce the heats belonging to the same campaign group be processed close to each other, we term the first heat in each campaign group as the *leader*, and call the other heats belonging to the same group as its *followers*. Similar to the discussions in Section IV-A, we require the leader to be processed first, and require its followers to be processed within the time ranges calculated according to their processing time durations. For example, consider group G1 of heats 1, 2 and 3 in the EAF stage (with two EAF units). The leader here is task $\mathrm{E}_1$, and suppose its start time is within $(a, b)$ at a certain node in the branch and bound algorithm. As there are two available furnaces, we require its followers $\mathrm{E}_2$ to be started between $(a, b + \tau_{E_1})$ and $\mathrm{E}_3$ to be started between $(a + \tau_{\mathrm{E}_1}, b + \tau_{\mathrm{E}_1})$. The principle of this requirement is to enforce the offsets and delays as if these followers are packed sequentially to the available equipment units. The relationship of start times for the above example can be described by the following dictionary $L$ that maps the leader to its followers and the corresponding offsets:

$$L[k_{E1}] = \{k_{E2} : (0, \tau_{E1}), k_{E3} : (\tau_{E1}, \tau_{E1})\}$$

With this concept of leader and followers and the restriction on start time described above, instead of branching on the start time intervals for all tasks, we can only branch on the leader tasks and restrict the start times for its follower tasks according to a pre-constructed $L$. The proposed branching method is described in Fig. 5 with parameter $\epsilon_d$ as the threshold to switch between branching by leader tasks and branching by all tasks. This method will greatly reduce the complexity of the branching procedure.

## V. Numerical Studies

Numerical studies on the daily scheduling for a typical steel plant are presented in this section to demonstrate the effectiveness of the proposed methods.

### A. Problem Parameters

The steel plant layout and parameters are taken from the typical scheduling example in [19]: there are two parallel units for each of the four stages, and Table I lists the nominal power consumption rates of these units; the group correspondences of the heats to produce are given in Table II, and Table III shows their nominal processing times; the transfer times $w_{\text{EA}}$, $w_{\text{AL}}$, and $w_{\text{LC}}$ are 10, 4, and 10 minutes, respectively, and the maximum waiting times $\bar{w}_{\text{EA}}$, $\bar{w}_{\text{AL}}$, and $\bar{w}_{\text{LC}}$ are 240, 240, and 120 minutes; the caster setup times are 70 minutes for CC1 and 50 minutes for CC2, which are the times needed for equipment maintenance between casting two groups of heats. From these tables we can tell that the EAF is the most energy-intensive process stage. The hourly energy prices for the case studies are taken from MISO, as displayed in Fig 6(c).

### B. Computational Results

Table IV presents the computational results for the methods proposed in Section IV: the column *Groups* gives the campaign groups to produce, e.g. the first row denotes scheduling groups 1 and 2 with the heats as indicated in Table II; the column *c0* stands for solving the original model by CPLEX's MIP solver, while the column *c1* stands for the method proposed in Section IV-A and the column *b1* denotes the tailored branch and bound algorithm proposed in Section IV-B ($\epsilon_d$ is chosen as 4); the row *Obj* gives the final objective value of the MIP problem; the row *CPU* shows the computation time for the corresponding test case, where the maximum computation time limit is set to 7200s; the row *lpNum* gives the iteration number of the solving process, where the maximum iteration number is set to 10000. Note that for the test case G1-5, the relaxation solution by CPLEX happens to be a feasible integer solution, hence the corresponding *lpNum* is 0.

From the computational results displayed above, we can observe the following: (1) by imposing additional constraints, method c1 reduces the computation time as well as the iteration number, and the final objective values remain the same except for a 0.005% increase for case G1-3; (2) the tailored branch and bound algorithm b1 is able to greatly reduce the iteration number, but the final objective value sees an increase of around 1%. The optimality loss suffered by b1 can be explained by the restriction that we require the heats in the same group be processed close to each other. For example, Fig. 6 displays the scheduling results comparison for test case G1-2, where method b1 schedules the LF/CC processing of heats 5, 6, 7 and 8 close to the EAF/AOD stages, which loses the opportunity to utilize the price valley around hours 12-17. Also note that the computation time for a single iteration of method b1 is much longer than that of c0 or c1 - this is due to the computation overhead with calling the LP solver, while the solving process of the CPLEX MIP solver has been coherently optimized. Hence, one has to be careful when comparing CPU time for b1 with any of the other as the time probably could be improved by a more professional implementation of the proposed branch and bound algorithm. For the numerical studies here, it might be more appropriate to focus on iteration numbers for which b1 achieves a significant improvement in most of the cases.

Another issue we want to emphasize here is the rounding procedure in algorithm b1. The current algorithm rounds each variable to its nearest integer value, which seldom succeeds in yielding a feasible solution, as there are so many constraints on binary variables in the scheduling model. As shown in Fig. 7, the upper bound seldom changes along the solving process. We plan to improve the rounding procedure by taking

TABLE I. Nominal power consumptions [MW] [19]

| equipment | $EAF_1$ | $EAF_2$ | $AOD_1$ | $AOD_2$ | $LF_1$ | $LF_2$ | $CC_1$ | $CC_2$ |
|---|---|---|---|---|---|---|---|---|
| power | 85 | 85 | 2 | 2 | 2 | 2 | 7 | 7 |

TABLE II. Steel heat/group correspondence [19]

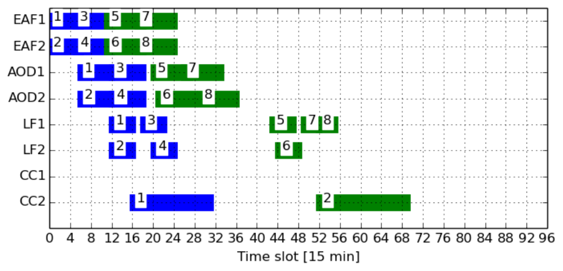
| group | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ |
|---|---|---|---|---|---|---|
| heats | $H_1-H_4$ | $H_5-H_8$ | $H_9-H_{12}$ | $H_{13}-H_{17}$ | $H_{18}-H_{20}$ | $H_{21}-H_{24}$ |

TABLE III. Nominal processing times [min] [19]

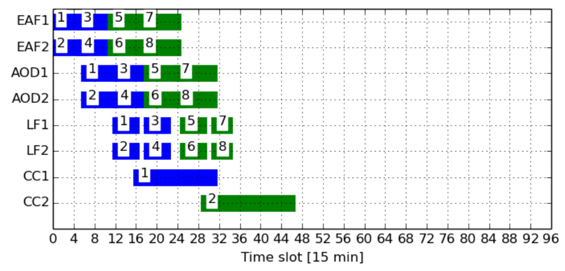
| heats | $EAF_1$ | $EAF_2$ | $AOD_1$ | $AOD_2$ | $LF_1$ | $LF_2$ | $CC_1$ | $CC_2$ |
|---|---|---|---|---|---|---|---|---|
| $H_1-H_4$ | 80 | 80 | 75 | 75 | 35 | 35 | 50 | 50 |
| $H_5-H_6$ | 85 | 85 | 80 | 80 | 40 | 40 | 60 | 60 |
| $H_7-H_8$ | 85 | 85 | 80 | 80 | 20 | 20 | 55 | 55 |
| $H_9-H_{12}$ | 90 | 90 | 95 | 95 | 45 | 45 | 60 | 60 |
| $H_{13}-H_{14}$ | 85 | 85 | 85 | 85 | 25 | 25 | 70 | 70 |
| $H_{15}-H_{16}$ | 85 | 85 | 85 | 85 | 25 | 25 | 75 | 75 |
| $H_{17}$ | 80 | 80 | 85 | 85 | 25 | 25 | 75 | 75 |
| $H_{18}$ | 80 | 80 | 95 | 95 | 45 | 45 | 60 | 60 |
| $H_{19}$ | 80 | 80 | 95 | 95 | 45 | 45 | 70 | 70 |
| $H_{20}$ | 80 | 80 | 95 | 95 | 30 | 30 | 70 | 70 |
| $H_{21}-H_{22}$ | 80 | 80 | 80 | 80 | 30 | 30 | 50 | 50 |
| $H_{23}-H_{24}$ | 80 | 80 | 80 | 80 | 30 | 30 | 50 | 60 |

TABLE IV. Branch and bound results with $t_0$ = 15min

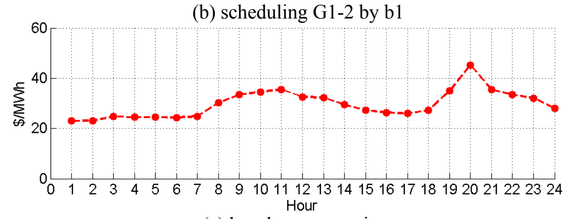
| Groups | | c0 | c1 | b1 |
|---|---|---|---|---|
| G1-2 | Obj(k$) | 24.553 | 24.553 | 24.698 |
| | CPU(s) | 5.8 | 3.7 | 6.2 |
| | lpNum | 2460 | 1985 | 57 |
| G1-3 | Obj(k$) | 39.306 | 39.308 | 39.665 |
| | CPU(s) | 155.4 | 60.7 | 50.0 |
| | lpNum | 9071 | 3835 | 228 |
| G1-4 | Obj(k$) | 57.857 | 57.857 | 58.694 |
| | CPU(s) | 60.4 | 42.7 | 197.8 |
| | lpNum | 3852 | 2745 | 280 |
| G1-5 | Obj(k$) | 69.737 | 69.737 | 70.194 |
| | CPU(s) | 4.3 | 7.2 | 861.0 |
| | lpNum | 0 | 0 | 478 |
| G1-6 | Obj(k$) | 86.352 | 86.352 | 86.799 |
| | CPU(s) | 104.9 | 80.4 | 2737.6 |
| | lpNum | 3698 | 2631 | 725 |



(a) scheduling G1-2 by c0



(b) scheduling G1-2 by b1



(c) hourly energy prices

Figure 6. Scheduling results comparison.

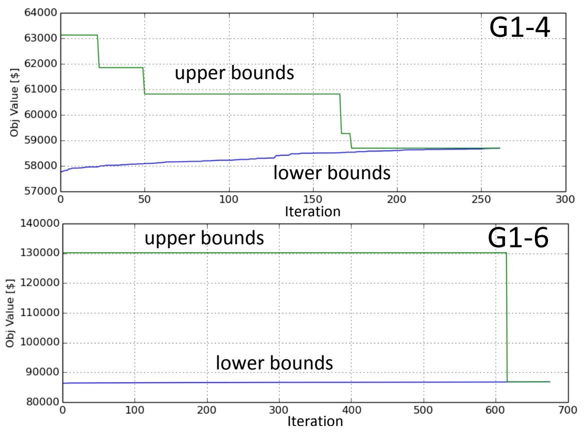


Figure 7. Branch and bound iterations.

into account the relationship among these binary variables in future work, e.g. their process-time sequence relationship. A better rounding procedure could potentially further reduces the iteration number.

## VI. Conclusion

Industrial loads such as steel manufacturing plants can play an important role in integrating more renewable generation into the electric power grid. However, the optimization and scheduling of these industrial plants are usually computationally difficult. In this paper, we focus on the steel plant scheduling problem and propose approaches to improve the related computational issues. As demonstrated through numerical studies, the proposed methods show potentials in reducing the computation time and iteration number of the problem considered. Meanwhile, there still lie possibilities to further improve the computation efficiency, e.g. to apply a better rounding method, which serve as future research directions. The proposed methods are effective in improving the computations, and may play an important role towards developing practical scheduling tools for the steel industry and its demand response.

## References

[1] N. Li, L. Chen, and M. Dahleh, "Demand response using linear supply function bidding," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1827–1838, July 2015.

[2] H. Zhong, L. Xie, and Q. Xia, "Coupon incentive-based demand response: Theory and case study," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1266–1276, 2013.

[3] Q. Huang, M. Roozbehani, and M. Dahleh, "Efficiency-risk tradeoffs in electricity markets with dynamic demand response," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 279–290, Jan 2015.

[4] E. C. Kara, Z. Kolter, M. Berges, B. Krogh, G. Hug, and T. Yuksel, "A moving horizon state estimator in the control of thermostatically controlled loads for demand response," in *IEEE SmartGridComm'13*, 2013, pp. 253–258.

[5] Y. Xu and L. Tong, "On the operation and value of storage in consumer demand response," in *IEEE Annual Conference on Decision and Control (CDC)*, Dec 2014, pp. 205–210.

[6] A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad, "Opportunities and challenges for data center demand response," in *IEEE Green Computing Conference*, 2014.

[7] Z. Liu, I. Liu, S. Low, and A. Wierman, "Pricing data center demand response," in *ACM International Conference on Measurement and Modeling of Computer Systems*, 2014, pp. 111–123.

[8] W. Shi, N. Li, X. Xie, C.-C. Chu, and R. Gadh, "Optimal residential demand response in distribution networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 7, pp. 1441–1450, July 2014.

[9] X. Zhang and G. Hug, "Optimal regulation provision by aluminum smelters," in *IEEE Power and Energy Society General Meeting*, 2014.

[10] ——, "Bidding strategy in energy and spinning reserve markets for aluminum smelters' demand response," in *IEEE Innovative Smart Grid Technologies Conference*, 2015.

[11] Q. Zhang, C. F. Heuberger, I. E. Grossmann, A. Sundaramoorthy, and J. M. Pinto, "Air separation with cryogenic energy storage: optimal scheduling considering electric energy and reserve markets," *AIChE Journal*, 2015.

[12] X. Zhang, G. Hug, Z. Kolter, and I. Harjunkoski, "Industrial demand response by steel plants with spinning reserve provision," in *47th North American Power Symposium*, 2015.

[13] T. Samad and S. Kiliccote, "Smart grid technologies and applications for the industrial sector," *Computers & Chemical Engineering*, vol. 47, pp. 76 – 84, 2012.

[14] D. Fabozzi, N. Thornhill, and B. Pal, "Frequency restoration reserve control scheme with participation of industrial loads," in *PowerTech*, 2013.

[15] S. Elmquist, "Alcoa sees aluminum surplus with lower demand growth," April 2015. [Online]. Available: http://www.bloomberg.com/news/articles/2015-04-08/alcoa-profit-beats-estimates-as-aluminum-demand-climbs

[16] I. Harjunkoski and I. E. Grossmann, "A decomposition approach for the scheduling of a steel plant production," *Computers & Chemical Engineering*, vol. 25, no. 1112, pp. 1647 – 1660, 2001.

[17] A. Haït and C. Artigues, "On electrical load tracking scheduling for a steel plant," *Computers & Chemical Engineering*, vol. 35, no. 12, pp. 3044–3047, 2011.

[18] K. Nolde and M. Morari, "Electrical load tracking scheduling of a steel plant," *Computers & Chemical Engineering*, vol. 34, no. 11, pp. 1899 – 1903, 2010.

[19] P. M. Castro, L. Sun, and I. Harjunkoski, "Resource–task network formulations for industrial demand side management of a steel plant," *Industrial & Engineering Chemistry Research*, vol. 52, no. 36, pp. 13 046–13 058, 2013.

[20] P. M. Castro, I. Harjunkoski, and I. E. Grossmann, "New continuous-time scheduling formulation for continuous plants under variable electricity cost," *Industrial & Engineering Chemistry Research*, vol. 48, no. 14, pp. 6701–6714, 2009.