Computers and Chemical Engineering 173 (2023) 108198

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

Coordination strategy based on hard-heuristics and price-updating scheme for copper smelting process

Hussain Ahmed *, Matti Vilkko

Automation Technology and Mechanical Engineering, Tampere University, Tampere, 33720, Finland

ARTICLE INFO

Keywords:
Copper smelting
Scheduling
Hierarchical framework
Price-based heuristics
Large-scale process
Coordination

ABSTRACT

Optimal scheduling of the copper smelting process is of great concern in the process industry due to the contradictory objectives of the process units and the presence of inter-dependencies between them. This in combination with commercial interests such as maximizing input concentrates to the Flash Smelting Furnace (FSF), continuous operation of the FSF, and production of the blister copper batches with a shorter batch time hinders the development of a scheduling algorithm based on linear optimization techniques. In this study, a hierarchical scheduling framework is developed that finds feasible schedules for the copper smelting process by solving process inter-dependencies using heuristics. For addressing the FSF inter-dependencies, this framework sees the FSF matte as a scarce resource, and the coordinator uses price-based heuristics for the optimal allocation of the matte to various Peirce-Smith converters. Two case studies are presented to demonstrate the effectiveness of the framework.

1. Introduction

In today's competitive world, copper is becoming scarce due to its massive use and demand in various industrial sectors such as the automobile industry, electronic industry, and construction industry (Schipper et al., 2018; IEA, 2022). This fast-growing need creates an imbalance between the copper supply and demand in the open market (International Copper Association, 2017). In addition, the copper industry is facing a decline in high-quality copper ores that increases its production cost and environmental footprint which further complicates the copper supply and demand in the open market (Ahmed et al., 2021, 2022).

To deal with growing copper demand, a relatively simple approach is to use low-quality ores for copper production. However, extracting copper from low-quality ores consumes more energy, which leads to higher greenhouse gas emissions (IEA, 2022). On the other hand, greenhouse gas emissions pose a growing global challenge, which includes stringent regulations that stakeholders are not interested in pursuing. Another option, which is relatively complex and that requires more investment, is to develop state-of-the-art scientific techniques to increase the copper smelters' throughput without compromising on its production costs or environmental footprint (Ahmed et al., 2021).

In the process industry, copper smelters are used to extract copper from its ores (Ahmed et al., 2021, 2022). Copper smelting is a complex industrial process consisting of various units. Among these units, two units that contribute most to the removal of undesirable elements from copper ores are Flash Smelting Furnace (FSF) and Pierce-Smith Converter (PSC) (Davenport et al., 2002). Both units are subjected to various local logistical and operational constraints; therefore, they are usually monitored and scheduled independently by experienced technical personnel (Ahmed et al., 2022). However, independent monitoring and scheduling of these units without a meaningful coordination mechanism ignores the inter-dependencies between these units, thus reducing the process efficiency by moving it from an optimal to a sub-optimal operational point. A sub-optimal operation means reducing the throughput of the copper smelting process, increasing the environmental footprint, and further exacerbating the imbalance between copper supply and demand. Hence, the copper industry requires new operating and scheduling techniques that can provide better coordination between the process units and operate the process at the optimal operating point.

For designing a coordination mechanism for the copper smelting process, two approaches are generally used. In the first approach, a coordination parameter is selected which is used by the coordination mechanism to resolve inter-dependencies. Selection of the coordination parameter typically requires a deep understanding of the process. For a large-scale process, such as the copper smelting process, this approach could be challenging because of the complex nature of the process, and the associated complications for the framework formulation. The second approach is to make use of heuristics. Heuristics are usually application-specific that rely on industrial practices, the experience of

* Corresponding author.
E-mail addresses: hussain.ahmed@tuni.fi (H. Ahmed), matti.vilkko@tuni.fi (M. Vilkko).

https://doi.org/10.1016/j.compchemeng.2023.108198
Received 12 August 2022; Received in revised form 12 February 2023; Accepted 19 February 2023
Available online 1 March 2023

Nomenclature

FSF

Set:

Process scheduling horizon h (min), $h \in \{1, 2, 3, \dots, H\} \subset \mathbb{Z}^+$, H is the maximum value of process scheduling horizon

Variables:

$feed^h_{FSF}$ (kg/min)	input concentrate feed rate at time h
$prod^h_{FSF}$ (kg/min)	FSF matte production at time h
$mass^h_{FSF}$ (kg)	mass of matte in the FSF at time h

Parameters:

mg (percentage)	matte grade
$feed_{max}$ (percentage)	maximum feed rate of input concentrates
$feed_{min}$ (percentage)	minimum feed rate of input concentrates
UL^h_{FSF} (kg)	FSF upper capacity limit at time h
LL^h_{FSF} (kg)	FSF lower capacity limit at time h
$mass^{h=0}_{FSF}$ (kg)	FSF initial inventory level
$Grad1$	gradient of the selected matte grade versus FSF matte production
$Grad2$	gradient of selected matte grade versus FSF feed rate
Y_{inter} (kg/min)	y-intercept of the matte grade versus FSF production and matte grade versus feed rate trajectories

PSC

Sets:

Batch problem horizon t (min), $t \in \{1, 2, 3, \dots, T\} \subset \mathbb{Z}^+$ *and* $T \ll H$, T is the maximum value of batch problem scheduling horizon

PSC unit number n, $n \in \{1, 2, 3, \dots, N\} \subset \mathbb{Z}^+$, N is the total number of PSC units employed

PSC batch number b, $b \in \{1, 2, 3, \dots, B\} \subset \mathbb{Z}^+$, B is the total number of batches per PSC unit

PSC operation $z_i \in Z = \{loading_i, slag_{blow_i}, slag_{skim_i}, copper_{blow}, batch_{end}\}$ where $i \in \{1, 2, 3, \dots, I\} \subset \mathbb{Z}^+$, and I is the total number of repeated operations

Variables:

$$b^t_{z_i} = \begin{cases} 1 & \text{operation } z_i \text{ is processed at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$s^t_{z_i} = \begin{cases} 1 & \text{operation } b^{n,b,t'}_{z_i} \text{ is completed at time } t' \quad \forall\, t \geq t' \\ 0 & \forall\, t < t' \end{cases}$$

$$idle^t = \begin{cases} 1 & b^t_{z_i} = 1 \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

$mass^t_m$ (kg)	mass of matte in PSC unit at time t
$mass^t_{ele}$ (kg)	content of unwanted element ele in the PSC matte at time t. This ele can be iron (Fe) or sulphur (S)
$mass^t_{batch}$ (kg)	mass of the matte required by batch problem at time t
$mass^t_{\mathrm{Cu}}$ (kg)	minimum copper loss point at time t. These points are calculated using the scheme presented in our previous work (Ahmed et al., 2021).

Parameters:

Parameters:

$\mathrm{Cu}_{loss_{z_i}}$ (kg/min)	copper loss rate during operation z_i
r_{ele} (kg/min)	oxidation rate of unwanted element ele
n_{z_i}	position number of operation z_i in set Z
$proc_{max_{z_i}}$ (min)	maximum processing duration of operation z_i
$proc_{min_{z_i}}$ (min)	minimum processing duration of operation z_i
$proc_{fix_{z_i}}$ (min)	fixed processing duration of operation z_i
$matte_{fix}$ (kg)	fixed amount of matte transferred from FSF to a PSC unit
B^t_{load} (min)	no loading operation at time t
B^t_{blow} (min)	no blow operation at time t
PL^t_{FSF}	price value for theFSF lower limit violation at time t
MDO^t_{PSC} (kg)	total demand of the FSF matte by batch problems on other PSC units at time t
LL^t_{FSF} (min)	FSF lower capacity limit at time t

$$B^{n,b,t}_{load}(\text{min}) = \begin{cases} t & loading \text{ time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

$$B^{n,b,t}_{blow}(\text{min}) = \begin{cases} t & slag \text{ or } copper \text{ blow time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

$$B^{n,b}_{start}(\text{min}) = \begin{cases} t & \text{start time of batch problem on PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

$$B^{n,b}_{end}(\text{min}) = \begin{cases} t & end \text{ time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

$$MD^{n,b,t}_{batch}(\text{kg}) = \begin{cases} \mathbb{Z}^+ & \text{matte demand of PSC unit } n \text{ during batch } b \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$B^{n,b}_{load_1}(\text{min}) = \begin{cases} t & \text{first } loading \text{ time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

$$B^{n,b}_{load_I}(\text{min}) = \begin{cases} t & \text{last } loading \text{ time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

Coordinator

Set:

Iteration number k, $k \in \{1, 2, 3, \dots, K\} \subset \mathbb{Z}^+$, K is the maximum number of iterations

Parameters:

$start^n_{PSC}$ (min)	starting time of PSC unit n
$prior^n_{PSC}$	priority of PSC unit n
$step_{size}$ (percentage)	part of a slot that will be considered to resolve FSF inter-dependencies
$slot_{size}$	total time instances in the selected slot
FSF_{count}	number of time instances in $slot_{size}$ that will be handled during a single iteration
$slot_{min}$	minimum number of time instances in a slot
UT^h_{FSF} (min)	FSF upper limit violation exists at time h
LL^h_{FSF} (min)	FSF lower limit violation exists at time h

process personnel, and process history. In addition, they are open to modifications and can provide a plant-wide solution with reasonable computational demands (Ahmed et al., 2022).

In a large-scale process with limited shared resources, inter-dependencies are generated among the process units due to the non-optimal distribution of shared resources to the process units.

PU^{h}_{FSF}	price value for FSF upper capacity violation at time h
PL^{h}_{FSF}	price value for FSF lower capacity violation at time h
MD^{h}_{PSC} (kg)	total demand of the FSF matte by all PSC units demand at time h
MDO^{h}_{PSC} (kg)	total demand of the FSF matte by batch problems on other PSC units at time h
ULV^{h}_{FSF} (min)	FSF upper capacity limit exist at time h
LLV^{h}_{FSF} (min)	FSF lower capacity limit exist at time h
UD^{h}_{FSF} (kg)	matte surplus value at time h
$UD^{k,h}_{FSF_{past}}$ (kg)	Matte surplus value at time h during iteration k
LT_{FSF} (min)	foremost lower capacity limit violation's time value
LD_{FSF} (kg)	matte scarcity value at time LT_{FSF}
$LD^{k,h}_{FSF_{past}}$ (kg)	matte scarcity value at time h during iteration k
$K_{P_{UD}}$	PI controller proportional gain value that solves FSF upper capacity limit violations
$K_{I_{UD}}$	PI controller integral gain value that solves FSF upper capacity limit violations
$K_{P_{LD}}$	PI controller proportional gain value that solves FSF lower capacity limit violations
$K_{I_{LD}}$	PI controller integral gain value that solves FSF lower capacity limit violations

$$batch^{n,b}_{FSF} = \begin{cases} 1 & \text{if FSF lower limit capacity violation exists on PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$$

$$PSC^{n,h}_{load} = \begin{cases} 1 & \textit{loading} \text{ is made to PSC unit } n \text{ at time } h \\ 0 & \text{otherwise} \end{cases}$$

$$PSC^{n,h}_{blow} = \begin{cases} 1 & \textit{copper} \text{ or } \textit{slag} \text{ blow is made to PSC unit } n \text{ at time } h \\ 0 & \text{otherwise} \end{cases}$$

$$Load^{h}_{conf} = \begin{cases} 1 & \text{logistical inter-dependency exists at time } h \\ 0 & \text{otherwise} \end{cases}$$

$$Flow^{h}_{conf} = \begin{cases} 1 & \text{flow inter-dependency exists at time } h \\ 0 & \text{otherwise} \end{cases}$$

$$is_{PSC} = \begin{cases} 1 & \text{PSC inter-dependencies exist in the schedule} \\ 0 & \text{otherwise} \end{cases}$$

$$is_{FSF} = \begin{cases} 1 & \text{FSF inter-dependencies exist in the schedule} \\ 0 & \text{otherwise} \end{cases}$$

If resources are not shared optimally among process units, one or more units may become a bottleneck for the entire process, which reduces process throughput and overall efficiency. Therefore, all the shared resources should be distributed optimally among the process units for profit maximization. Possible solutions for optimal resource distribution could be a centralized scheduling approach or a hierarchical approach based on hard heuristics. Both these approaches have shortcomings considering the exceptionally high computational costs and the rigorous scheduling of process units (Cheng et al., 2004a; Rokhforoz and Fink, 2021; Gao et al., 2018; Ahmed et al., 2022). Another solution is to use a heuristics-based coordination mechanism where prices are used for the optimal allocation of shared resources among process units (Ruben et al., 2013; Lavios Villahoz et al., 2010; Martí et al., 2013).

In the current study, a heuristics-based coordination mechanism is proposed for the copper smelting process that resolves all process inter-dependencies that may arise during the process operation. The proposed coordination mechanism operates the FSF at optimal operating point by finding a balance between FSF matte's supply and demand. This study is an extension of our previous work, which resolves the process inter-dependencies using hard heuristics (Ahmed et al., 2022). In the present work, inter-dependencies generated by the PSC units are solved in the same way as in the previous work, while a new price-based coordination mechanism is proposed to resolve the FSF inter-dependencies. We show that the price-based coordinator has the potential to solve the process inter-dependencies by allowing the units to make their own decisions rather than the coordinator dictating its choices, as presented in our previous study (Ahmed et al., 2022). The objective is to design a hierarchical scheduling framework using discrete-time mixed integer linear programming (MILP) techniques. It produces a schedule for multiple-PSC units and a multiple-batch process while maximizing the FSF throughput, satisfying the FSF capacity limit constraints, minimizing the copper losses during the PSC operation, and resolving the scheduling inter-dependencies. Unlike the rigorous scheduling approach, the proposed price-based coordination can enable process personnel to operate a processing unit independently, foresee the consequences of their operational decisions, and improve their process understanding without in-depth knowledge of the entire process.

The remainder of the paper is organized as follows. In Section 2, scheduling frameworks are briefly discussed, and the scheduling literature is presented. Section 3 provides a summary of the copper smelting process operation, and Section 4 highlights the problem statement. The heart of this work is presented in Section 5 which elaborates on the formulation of the hierarchical framework and the coordination mechanism. Section 6 describes two case studies to show that the hierarchical framework with price-based coordination has the potential to operate the copper smelting process at the optimal operation point. Section 7 presents the conclusion and offers an outlook for future work.

2. Scheduling framework

For designing a scheduling framework for large-scale industrial processes, such as the copper smelting process, two common approaches are centralized and hierarchical. The centralized approach is built around the idea to gather all information in one place and then design a single large-scale scheduling framework for the entire process. This approach is suitable for small-scale processes that comprise a few units. As copper smelters are becoming increasingly complex, this approach is not ideal for designing a scheduling framework for this process (Ahmed et al., 2022; Kelly and Zyngier, 2008; Cheng et al., 2007; Martí et al., 2013; Christodoulopoulos et al., 2009a; Moroşan et al., 2010). Moreover, the centralized approach is vulnerable to single-point failure, requires high computational demands, has scalability issues, and is hard to repair in case of a failure (Christodoulopoulos et al., 2009b).

For designing a scheduling framework for a large-scale process, the most popular choice is to use the hierarchical approach, which has been applied in various industrial processes (Ahmed et al., 2022; Cheng et al., 2006, 2007, 2004b; Popa, 2014; Harjunkoski and Grossmann, 2001). In this approach, a large-scale process is decomposed into two layers: a lower layer and an upper layer. The lower layer consists of unit-level scheduling problems, which are scheduled and operated independently. The upper layer is made of a coordinator that enables the communication between the process units thus, enabling the framework to achieve an optimal operation of the process. A generic flow diagram of the hierarchical approach is shown in Fig. 1. The performance of a hierarchical framework usually depends on the coordination mechanism. If promising heuristics, such as price-based heuristics, are used for the coordination mechanism, it will enable the coordinator to resolve

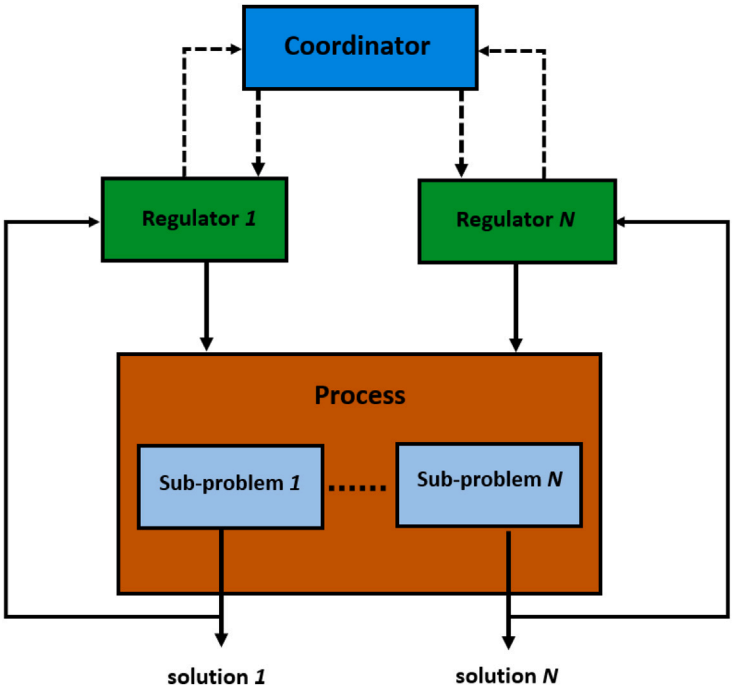


Fig. 1. Hierarchical framework architecture.

process inter-dependencies effectively and return a solution with high quality and lower computational demands.

Price-based heuristics in the coordination mechanism have been used in various industrial processes. Jose and Ungar (2000) used price-based heuristics for the optimal distribution of inter-process stream between two units to maximize profit by maintaining an equilibrium between the steam supply and demand. Here, the problem is decomposed into two sub-problems, and prices are used to achieve an optimal solution. Voos (2007) introduced an interesting distributed agent-based algorithm for the dynamic resources' allocation in a continuous production process where price-based heuristics are used for the optimal allocation of resources. The solution is applied in a real-time industrial process to demonstrate its novelty. Martí et al. (2013) presented another price-based coordination mechanism for the optimal distribution of a scarce resource among the process units. Here, the problem is divided into multiple sub-problems, and price-based coordination is formulated using a feedback control law. The results showed that the distributed price-based coordination improved the operation of the process and provided a near-centralized solution.

For the copper smelting process, various scheduling solutions have been presented. Harjunkoski et al. (2006) presented a continuous-time centralized scheduling formulation for the copper smelting process. This framework uses MILP techniques, where the objective is to maximize the process throughput. Suominen and Mörsky (2016) introduced an interesting scheduling framework for the copper smelting process that consists of a single FSF and three PSC units. This framework, which is based on continuous-time MILP techniques, maximizes the smelting throughput. This formulation is applied in a real-time smelter to show the novelty of the framework. Pradenas et al. (2003) proposed another framework for the copper smelting process where the objective is to maximize smelter production. This framework generated schedules for a process that has numerous operational, metallurgical, and environmental constraints. The framework is applied to a copper smelter to show its novelty.

In our previous work (Ahmed et al., 2022), we developed discrete-time MILP centralized and hierarchical scheduling frameworks that find optimal schedules for a smelting process that consists of one FSF and multiple PSC units. A coordination scheme based on rigorous practices is proposed to solve scheduling inter-dependencies among the process units and finds a near-optimal schedule. The centralized and hierarchical frameworks are compared in terms of quality and computational demands.

3. Process description

A generic flow diagram of the copper smelting process is shown in Fig. 2. The input concentrates are fed to the FSF first, where oxygen passes through the input concentrates to produce copper-enriched molten matte, slag, and gases (Liu et al., 2014; Davenport and Partelpoeg, 2015). The slag is transferred to the slag processing unit, while matte is moved to the PSC units during repeated loading operations. This matte has a specific matte grade, which is generally defined by the process personnel beforehand.

In PSC, oxygen passes through the matte in repeated slag blowing operations to oxidize iron and sulphur in the matte to slag and gases. The slag is removed after each slag blowing operation. After the last slag blow, a single long copper blow begins that oxidizes the remaining sulphur from the matte. Gases produced during the FSF and PSC operation are transferred to the acid production unit using the installed pipelines.

During slag blowing operations, copper is lost to the slag. This copper loss becomes exponential as the iron content in matte reaches the zero level (Tan, 2007; Ahmed et al., 2021). Therefore, it is essential to determine the optimal duration of slag blowing operations as they affect the total copper losses during the process operation. The last slag and copper blows continue until the maximum amount of iron and sulphur oxidize from the matte. The final product is generally referred to as blister copper (≈ 98 percent). Details about the PSC operation can be found in Ahmed et al. (2021).

4. Problem statement

This study considers a relatively simple copper process, as shown in Fig. 3. Although the actual copper smelting process has more complex dynamics, this formulation still reflects the main aspects of the copper smelting process.

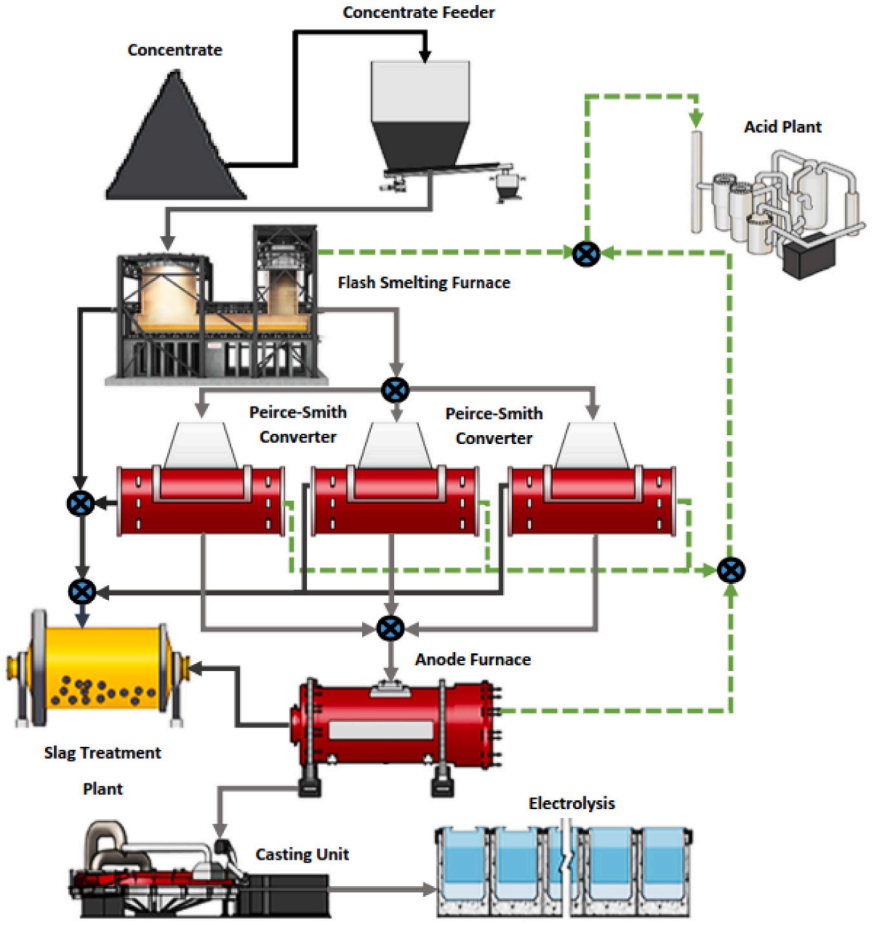


Fig. 2. Copper smelting process. *Source:* Adapted from Ahmed et al. (2022).

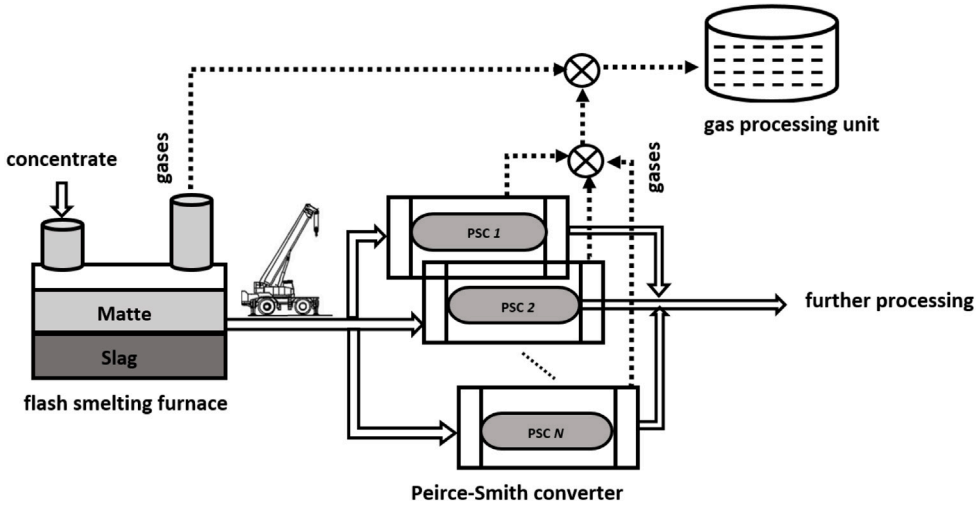


Fig. 3. Simplified copper smelting process.

Given: A copper smelting process comprising one FSF and several PSC units. The FSF receives input concentrates and produces matte, which is then processed by the PSC units. The FSF has limited storage capacity, and it is operated continuously without any break. Each PSC unit produces multiple blister copper batches using a predefined scheduling recipe. All units are scheduled and operated independently.

Goal: The goal is to design a coordination scheme for the hierarchical framework where the framework solves inter-dependencies between the PSC units without compromising on the FSF throughput and to produce blister copper batches in a synchronized manner. The proposed framework solves inter-dependencies between PSC units in the same way as in our previous work (Ahmed et al., 2022), where a unit's operation is stopped deliberately to resolve inter-dependencies. As the FSF operates continuously, halting its operation may result in serious consequences. Therefore, this framework uses price-based coordination to solve FSF inter-dependencies. This new price-based coordination will

enable the FSF to decide on its operation without getting dictation from the coordinator. The motivation is to show that price-based coordination has the potential to offer a near-optimal solution, and it can enable process personnel to operate a processing unit without in-depth knowledge of the entire process.

In this study, both the FSF and PSC units generate inter-dependencies, which are discussed in turn.

4.1. PSC inter-dependencies

In the copper smelting process, multiple PSC units are functioning in parallel. Matte is loaded to these PSC units using a crane that is installed in the vicinity. This crane can transfer only a fixed quantity of matte from the FSF to a PSC unit in a unit of time. As all the PSC units are functioning independently, PSC units will likely request matte loading at the same time, which generates logistical inter-dependencies among PSC units. In addition to that, PSC units are assigned priorities based on their availability. For producing blister copper batches in a synchronized manner, loading operations to the high-priority PSC unit must be executed first, followed by loading operations to the next succeeding high-priority PSC unit, and so on. As PSC units may request matte loading at any time, logistical inter-dependencies are generated when a low-priority PSC unit demands matte loading before the preceding high-priority PSC units.

In the copper smelting process, the FSF and PSC units produce gases. These gases are transferred to the gas processing unit using the installed gas pipelines, which have limited transfer capacity. To respect the gas-pipelines capacity constraint and consider that stopping the FSF will have serious consequences, PSC units must be scheduled so that only one PSC unit is in slag blow or copper blow at any time. Hence, flow inter-dependencies are generated among the PSC units if two or more units are simultaneously in the slag or copper blow operations.

4.2. FSF inter-dependencies:

FSF inter-dependencies arise due to its upper and lower capacity limit violation. In the copper smelting process, if the PSC units do not process the matte at the same rate as that produced by the FSF, the PSC units become the bottleneck for the entire process; consequently, the matte in the FSF surpasses its maximum storage limit. Likewise, when the PSC units' matte demand is high, but the FSF does not meet this demand due to a high matte grade selection or low concentrate feed rate to the FSF, the FSF becomes the bottleneck for the entire process. Thus, the FSF lower limit will reach that generates FSF inter-dependencies.

4.3. Assumptions

The copper smelting process has complex thermodynamics (Davenport and Partelpoeg, 2015). Since the objective is not to study the thermodynamic dynamics of the process, several assumptions are made in this study. They are outlined below.

- The FSF processes the same type of concentrates. This study does not focus on a specific type of FSF; therefore, FSF storage limits are selected arbitrarily. There is always initial inventory available at the beginning of the process. It is assumed that the FSF never shut down, and any such attempt will have serious consequences. The FSF matte grade is selected beforehand and stays unchanged.
- All the PSC units are the same in dimensions, and they do not require any maintenance break. Furthermore, all the PSC units follow the same scheduling recipe to produce blister copper batches. This scheduling recipe is defined by the process personnel beforehand. The slag type during PSC operation remains the same.
- Based on the PSC units' priority, the highest priority unit finishes loading operations first, while the lowest priority PSC unit executes the loading operations lastly. If PSC inter-dependencies exist in the schedule, the framework always re-schedules the batch on the lowest priority unit among the conflicting PSC units to resolve PSC inter-dependencies.
- The oxygen affects the oxidation rate of the elements in matte and the temperature in the FSF and PSC units. As this work does not focus on the thermodynamics of the process, a constant oxygen supply is considered here. Thus, oxygen does not affect the oxidation rate of elements, and it is not a deciding factor.
- The temperature inside the FSF and PSC units is maintained within a feasible range by pre-selecting a constant oxygen supply.
- Each slag-blowing operation must produce a minimum amount of slag, which is accomplished by setting a minimum duration constraint for all slag blowing operations.

5. Hierarchical framework

5.1. FSF model

The FSF produces matte continuously, and its production rate depends on the matte grade mg and the feed rate $feed^h_{FSF}$, as given in Eqs. (1)–(2). Increasing the $feed^h_{FSF}$ increases matte production as concentrates are available at a faster rate for processing. However, the production rate decreases with the increase in the mg as the FSF requires more time to oxidize the unwanted elements from the input concentrates. The objective function of the FSF is given in Eq. (3). As the matte grade is decided beforehand that remains unchanged during the process operation, maximizing the feed rate is the obvious objective.

During each iteration, the FSF receives the price vector PU^h_{FSF}, total demand of matte MD^h_{PSC}, FSF upper capacity limit violating time instances UT^h_{FSF}, and the FSF maximum storage capacity limit UL^h_{FSF}. At the beginning of the simulation, the framework initializes the UT^h_{FSF} with all the scheduling time instances ($UT^h_{FSF} = h$) and solves the FSF model. If FSF upper capacity limit violations exist in the schedule during an iteration k, the framework solves the FSF model during the next iteration $k+1$ only for the time instances, which are stored in UT^h_{FSF}. However, if no FSF upper limit violations exist in the schedule during an iteration k, the framework skips the FSF model and uses the latest available solution. The FSF model calculates the FSF mass and input feed trajectories and sends them to the coordinator. Thereafter, it is awaiting further instructions from the coordinator.

$$prod^h_{FSF} = (Grad1 \times mg) + (Grad2 \times feed^h_{FSF} + Y_{inter}) \quad \forall Grad1, Grad2, Y_{inter} \in R^+ \tag{1}$$

$$mass^h_{FSF} = mass^{h=0}_{FSF} + mass^{h-1}_{FSF} + prod^h_{FSF} \quad \forall\ h \in UT^h_{FSF} \tag{2}$$

$$\max_{feed} \sum_{h=1}^{H} feed^h_{FSF} + \sum_{h=UT^h_{FSF}}^{H} PU^h_{FSF} \times (MD^h_{PSC} + UL^h_{FSF} - mass^h_{FSF}) \tag{3}$$

$$feed_{min} \le feed^h_{FSF} \le feed_{max}$$

5.2. Blister copper batch model

The proposed hierarchical framework simulates and schedules all the PSC units independently. Therefore, it solves a maximum number of $N \times B$ independent scheduling problems during each iteration. For ease, the blister copper batch will be referred to as the *batch problem*.

Batch problems receive two sets of information from the coordinator during each iteration. The first set contains the batch problems start time $B^{n,b}_{start}$, time instances of logistical inter-dependencies $Load^h_{conf}$, and time instances of flow inter-dependencies $Flow^h_{conf}$. The second set consists of price vector PL^h_{FSF}, total matte demand by batch problems on other PSC units MDO^h_{PSC}, and FSF lower capacity limit LL^h_{FSF}. The coordinator uses the first set to address PSC inter-dependencies, while the second set is used to resolve FSF lower limit violations. The coordinator calculated this information during the previous iteration.

The batch problem uses $B^{n,b}_{start}$ to get the conflicting time instances from the $Load^h_{conf}$ and $Flow^h_{conf}$, and store them in B^t_{load} and B^t_{blow} by assigning a value of 1 against each violating time instant, as given in Eqs. (4) and (5). To ensure that the batch problem executes only

one operation at a time, it uses Eq. (6). The duration of operations is either defined by the process personnel, or its optimal values are calculated by the model as given in Eqs. (7)–(8). For handling the PSC inter-dependencies, the batch problem uses Eq. (9) to prevent the batch problem from executing loading or blowing operations at conflicting times.

$$B^t_{load} = \begin{cases} 1 & t = h - B^{n,b}_{start} \text{ and } Load^h_{conf} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$B^t_{blow} = \begin{cases} 1 & t = h - B^{n,b}_{start} \text{ and } Flow^h_{conf} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\sum_{\substack{z_i \in \\ operation}} b^t_{z_i} \le 1 \tag{6}$$

$$\sum_{t=1}^{T} b^t_{z_i} = proc_{fix_{z_i}} \qquad \forall z_i = loading_i,\ slag_{skim_i} \tag{7}$$

$$proc_{min_{z_i}} \le \sum_{t=1}^{T} b^t_{z_i} \le proc_{max_{z_i}} \qquad \forall z_i = slag_{blow_i},\ copper_{blow} \tag{8}$$

$$\sum_{t=1}^{T} b^t_{z_i} = 0 \qquad \text{if } \left(B^t_{load} = 1 \text{ or } B^t_{blow} = 1\right)$$
$$\forall z_i = loading_i,\ slag_{blow_i},\ copper_{blow} \tag{9}$$

To perform the PSC operations in the same manner as described in the set Z, the batch problem uses a combination of two binary variables. One set of variables, as represented by $b^t_{z_i}$, corresponds to the occurrence of operation z_i at time t. The other set of variables, as represented by $s^t_{z_i}$ ensure that the batch problem schedules the operations rightly, as given in Eqs. (10)–(11).

$$s^t_{z_i} = s^{t-1}_{z_i} + b^t_{z_i} \tag{10}$$

$$\sum_{\substack{z_i \in \\ operation}} \sum_{t=1}^{T} s^t_{z_i} \ge \left(n_{z_i} \times b^t_{z'_i}\right) \qquad z'_i = \text{operation preceding } z_i \tag{11}$$

The batch problem calculates four different masses: the mass of matte in the PSC $mass^t_m$; the mass of unwanted elements in the PSC $mass^t_{ele}$; the amount of matte demanded by the batch problem $mass^t_{batch}$; and the amount of copper loss to the slag $mass^t_{cu}$. All the masses are represented by continuous variables, which are calculated using Eqs. (12)–(15). For keeping copper losses at a lower level, a simple but efficient mathematical scheme is proposed in our previous work that finds the optimal slag-blowing duration for maintaining a balance between iron in matte and the amount of copper in slag. Details about this scheme can be found in Ahmed et al. (2021). For keeping the batch time shorter, the batch problem uses a penalty term, which is calculated using Eq. (16).

$$mass^t_m = mass^{t-1}_m + \sum_{\substack{z'_i \in \\ operation}} matte_{fix} \times b^t_{z'_i} - \sum_{\substack{z_i \in \\ operation}} (\mathrm{Cu}_{loss_{z_i}} + r_{ele}) \times b^t_{z_i}$$
$$\forall z'_i \in loading_i,\ z_i \in slag_{blow_i},\ copper_{blow} \tag{12}$$

$$mass^t_{elm} = mass^{t-1}_{elm} + \sum_{\substack{z'_i \in \\ operation}} f[(mg)] \times b^t_{z'_i} - \sum_{\substack{z_i \in \\ operation}} (\mathrm{Cu}_{loss_{z_i}} + r_{ele}) \times b^t_{z_i}$$
$$\forall z'_i \in loading_i,\ z_i \in slag_{blow_i},\ copper_{blow} \tag{13}$$

$$mass^t_{batch} = mass^{t-1}_{batch} + \sum_{\substack{z'_i \in \\ operation}} matte_{fix} \times b^t_{z'_i} \qquad \forall z'_i \in loading_i \tag{14}$$

$$mass^t_{\mathrm{Cu}} = mass^{t-1}_{\mathrm{Cu}} + \sum_{\substack{z_i \in \\ operation}} \mathrm{Cu}_{loss_{z_i}} \times b^t_{z_i} \qquad \forall z_i \in slag_{blow_i} \tag{15}$$

$$idle^t = \sum_{\substack{z_i \in \\ operation}} b^t_{z_i} \qquad \forall z_i \in Z \tag{16}$$

For addressing FSF inter-dependencies due to its lower limit violation, the batch problem uses the second set of information. It uses $B^{n,b}_{start}$ to extract pertinent information from the PL^h_{FSF}, MDO^h_{PSC}, and LL^h_{FSF} using Eqs. (17)–(19), and uses this information in the objective function of the batch problem as given in Eqs. (20). The objective of the batch problem is to minimize impurities in the matte, copper losses during the slag blowing, and unnecessary idle time in the schedule, and resolve the FSF inter-dependencies due to its lower limit violation. From the solution, the batch problem calculates the loading times, blow times, batch end time, and the demand for the matte using Eqs. (17)–(26) and shares it with the coordinator.

$$PL^t_{FSF} = \begin{cases} PL^h_{FSF} & t = h - B^{n,b}_{start} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$MDO^t_{PSC} = \begin{cases} MDO^h_{PSC} & t = h - B^{n,b}_{start} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$LL^t_{FSF} = \begin{cases} LL^h_{FSF} & t = h - B^{n,b}_{start} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

$$\min_{mass_{\mathrm{Cu}}, mass_{ele}, \mathrm{Cu}_{ratio}, idle} \sum_{t=1}^{T} mass^t_{\mathrm{Cu}} + \sum_{t=1}^{T} mass^t_{ele} + \sum_{t=1}^{T} \mathrm{Cu}^t_{ratio} + \sum_{t=1}^{T} idle^t + \sum_{t=LT_{FSF}-B^{n,b}_{start}}^{T} PL^t_{FSF} \times \left(MDO^t_{PSC} + mass^t_{batch} - LL^t_{FSF}\right) \tag{20}$$

$$B^{n,b,t}_{load} = \begin{cases} t & b^{n,b,t}_{z_i} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall z_i = loading_i \tag{21}$$

$$B^{n,b,t}_{blow} = \begin{cases} t & b^{n,b,t}_{z_i} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall z_i = slag_{blow_i},\ copper_{blow} \tag{22}$$

$$B^{n,b}_{end} = \begin{cases} t + B^{n,b}_{start} & b^{n,b,t}_{z_i} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall z_i = batch_{end} \tag{23}$$

$$B^{n,b}_{load_1} = \begin{cases} t & b^{n,b,t}_{z_i} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall z_i = loading_1 \tag{24}$$

$$B^{n,b}_{load_I} = \begin{cases} t & b^{n,b,t}_{z_i} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall z_i = loading_I \tag{25}$$

$$MD^{n,b,t}_{batch} = \begin{cases} mass^t_{bt} & \\ 0 & \text{otherwise} \end{cases} \qquad \forall z_i = loading_i \tag{26}$$

5.3. Coordinator

The schematic diagram of the hierarchical framework is shown in Fig. 4. In this framework, the job of the coordinator is to coordinate between the FSF and batch problems to provide a feasible schedule. Therefore, the coordinator is responsible for the optimal operation of the FSF operation considering its storage limits and resolving inter-dependencies between the batch problems.

To attain a feasible schedule, the coordinator will use a price-based coordination mechanism, which is the core of this study. In contrast to our earlier work (Ahmed et al., 2022), this study focuses on finding a coordination mechanism that allows the units to make their own decisions to resolve process inter-dependencies. The framework will solve the PSC inter-dependencies using rigorous heuristics, while for resolving FSF inter-dependencies, it uses the price adjustment technique based on market economics theory.

According to market economics, demand and supply of a quantity determine how buyers and sellers interact to determine the transaction prices for some scarce resources (Eastin and Arbogast, 2011). In an ideal market scenario, the demand and supply of a quantity should be equal. However, in many real-time situations, the supply and demand of scarce resources keep fluctuating; therefore, their prices vary from

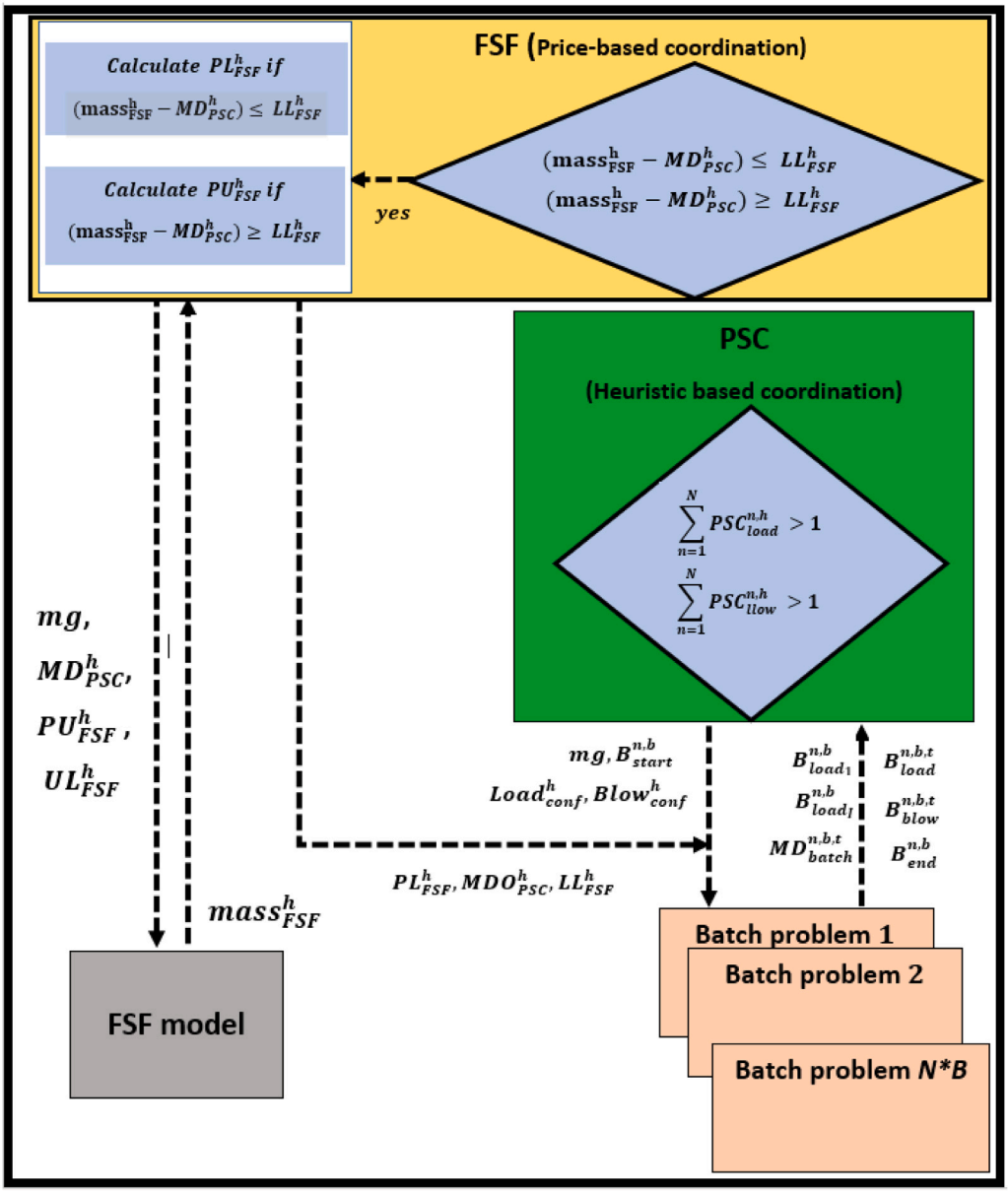


Fig. 4. Hierarchical framework.

time to time. In economic theory, if supply exceeds the demand for a resource, market forces act to lower its price to increase its demand in the market. On the other hand, if demand exceeds supply, market forces will raise its price to decrease the demand.

In this study, it is assumed that a limited quantity of matte is available during the process operation. Thus, the FSF matte is the scarce resource here, and price-based coordination is used to find the price trajectory that will ensure a balance between its supply and demand.

In this framework, PSC units' availability is known in advance using the parameter PSC^n_{start} as shown in Fig. 5. Based on availability, the coordinator assigns a priority number to each PSC unit. The PSC unit that is available earliest is assigned the highest priority number, and the last available unit will have the lowest priority number, as given in Eq. (27). This priority number determines the order in which the PSC units will execute their operations.

$$prior^n_{PSC} = \begin{cases} highest & \text{unit } n \text{ has the lowest } start^n_{PSC} \text{ value} \\ highest - 1 & \text{unit } n \text{ has the second lowest } start^n_{PSC} \text{ value} \\ \vdots & \\ lowest & \text{unit } n \text{ has the highest } start^n_{PSC} \text{ value} \end{cases} \quad (27)$$

To find inter-dependencies in the schedule, the coordinator collects the information from the FSF model and batch problems. The coordinator formulates the schedule and examines it in ascending order of the scheduling horizon. The handling of PSC and FSF inter-dependencies is discussed next.

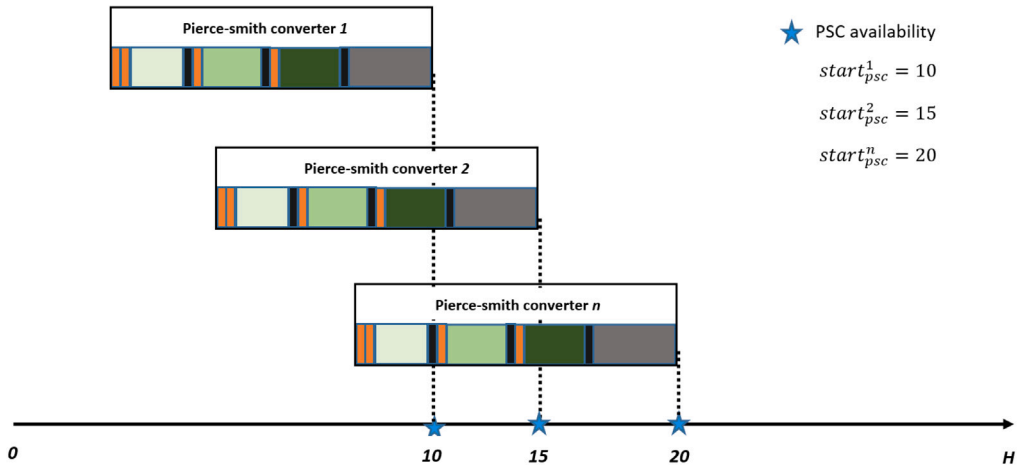


Fig. 5. PSC units availability.
Source: Adapted from Ahmed et al. (2022)

5.3.1. PSC inter-dependencies

During each iteration, batch problems solve their scheduling problems and return the required information to the coordinator, as discussed in Section 5.2. As batch problems begin their operations at different times, the coordinator arranges the batch problems' loading and blow times in ascending order of their start time, as given in Eqs. (28) and (29). The updated loading and blowing times are stored in $PSC^{n,h}_{load}$ and $PSC^{n,h}_{blow}$ by assigning a value of 1 to each active time instance.

$$PSC^{n,h}_{load} = \begin{cases} 1 & h = (B^{n,b}_{start} + B^{n,b,t}_{load}) \quad \forall B^{n,b,t}_{load} \neq 0 \\ PSC^{n,h-1}_{load} & h \leq (B^{n,b}_{start} + B^{n,b}_{load_I}) \quad \forall B^{n,b,t}_{load} = 0 \end{cases} \tag{28}$$

$$PSC^{n,h}_{blow} = \begin{cases} 1 & h = (B^{n,b}_{start} + B^{n,b,t}_{blow}) \quad \forall B^{n,b,t}_{blow} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

Using Eqs. (30) and (31), the coordinator finds the timing of logistical and flow inter-dependencies and stores them in $Load^h_{conf}$ and $Flow^h_{conf}$. Furthermore, it calculates the new start time of the batch problems using Eq. (32).

$$Load^h_{conf} = \begin{cases} 1 & \sum_{n=1}^{N} PSC^{n,h}_{load} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

$$Flow^h_{conf} = \begin{cases} 1 & \sum_{n=1}^{N} PSC^{n,h}_{blow} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

$$B^{n,b}_{start} = B^{n,b-1}_{end} + 1 \tag{32}$$

The coordinator determines if PSC inter-dependencies exist in the schedule using Eq. (33). If PSC inter-dependencies are found, the coordinator sets the is_{PSC} to value 1 using Eq. (33) and shares the $B^{n,b}_{start}$, $Load^h_{conf}$ and $Flow^h_{conf}$ with all the batch problems, and a new iteration is performed. Following that, the coordinator resets the value of is_{PSC} back to value 0. The batch problems solve their schedules and return the corresponding information to the coordinator. If PSC inter-dependencies still exist, the coordinator performs the same steps and performs another iteration. This exchange of information between the coordinator and batch problems continues until the coordinator finds a schedule free of PSC inter-dependencies.

$$is_{PSC} = \begin{cases} 1 & \sum_{h=1}^{H} Load^h_{conf} \neq 0 \text{ or } \sum_{h=1}^{H} Flow^h_{conf} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

5.3.2. FSF inter-dependencies

To determine if FSF inter-dependencies exist in the schedule, the coordinator uses Eq. (34) to calculate the total matte demanded by all the batch problems. Then, it uses Eqs. (35) and (36) to determine the time instances at which the FSF upper or lower capacity limit is violated. If FSF inter-dependencies exist in the schedule, the coordinator sets the is_{FSF} to value 1, as given in Eq. (37).

$$MD^h_{PSC} = \begin{cases} \sum_{n=1}^{N} \sum_{b=1}^{B} MD^{n,b,t}_{batch} & h = (B^{n,b}_{start} + t) \\ MD^{h-1}_{PSC} & \text{otherwise} \end{cases} \tag{34}$$

$$ULV^h_{FSF} = \begin{cases} h & (mass^h_{FSF} - MD^h_{PSC}) \geq UL^h_{FSF} \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

$$LLV^h_{FSF} = \begin{cases} h & (mass^h_{FSF} - MD^h_{PSC}) \leq LL^h_{FSF} \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

$$is_{FSF} = \begin{cases} 1 & \sum_{h=1}^{H} ULV^h_{FSF} \neq 0 \text{ or } \sum_{h=1}^{H} LLV^h_{FSF} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

FSF upper capacity limit violations

The FSF upper capacity limit violations can exist in the schedule at different times; therefore, this framework divides all the violating time instances into different slots. Every slot comprises one or more conflicting time instances such that each slot has consecutive conflicting time instances, as shown in Fig. 6. If the coordinator attempts to address all slots simultaneously, the framework will likely start bouncing between two local operating points, and it will not return a feasible solution.

To ensure that the framework always returns a feasible schedule, the coordinator solves only the foremost slot, as shown in Fig. 6. Furthermore, if the number of time instances in the selected slot is high, considering the entire slot during one iteration means decreasing the FSF feed rate for all the time instances in the given slot. As the FSF behaves as an integrator for the matte storage, it is highly possible that decreasing the FSF feed rate only a few time instances in the slot could resolve the FSF upper capacity limit violations for the entire slot. For example, if a selected time slot consists of multiple conflicting time instances, addressing only 10 percent of time instances in it could resolve all the violations, or at a minimum, it will reduce the total number of conflicting time instances in the selected time slot. Therefore, this

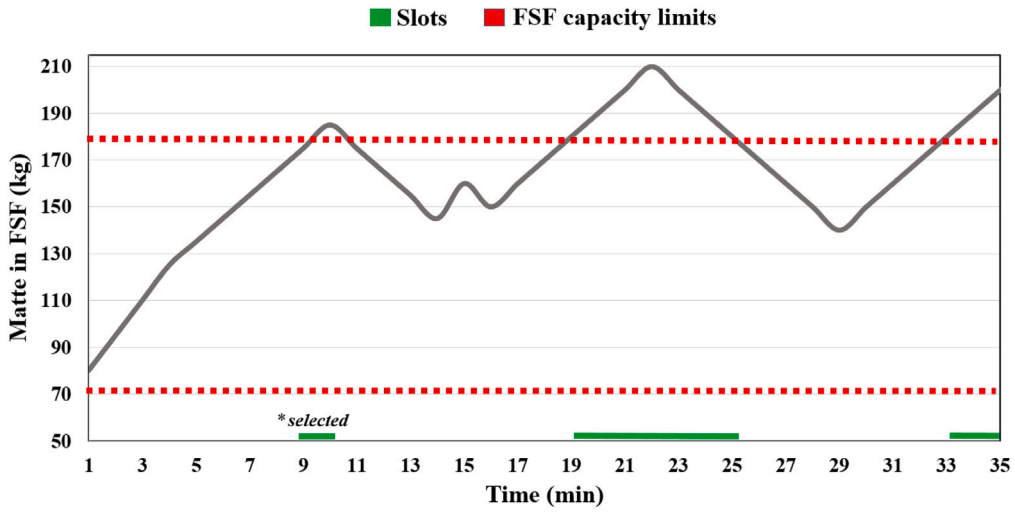


Fig. 6. Division of the FSF upper capacity limit violations.

study considers only part of a slot during a single iteration. Such a practice allows the coordinator to solve FSF inter-dependencies without compromising on the FSF objective; hence, operating the FSF at the optimal operating point.

To calculate FSF upper capacity violations, the coordinator uses Algorithm 1, which consists of two steps. In the first step, the coordinator selects the foremost time slot and then calculates the number of time instances that will be addressed during the next iteration. The FSF count always rounds off to the nearest integer value.

The selection of the $step_{size}$ affects the solution quality. Here, the value of $step_{size}$ depends on the FSF capacity limits, FSF matte production, required quality of the solution, available computing power, and process personnel experience. A small $step_{size}$ provides a better solution quality at the expense of computational costs, while a big $step_{size}$ means a solution with poor quality but lower computational costs. Finally, the coordinator finds the FSF upper limit violating times and saves them in UT^{h}_{FSF}. At this point, the first step is completed.

The FSF model solves the FSF upper capacity limit violations by adjusting the input feed rate. However, it is possible that despite decreasing the feed rate at conflicting times, the FSF model may not resolve the FSF upper capacity limit violations, and they will appear again in the schedule. It happens especially when the FSF production is high, excessive initial inventory is available in the FSF, or the PSC units are processing the matte slowly due to some technical or other operational reasons. Hence, adjusting the feed rate only at conflicting times will not solve the FSF upper limit violations, and the FSF model will return the same solution that is calculated during a previous iteration where the feed rate is already fixed to the lowest value.

To address this issue, the second step of Algorithm 1 becomes active. In this step, the coordinator checks the feed rate of the conflicting time instances. If the feed rate is not set to its lowest level at the conflicting times, the coordinator sends the UT^{h}_{FSF} directly to the FSF model. However, if the feed rate is already set to the minimum value, the coordinator adds one more time instance to the UT^{h}_{FSF}. This new time instance is always preceding the foremost time instance in the UT^{h}_{FSF}.

The final task of the coordinator is to calculate the prices for the conflicting time instances. The coordinator calculates prices using a proportional–integral (PI) controller that takes into account the difference between the matte demand and availability; therefore, the coordinator calculates the matte imbalance using Eq. (38) and stores it in $UD^{k,h}_{FSF_{past}}$ using Eq. (39). The $UD^{k,h}_{FSF_{past}}$ enables the coordinator to monitor the history of matte surplus; thus, the coordinator will calculate the true prices to maintain a balance between the FSF matte supply and demand. Finally, the coordinator calculates the prices using Eq. (40). Here, $K_{P_{UD}}$ is the proportional gain, and $K_{I_{UD}}$ is the integral

Algorithm 1: FSF upper capacity limit violating time instances

1: Initialization
$slot_{size}$=0
stop=false
count=false
extra=false
First Step
2: **while** *stop* = *false* **do**
  $H \leftarrow h$;
  if $(ULV^{h}_{FSF} + ULV^{h+1}_{FSF}) > 1$ *and* $ULV^{h}_{FSF} \neq 0$ **then**
    $slot_{size} = slot_{size} + 1$
  else
    $stop = true$
  Goto step 2
3: **if** $slot_{size} >> slot_{min}$ **then**
  $FSF_{count} = step_{size} \times slot_{size}$
else
  $FSF_{count} = slot_{min}$
4: **while** *count* = *false* **do**
  $H \leftarrow h$;
  if $(ULV^{h}_{FSF} \neq 0)$ **then**
    $UT^{h}_{FSF} = h$
    $FSF_{count} = FSF_{count} - 1$
  if $FSF_{count} = 0$ **then**
    $count = true$
  Goto step 4

Second Step
5: **while** *extra* = *false* **do**
  $H \leftarrow h$;
  if $UT^{h}_{FSF} \neq 0$ *and* $feed^{h} = feed_{min}$ **then**
    $UT^{h-1}_{FSF} = h - 1$
    $extra = true$
  Goto step 5

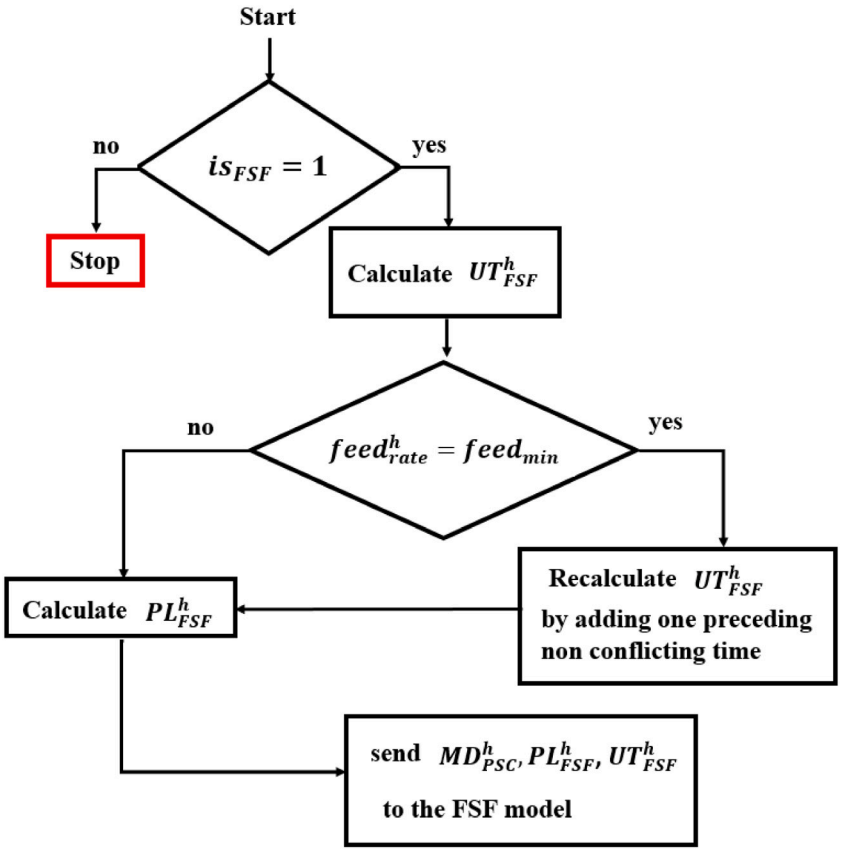


Fig. 7. FSF upper limit violations handling.

gain value.

$$UD^h_{FSF} = \begin{cases} mass^h_{FSF} - MD^h_{PSC} - UL^h & \forall\, UT^h_{FSF} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

$$UD^{k,h}_{FSF_{past}} = \begin{cases} UD^h_{FSF} & UD^h_{FSF} \neq 0,\ k++ \\ UD^{k,h}_{FSF_{past}} & \text{otherwise} \end{cases} \tag{39}$$

$$PU^h_{FSF} = \left(K_{P_{UD}} \times UD^h_{FSF}\right) + \left(K_{I_{UD}} \times \sum_{k=1}^{K} UD^{k,h}_{FSF_{past}}\right) \tag{40}$$

The coordinator sends the UT^h_{FSF}, PU^h_{FSF} and MD^h_{PSC} to the FSF model. The FSF model solves the optimization problem and returns the $mass^h_{FSF}$ and $feed^h_{FSF}$ to the coordinator. If FSF upper limit violations exist in the schedule, it repeats the same actions, and this process continues until all such inter-dependencies are resolved. The manner in which the coordinator addresses the FSF upper limit violations is summarized in Fig. 7.

FSF lower capacity limit violations

The FSF lower limit violations exist in the schedule if the demand for the matte is higher than its supply. It happens when a high matte grade is selected, matte production is low, insufficient initial inventory is available in the FSF at the beginning of the process, or PSC units are operating at a more rapid pace; thus, making the FSF the bottleneck for the entire process.

Like the FSF upper limit violations, multiple FSF lower limit violations can exist in the schedule. Solving all the lower limit violations simultaneously could add unnecessary idle times to the process schedule, which increases the batch time and reduces the solution quality. Furthermore, resolving several lower limit violations means all lower limit violations mean solving multiple batch problems concurrently. As a result, a batch problem may stuck between two local solutions, which could fail the coordinator to provide a feasible solution.

To obtain a schedule with good solution quality, and reasonable computational load demand, this study considers solving only the foremost FSF lower limit violation during a single iteration. As the objective of this framework is to produce schedules with shorter times, solving the foremost FSF lower limit violation implies readjusting only one loading operation. By doing so, the batch time of the conflicting batch problem increases marginally; therefore, the solution's quality is maintained. Moreover, in situations when the matte's demand is not high or fewer batches are produced, solving the foremost lower limit violation can potentially resolve the succeeding FSF lower limit violations in the schedule.

To determine the foremost FSF lower limit violation, the coordinator uses Algorithm 2. Furthermore, it also identifies the conflicting batch problem which generates this FSF lower limit violation. The coordinator sets $batch^{n,b}_{FSF}$ to value 1 if the conflicting time instance belongs to batch problem b on PSC unit n; otherwise, it assigns a value 0. After that, the coordinator calculates the difference between the matte demand and its availability at the time instance LT_{FSF}. Lastly, it determines the total demand for the matte, which was requested by all the batch problems on other PSC units at the conflicting time during the previous iteration. For tracking the matte imbalance, the coordinator saves the LD_{FSF} in $LD^{k,h}_{FSF_{past}}$ using Eq. (41).

$$LD^{k,h}_{FSF_{past}} = \begin{cases} LD_{FSF} & h = LT_{FSF},\ k++ \\ LD^{k,h}_{FSF_{past}} & \text{otherwise} \end{cases} \tag{41}$$

The final activity of the coordinator is to calculate prices to adjust the matte demand at the conflicting time instance. Similar to FSF upper capacity limit violations, the coordinator uses the PI controller to calculate these prices, as given in Eq. (42). Here, $K_{P_{LD}}$ is the proportional gain, and $K_{I_{LD}}$ is the integral gain. If high controller gains values are selected to calculate PL^h_{FSF}, the batch time could increase due to the

Algorithm 2: FSF foremost lower limit violating time instance and matte demand

1: Initialization
$stop$=false
2: **while** $stop = false$ **do**
  $H \leftarrow h$;
  if $LLV^{h}_{FSF} \neq 0$ **then**
    $LT_{FSF} = h$
  else
    $LT_{FSF} = 0$
  $stop = true$
  Goto step 2
3: $N \leftarrow n$;
  $B \leftarrow b$;
if $\left(B^{n,b}_{start} + B^{n,b}_{load_1}\right) \leq LT_{FSF} \leq \left(B^{n,b}_{start} + B^{n,b}_{load_I}\right)$ **then**
  $batch^{n,b}_{FSF} = 1$
else
  $batch^{n,b}_{FSF} = 0$
4: $H \leftarrow h$;
if $h = LT_{FSF}$ **then**
  $LD_{FSF} = MD^{h}_{PSC} - mass^{h}_{FSF} + LL^{h}$
else
  $LD_{FSF} = 0$
5: $H \leftarrow h$;
if $h = LT_{FSF}$ **then**
  $T \leftarrow t$;
  if $t = h - B^{n,b}_{start}$ **then**
    $N \leftarrow n$;
    $B \leftarrow b$;
    if $batch^{n,b}_{FSF} = 0$ **then**
      $MDO^{h}_{PSC} = \sum_{n=1}^{N}\sum_{b=1}^{B} MD^{n,b,t}_{batch}$
    else
      $MDO^{h}_{PSC} = MDO^{h-1}_{PSC}$

presence of unnecessary idle times to the schedule. Therefore, relatively $K_{P_{LD}}$ and $K_{I_{LD}}$ should be selected smaller as compared to the $K_{P_{UD}}$ and $K_{I_{UD}}$.

$$PL^{h}_{FSF} = \left(K_{P_{LD}} \times LD_{FSF}\right) + \left(K_{I_{LD}} \times \sum_{k=1}^{K} LD^{k,h}_{FSF}\right) \tag{42}$$

The coordinator sends MDO^{h}_{PSC}, PL^{h}_{FSF} and LL^{h}_{FSF} to the conflicting batch problem ($batch^{n,b}_{FSF} = 1$). The batch problem solves the scheduling problem and returns its solution to the coordinator. The coordinator inspects the schedule against the FSF lower capacity limit violations. If such violations exist, it repeats the same actions; otherwise, it checks for other inter-dependencies. If the coordinator finds anything in the schedule that contradicts the scheduling policy, it performs a new iteration. However, if the schedule is free of all inter-dependencies, the coordinator returns the final schedule, and the framework terminates.

During the simulation, if one type of inter-dependencies exists in the schedule, the coordinator will solve it without hindrance. However, if FSF and PSC inter-dependencies appear simultaneously, the coordinator must decide the order of inter-dependencies handling. In principle, the coordinator can choose any order. However, the choice of inter-dependencies handling affects the solution quality and computational demands. For example, if logistical inter-dependencies and FSF lower capacity limit violations appear in the schedule simultaneously, solving the logistical inter-dependencies first can potentially solve the FSF lower capacity limit violations. Similarly, if the logistical inter-dependencies and flow inter-dependencies appear simultaneously, solving the logistical inter-dependencies before the flow inter-dependencies can provide a solution with lower computational demands. The order in which the coordinator handles the inter-dependencies is shown in Fig. 8.

Table 1
Framework parameters.

Parameter	Description	Value
mg	Matte grade	65 (percent)
$feed_{min}$	Minimum feed rate	5 (percentage)
$feed_{max}$	Maximum feed rate	100 (percentage)
LL^{h}_{FSF}	Minimum FSF capacity limit	70 (kg)
UL^{h}_{FSF}	Maximum FSF capacity limit	180 (kg)
$step_{size}$	Step size to solve FSF inter-dependencies	10 (percentage)
$slot_{min}$	Minimum time instances in a given slot	1
$matte_{fix}$	Matte transferred from FSF to PSC	20 (kg)
r_{Fe}	Iron oxidation rate	0.240 (kg/min)
r_{S}	Sulphur oxidation rate	0.0330 (kg/min)
$proc_{max_{slag_{blow_1}}}$	Slag blow 1 maximum length	50 (min)
$proc_{min_{slag_{blow_1}}}$	Slag blow 1 minimum length	5 (min)
$proc_{max_{slag_{blow_2}}}$	Slag blow 2 maximum length	60 (min)
$proc_{min_{slag_{blow_2}}}$	Slag blow 2 minimum length	5 (min)
$proc_{fix_{slag_{skimming_i}}}$	Slag removal time	1 (min)
$proc_{fix_{loading_i}}$	Loading time	1 (min)
$Cu_{loss_{slag_{blow_1}}}$	Copper loss rate during slag blow 1	0.103 (kg/min)
$Cu_{loss_{slag_{blow_2}}}$	Copper loss rate during slag blow 2	0.182 (kg/min)
$Cu_{loss_{slag_{blow_3}}}$	Copper loss rate during slag blow 3	0.80 (kg/min)

6. Case studies

In this section, two case studies are presented to demonstrate the effectiveness of the proposed hierarchical framework. In both case studies, the FSF feed rate, matte storage capacity limits, and matte grade value remain the same. Each PSC unit begins with two loading operations, followed by a slag blowing operation and a slag removal operation. It is assumed that the PSC requires four loading operations, three slag blowing operations, three slag-removal operations, and a single long copper blow operation to produce a blister copper batch; therefore, $I = 3$. The blister copper batch is ready when the iron and sulphur content in the matte falls below 2 percent (98 percent copper). Both case studies use a relatively small step size to solve FSF inter-dependencies, and the coordinator considers 10 percent of the time instances from a given slot during a single iteration. Other process parameters are given in Table 1. Both case studies are solved with GAMS CPLEX 12.7, and the optimality gap is kept as the default 10 percent (Bussieck and Meeraus, 2004; IBM, 2017). The computing machine has a 2.60 GHz IntelCoreTM i7 6700HQ processor with 32 GB of RAM, running Windows 10 Enterprise 64-bit.

CPLEX uses the branch-and-cut algorithm to find an optimal solution for discrete-time problems. Generally, a warm start could reduce computational time for the branch-and-cut algorithm. As batch problems contribute much to the overall computational demands, the framework initializes each batch problem with the solution that is calculated by the same batch problem during the previous iteration. Such a warm can potentially reduce the computational demand of the framework.

For PSC inter-dependencies, the coordinator divides all the batch problems into groups. The group number of a batch problem depends on its starting time. Generally, the first batch problem of each PSC unit is assigned to Group 1, the second batch of each PSC unit to Group 2, and the last batch problem of each PSC unit is assigned to Group N. However, if the start time of a batch problem in a group is higher than the start time of the batch problem on the lowest priority PSC unit in

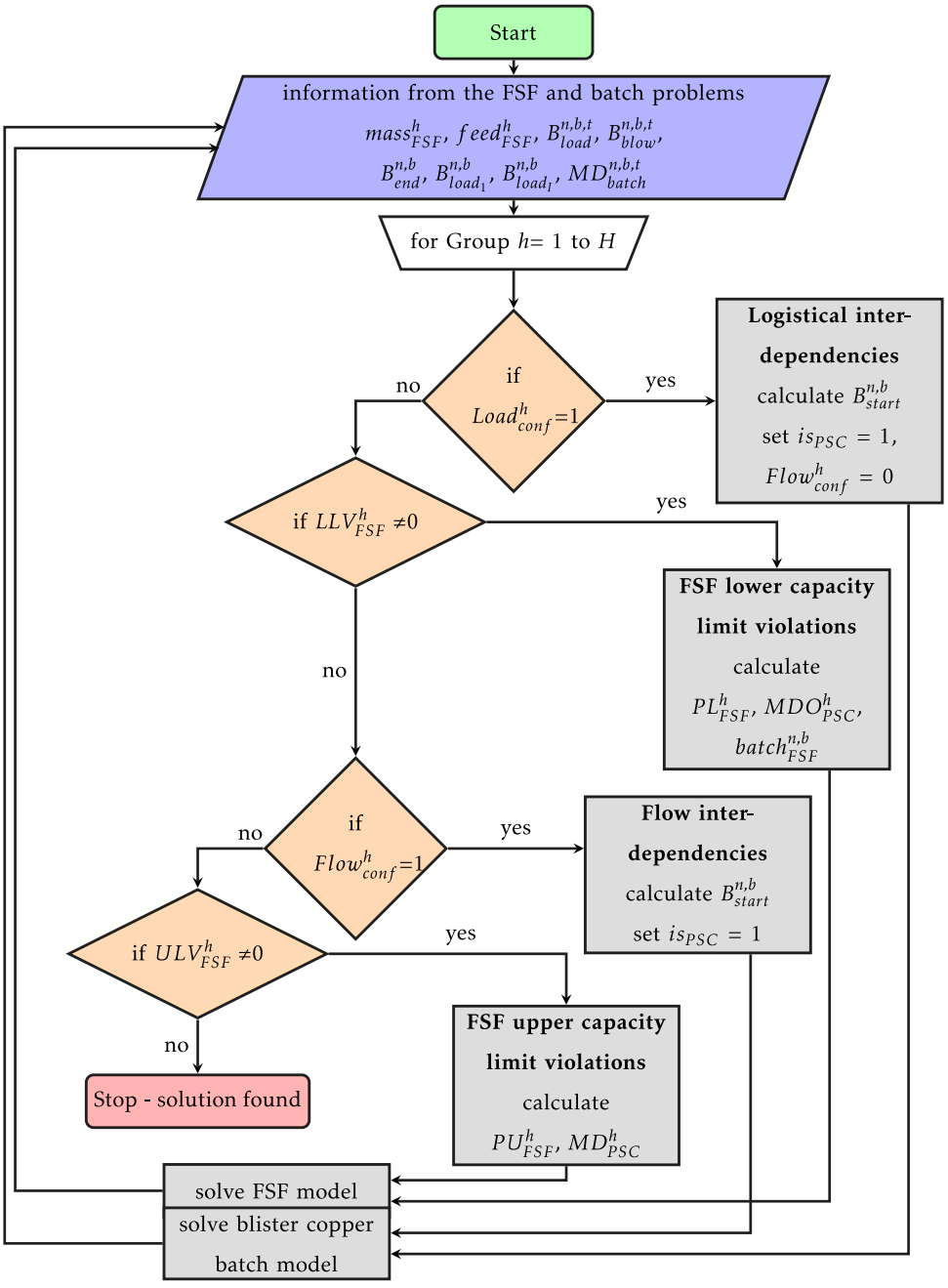


Fig. 8. Order of handling inter-dependencies.

the next group, the batch problem is not added to the current group but to the next group as shown in Fig. 9. Details about it can be found in Ahmed et al. (2022).

In this study, the hierarchical framework can solve one or multiple batch problems simultaneously. By grouping the batch problems, the coordinator can quickly identify the batch problem, which is the source of the foremost FSF or PSC inter-dependencies, and then solve only the conflicting batch problem during the next iteration. By solving only one batch problem in a single iteration, the coordinator will not catch between two local solutions, thus enabling the coordinator to provide a feasible solution in all conditions and to maintain a better solution quality by minimizing idle times in the schedule.

6.1. Case study 1

The first case study considers one FSF and three PSC units, where each PSC unit produces three batches in a synchronized manner. All the PSC units are available at once; therefore, the coordinator assigns priorities to the PSC units randomly. The FSF initial inventory, FSF capacity limits, and the PSC units' priorities are given in Table 2.

During each iteration, the framework solves a maximum number of nine independent batch problems and one FSF problem. After each iteration, the coordinator receives information from the batch problems. Based on the start time, the coordinator divides batch problems into three groups and checks for the FSF and PSC inter-dependencies. Group 1 consists of the first batch problem of each PSC unit, Group 2 has the

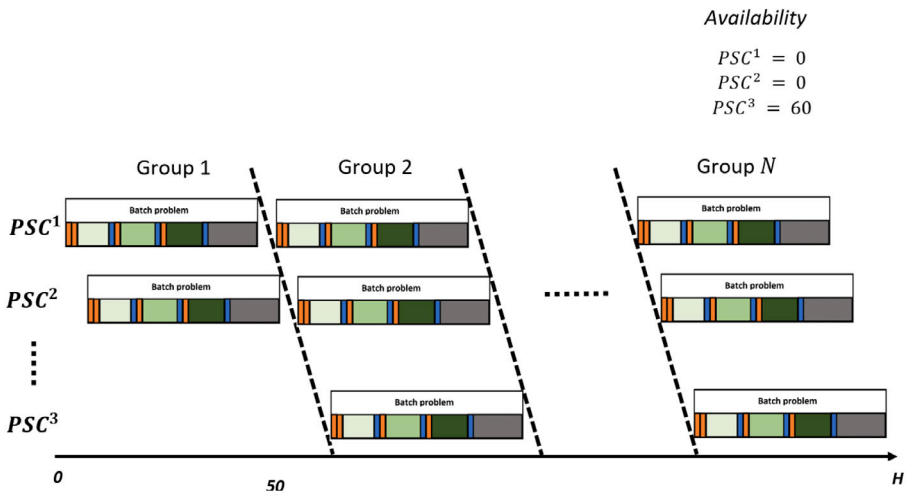


Fig. 9. Batch problems grouping.

Table 2
Framework parameters — Case study 1.

Parameter	Description	Value
$mass_{FSF}^{h=0}$	FSF initial mass	180 (kg)
$start_{PSC}^{c}$	PSC starting time	{0, 0, 0} (min)
H	Complete scheduling horizon	450 (min)
T	Batch problem scheduling horizon	220 (min)
K	Maximum number of iterations	1000

second batch problem of each PSC unit, and Group 3 consists of the third batch problem of each PSC unit. If FSF inter-dependencies exist in the schedule, the coordinator calculates the appropriate prices for the given conflicting time instances using the PI controller and sends those prices to the FSF or conflicting batch problem, depending on the type of FSF violations. On the other hand, if PSC inter-dependencies are present in the schedule, they are addressed by first finding the group of the conflicting batch problems, and the framework simulates the batch problem on the lowest priority PSC unit among the conflicting PSC units. However, if FSF and PSC inter-dependencies exist simultaneously, FSF inter-dependencies are handled before the PSC inter-dependencies.

The FSF feed rate, FSF matte trajectory, and PSC schedule are given in Fig. 10. As enough FSF initial inventory is available and not many blister copper batches are produced, the FSF mass trajectory remains above its lower limit, as shown in Fig. 10(a). Consequently, this case study does not have FSF lower capacity limit violation. However, FSF upper capacity limit violations appear in the schedule at different times. The coordinator divided these times into various slots and addressed only the earliest slot during a single iteration. Fig. 10(b) shows that the coordinator successfully resolved all the FSF upper capacity limit violations by adjusting the FSF feed rate.

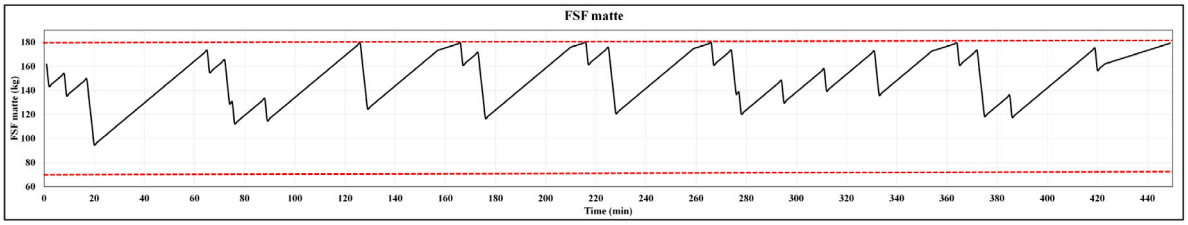
The PSC schedule, as shown in Fig. 10(c), asserts that the PSC units performed loading operations according to their priority numbers, and only one PSC unit is in the slag or blow operation at a single time. Thus, the coordinator efficiently resolved all the logistical and flow inter-dependencies between the PSC units. Furthermore, Fig. 10(c) shows that PSC unit 1 has the shortest batch time, while PSC unit 3 has the highest batch time. Prioritizing the PSC units and grouping the batch problems enable the framework to produce a schedule in which the PSC units produce batches in a synchronized manner and according to the PSC units' priorities. The CPU time required to solve this case study is given in Table 3.

The performance of the PI controller depends on its gain values. Generally, the gain values are calculated from the process step response (Martí et al., 2013). However, such an approach is suitable for simple input–output models. For complex scheduling applications such as the copper smelting process, finding the step response of the process is always challenging. An alternative approach is to use the hit-and-trial method. Here, arbitrary gain values are selected first and adjusted accordingly to achieve the desired performance. In the hit-and-trial, selecting small gain values implies that the coordinator will need more iterations to find the appropriate prices, while high gain values will result in a lower number of iterations, but it can adversely affect the quality of the solution.

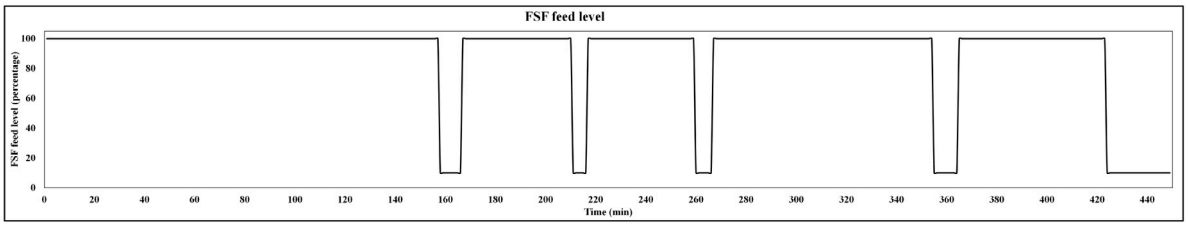
In this study, we used the hit-and-trial approach. The gain values are selected arbitrarily so that the proportional gain value is always lower than the integral gain value. A low proportional gain value means moving steadily towards a feasible solution, while a high integral gain value means moving quickly if the conflicting time instances start repeating during the simulation. Such choices will ensure that the coordinator finds the appropriate prices with moderate computational costs without compromising the solution's quality.

This case study is simulated for three different PI controller gain values as given in Table 3. Generally, PI controller gain values directly affect the computational time of the hierarchical framework, the number of iterations performed by the coordinator to resolve the FSF inter-dependencies, and the number of idle times present in the schedule. Table 3 shows that the framework returned solutions with varying computational times for the given gain values. As FSF inter-dependencies exist in the schedule only due to its upper limit violations, variation in the computational time is due to the simulation time of the FSF model.

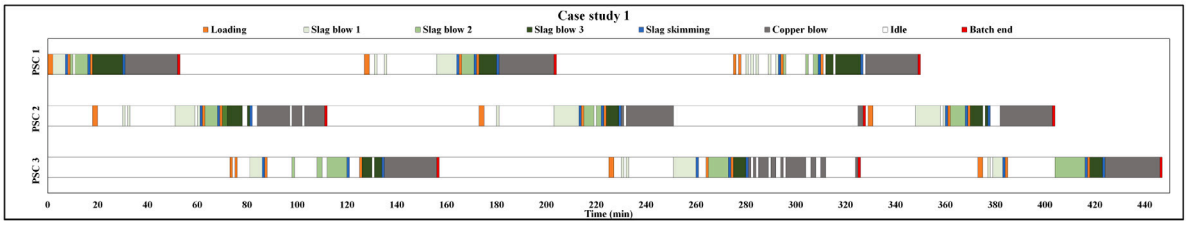
The FSF model treats the prices as weights of the penalty in the objective function; therefore, high gain values increase the FSF model computational time but decrease the number of iterations. For the given gain values, the framework returned the same schedule, as shown in Fig. 10(c). Furthermore, the copper losses depend on the selection of the slag blowing duration during the PSC operation. Since the slag blowing duration of all the batch problems remains the same, the copper losses remain the same. As the objective of this study is not to compare schedules exhaustively based on their computational demands, the process personnel will express interest in using all the given gain values as they all returned feasible schedules with optimal batch time and minimum copper losses.



(a) FSF matte



(b) FSF feed rate



(c) PSC schedule

Fig. 10. Case study 1.

Table 3
Simulation results — Case study 1.

Trial	$K_{P_{UD}}$	$K_{I_{UD}}$	$K_{P_{LD}}$	$K_{I_{LD}}$	Iterations	Computational time (min)	Batch time (min)	Copper loss (kg)
1	20	60.6	2	6.1006	111	462	449	9.0122
2	200	606.6	20	60.6	80	467	449	9.0122
3	20000	60606.6	2000	6060.6	79	629	449	9.0122

6.2. Case study 2

This case study considers one FSF and three PSC units, where each PSC unit produces five batches in a synchronized manner. Therefore, a total number of 15 blister copper batches are produced in this case study. The FSF initial inventory, capacity limits, and PSC units' priorities are given in Table 4. As more blister copper batches are produced here compared to Case study 1, this case study is assumed to have a higher number of FSF and PSC inter-dependencies. Therefore, the coordinator will require more iterations, and the framework will have higher computational demands. Similar to Case study 1, this case study is also simulated for three different PI controller gain values, as given in Table 5; and the FSF feed rate, FSF mass trajectory, and PSC schedule for those gain values are shown in Figs. 11–13.

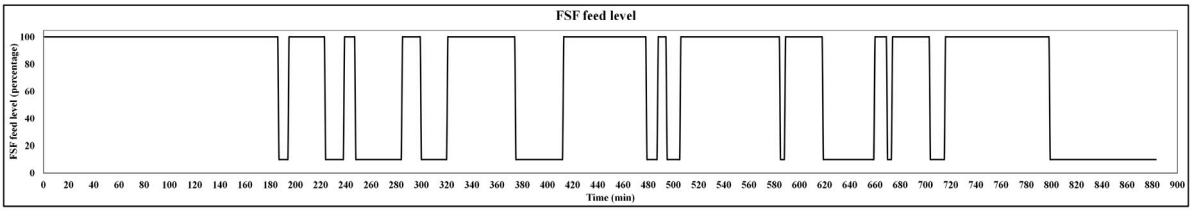
At the end of each iteration, all batch problems share the loading times, blow times, and other related information with the coordinator. The coordinator divides the batch problems into five groups: Group 1 consists of the first batch problem of each PSC unit, Group 2 has the second batch problem of each PSC unit, and so on. During each iteration, the coordinator checks for the FSF and PSC inter-dependencies in ascending order of the scheduling horizon. If inter-dependencies exist between the batch problems of Group 1, the coordinator will address them first. However, if Group 1 is free of all inter-dependencies, it begins to examine Group 2, and this process continues until the coordinator successfully resolves all the inter-dependencies. When this happens, the framework stops and returns the final solution.

Table 4
Framework parameters — Case study 2.

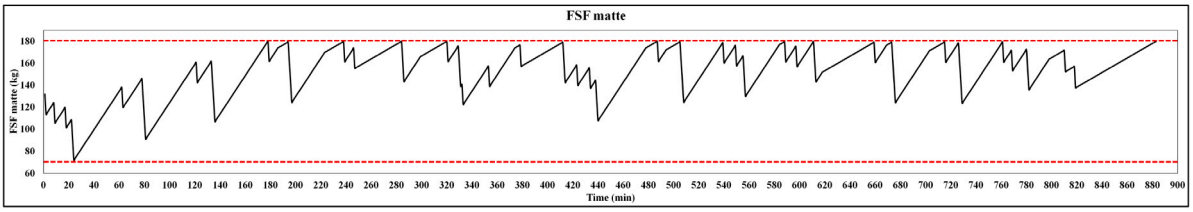
Parameter	Description	Value
$mass_{FSF}^{h=0}$	FSF initial mass	150 (kg)
$start_{PSC}^{c}$	PSC starting time	{0, 0, 0} (min)
H	Scheduling horizon	1000 (min)
T	Batch problem scheduling horizon	220 (min)
K	Maximum number of iterations	2000

Table 5
Simulation results — Case study 2.

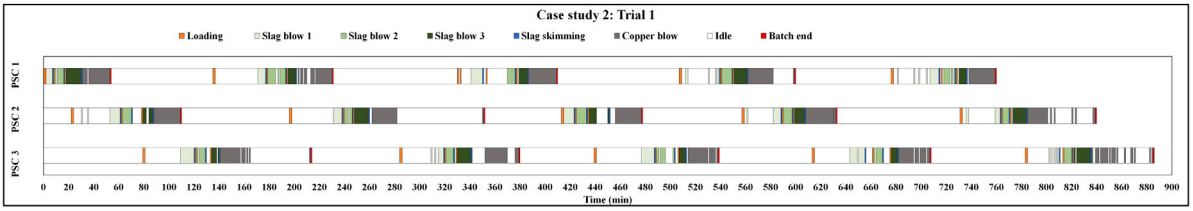
Trial	$K_{P_{UD}}$	$K_{I_{UD}}$	$K_{P_{LD}}$	$K_{I_{LD}}$	Iterations	Computational time (min)	Batch time (min)	Copper loss (kg)
1	20	60.6	2	6.1006	293	1769	883	15.9051
2	200	606.6	20	60.6	176	1484	769	14.6934
3	20000	60606.6	2000	6060.6	211	1549	845	14.8669



(a) FSF feed rate



(b) FSF matte



(c) PSC schedule

Fig. 11. Case study 2.

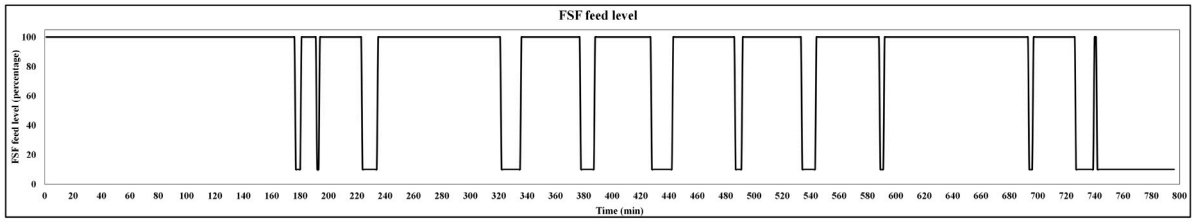
For each group, the coordinator first resolves logistical inter-dependencies, followed by the FSF lower capacity limit violations, and then the flow inter-dependencies as presented in Fig. 8. The coordinator handles the FSF upper capacity limit violations as they appear in the group. If logistical or flow inter-dependencies exist in the schedule, the coordinator determines the conflicting PSC units and the conflicting loading or blows times in the corresponding group. Then, the batch problem on the lowest priority PSC unit from the conflicting units is rescheduled to resolve the logistical or flow inter-dependencies. If FSF inter-dependencies are found in the schedule, the coordinator calculates the prices using the PI controllers and then re-simulates the FSF model or the conflicting batch problem to maintain a balance between the FSF matte supply and demand.

In this case study, FSF lower capacity limit violations exist between time t = 20–25 (min) due to insufficient FSF initial inventory. The coordinator resolved these inter-dependencies by calculating prices using the PI controller so that the first batch problem on PSC unit 2 reduces its demands at the conflicting time instance only, as shown in Figs. 11(b)–11(c), 12(b)–12(c), and 13(b)–13(c). The FSF upper limits violations appear multiple times in the schedule, and the coordinator addresses them as they appear in a group by adjusting the feed rate at the conflicting time instances as shown in Figs. 11(a), 12(a), and 13(a). Figs. 11–13 show that despite the higher number of FSF and PSC inter-dependencies in the schedule, the hierarchical framework provided feasible schedules efficiently with reasonable computational demands.

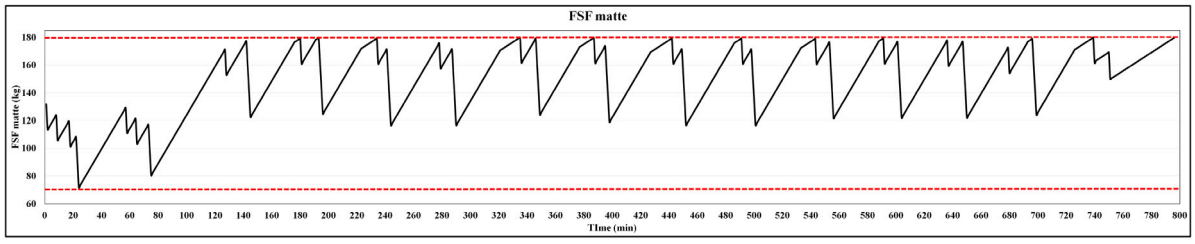
The number of iterations and framework computational demands are presented in Table 5. PSC units are not processing the FSF matte promptly; therefore, they are the bottleneck for the entire process. As a result, a higher number of FSF upper limit violations are present in the schedule. As more batches are produced, this case study has a higher number of logistical and flow inter-dependencies. Thus, the coordinator requires more iterations to solve the FSF and PSC inter-dependencies.

In this study, batch time determines the quality of the solution, and copper losses depend on the solution quality. The blister copper batch model is defined as the minimization problem; therefore, a higher objective function value means that slag blowing durations are not determined optimally, which reduces the solution quality and increases the batch time. Consequently, copper losses during the process operation will be higher. As the Trial 1 batch time and copper losses are higher than Trials 2 and 3, the solution quality of Trial 1 can be termed as the worse in term of copper losses.

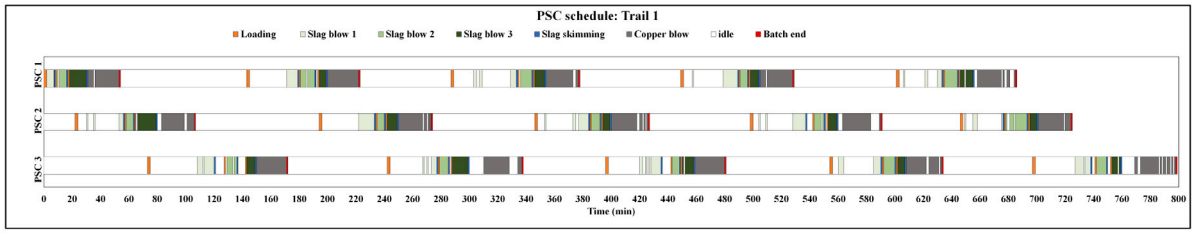
Similar to Case Study 1, the performance of the hierarchical framework depends on the PI controller gain values. For Trial 1 gain values, the number of iterations, computational time, and batch time is higher than in Trials 2 and 3. The grounds for this is that for low gain values, the coordinator performs more iterations to calculate prices using the PI controller; hence, the number of iterations and computational time are



(a) FSF feed rate



(b) FSF matte



(c) PSC schedule

Fig. 12. Case study 2.

higher. Moreover, as FSF inter-dependencies appeared in the schedule repeatedly, the variation in the calculated prices becomes exponential for the Trial 1 gain values. Thus, the FSF or blister copper batch model is penalized with a higher penalty term, which increases the batch time and reduces the solution quality.

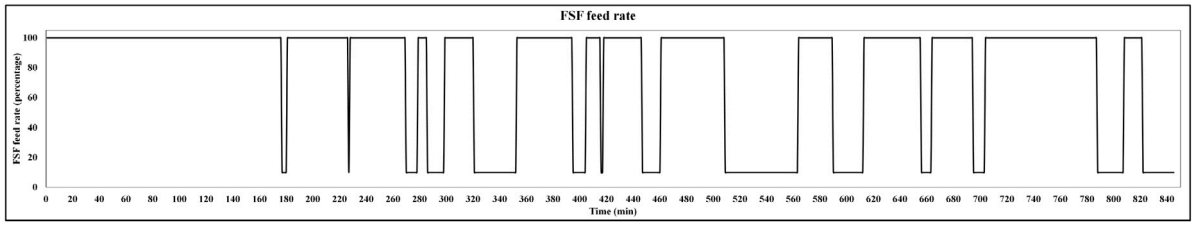
High gain values help the coordinator to determine the appropriate prices quickly to resolve the FSF inter-dependencies; hence, improving the solution quality. Therefore, the number of iterations, computational time, and batch time is lower for Trials 2 and 3 than for Trial 1. For Trial 2, the objective functions of FSF and blister copper batch model are penalized by a small term in contrast to Trial 1, which improves the solution quality. However, choosing too high gain values; for example, in Trial 3, does not improve the solution quality significantly. Choosing too high gain values implies that the PI controller will calculate high prices for the FSF inter-dependencies. As a result, the objective functions are penalized by a higher penalty term that adds extra idle times to the schedule which increased the batch time. Furthermore, an increase in the batch time also increases the number of FSF inter-dependencies, thus increasing the number of iterations and computational time. From a process personnel perspective, as a feasible schedule with shorter batch time and minimum copper losses is always preferred, the solution computed using Trial 2 gain values will be preferred over Trials 1 and 3.

The PI controller gain values in Table 5 are chosen arbitrarily to demonstrate the effects of gain values on the hierarchical framework performance. The hierarchical framework provided feasible schedules for all the given gain values. It implies that process personnel can theoretically use any gain value. The choice of gain values depends on the available computational power, the size of the scheduling problem, the optimality gap, the experience of the process personnel, and the type of scheduling application in hand. Depending on the size and complexity of the scheduling problem, the process personnel can use a step response or hit-and-trial method to determine these optimal controller gain values.

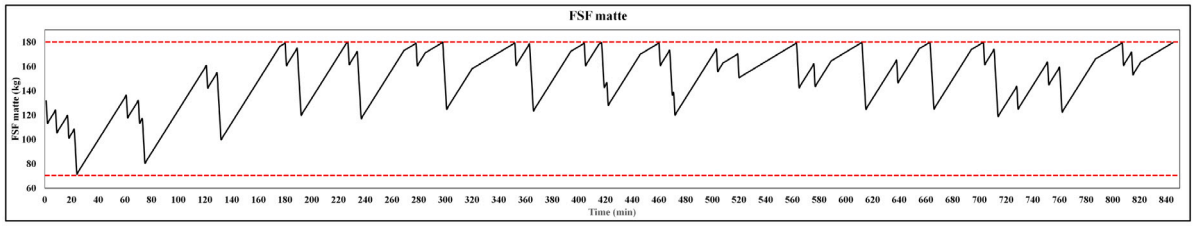
Theoretically, PID controllers perform better than traditional PI controllers. Since the PI controller gain values are chosen arbitrarily in this study, and a wrong gain value may lead to poor solution quality, tuning the PID controller means finding an additional gain value. Furthermore, PID controllers are typically used in processes to overcome process signal overshoot. As the controller goal in this hierarchical framework is to calculate prices considering the present and past conflicting times, the PI controller has the potential to perform this task efficiently. Hence, the PID controller is not used in this study.

An important factor that affects the hierarchical framework performance is the value of the batch problem scheduling horizon T. A higher value of T will increase the batch problem computational time; thus, the computational time of the framework increases. On the contrary, a smaller value may result in an infeasible solution if a batch problem is the source of multiple FSF lower limit violations or PSC inter-dependencies. Therefore, a reasonable T must be selected considering the available computing power, FSF capacity limits, number of PSC units, number of batch problems per unit, and the scheduling recipe.

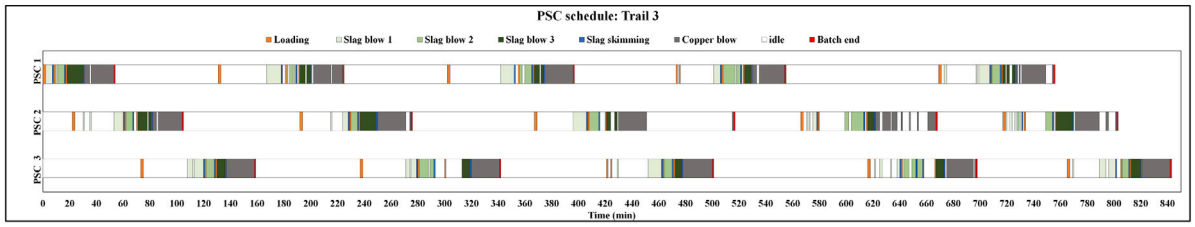
The framework presented in this study allows the FSF to resolve inter-dependencies without using rigorous techniques. In contrast to our previous study [article2] where the FSF unit is intentionally shut-down to resolve the FSF inter-dependencies, this framework provides another way of process operation where the framework allows the units to make their own decisions, resolve their inter-dependencies at the unit level, and provides an optimal process operation. Contrary to a framework based on rigorous techniques, process personnel could use



(a) FSF feed rate



(b) FSF matte



(c) PSC schedule

Fig. 13. Case study 2.

the proposed framework to see the consequences of their operational decisions and improve the process understanding by simulating the framework in varying operating conditions, such as a change in the target matte grade, FSF feed rate or FSF capacity limits.

7. Conclusion

This study presents a novel MILP-based scheduling hierarchical framework for the copper smelting process that consists of one FSF and multiple PSC units. The key feature of this hierarchical framework is to allow continuous operation of the FSF, respect FSF capacity limits, produce PSC batches in a synchronized manner using the predefined scheduling recipe, reduce batch time, and resolve the FSF and PSC inter-dependencies. The proposed scheduling framework can enable the process personnel to foresee the consequences of their operational decisions without a deep understanding of the entire process.

The coordinator has the potential to find a feasible schedule for any PI controller gain values, but the quality of the solution depends on the gain values selection. Selecting too-small gain values increases the computational time of the framework, while too-high gain values do not guarantee improvement in the quality of the solution. Furthermore, only deterministic case studies are presented here to demonstrate that the proposed coordination mechanism can operate the smelting at a near-optimal operational point. Future work will include a thorough sensitivity analysis of the hierarchical framework for the PI controller gain values, extending the price-updating strategy to resolve both the FSF and PSC inter-dependencies and finding other potential price-updating mechanisms for the coordinator that are computationally efficient and provide a better solution quality.

CRediT authorship contribution statement

Hussain Ahmed: Conceptualization, Methodology, Software, Validation, Writing—original draft, Writing – review & editing, Visualization. **Matti Vilkko:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Hussain Ahmed reports was provided by EU Framework Programme for Research and Innovation Research Infrastructures and E-Infrastructures.

Data availability

No data was used for the research described in the article.

References

Ahmed, Hussain, Ricardez-Sandoval, Luis, Vilkko, Matti, 2021. Optimal scheduling of the peirce–smith converter in the copper smelting process. Processes (ISSN: 2227-9717) 9 (11), http://dx.doi.org/10.3390/pr9112004.

Ahmed, Hussain, Ricardez-Sandoval, Luis A., Vilkko, Matti, 2022. Centralized and hierarchical scheduling frameworks for copper smelting process. Comput. Chem. Eng. (ISSN: 0098-1354) 164, 107864. http://dx.doi.org/10.1016/j.compchemeng.2022.107864.

Bussieck, Michael R., Meeraus, Alex, 2004. General algebraic modeling system (GAMS). In: Modeling Languages in Mathematical Optimization. Springer US, Boston, MA, ISBN: 978-1-4613-0215-5, pp. 137–157. http://dx.doi.org/10.1007/978-1-4613-0215-5_8.

Cheng, Ruoyu, Forbes, J. Fraser, Yip, W. San, 2006. Coordinated decentralized MPC for plant-wide control of a pulp mill benchmark problem. IFAC Proc. Vol. (ISSN: 1474-6670) 39 (2), 971–976. http://dx.doi.org/10.3182/20060402-4-BR-2902.00971, 6th IFAC Symposium on Advanced Control of Chemical Processes.

Cheng, R., Forbes, J.F., Yip, W.S., 2007. Price-driven coordination method for solving plant-wide MPC problems. J. Process Control (ISSN: 0959-1524) 17 (5), 429–438. http://dx.doi.org/10.1016/j.jprocont.2006.04.003, Selected papers presented at the 2005 IFAC World Congress.

Cheng, Ruoyu, Fraser Forbes, J., San Yip, W., 2004a. Dantzig-wolfe decomposition and large-scale constrained MPC problems. IFAC Proc. Vol. (ISSN: 1474-6670) 37 (9), 953–958. http://dx.doi.org/10.1016/S1474-6670(17)31931-6, 7th IFAC Symposium on Dynamics and Control of Process Systems 2004 (DYCOPS -7), Cambridge, USA, 5-7 July, 2004.

Cheng, Ruoyu, Fraser Forbes, J., San Yip, W., 2004b. Dantzig-wolfe decomposition and large-scale constrained MPC problems. IFAC Proc. Vol. (ISSN: 1474-6670) 37 (9), 953–958. http://dx.doi.org/10.1016/S1474-6670(17)31931-6, 7th IFAC Symposium on Dynamics and Control of Process Systems 2004 (DYCOPS -7), Cambridge, USA, 5-7 July, 2004.

Christodoulopoulos, K., Sourlas, V., Mpakolas, I., Varvarigos, E., 2009a. A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in grid networks. Comput. Commun. (ISSN: 0140-3664) 32 (7), 1172–1184. http://dx.doi.org/10.1016/j.comcom.2009.03.004.

Christodoulopoulos, K., Sourlas, V., Mpakolas, I., Varvarigos, E., 2009b. A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in grid networks. Comput. Commun. (ISSN: 0140-3664) 32 (7), 1172–1184. http://dx.doi.org/10.1016/j.comcom.2009.03.004.

Davenport, W.G.I., King, M., Schlesinger, M.E., Biswas, A.K., 2002. Extractive metallurgy of copper. Elsevier Science, ISBN: 9780080531526, https://books.google.fi/books?id=VdILLpCu9ooC.

Davenport, W.G., Partelpoeg, E.H., 2015. Flash Smelting: Analysis, Control and Optimization, second ed. Elsevier Science, ISBN: 9781483150901, pp. 1–77, 187–189, 2206–2213, https://books.google.fi/books?id=cMcgBQAAQBAJ.

Eastin, Richard V., Arbogast, Gary L., 2011. Demand and supply analysis: Introduction. https://www.cfainstitute.org/-/media/documents/support/programs/cfa/prerequisite-economics-material-demand-and-supply-analysis-intro.pdf. (Accessed 19 July 2022).

Gao, Hongjun, Liu, Junyong, Wang, Lingfeng, Wei, Zhenbo, 2018. Decentralized energy management for networked microgrids in future distribution systems. IEEE Trans. Power Syst. 33 (4), 3599–3610. http://dx.doi.org/10.1109/TPWRS.2017.2773070.

Harjunkoski, Iiro, Borchers, Hans Werner, Fahl, Marco, 2006. Simultaneous scheduling and optimization of a copper plant. In: Marquardt, W., Pantelides, C. (Eds.), 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering. In: Computer Aided Chemical Engineering, vol. 21, Elsevier, (ISSN: 1570-7946) pp. 1197–1202. http://dx.doi.org/10.1016/S1570-7946(06)80209-9.

Harjunkoski, Iiro, Grossmann, Ignacio E., 2001. A decomposition approach for the scheduling of a steel plant production. Comput. Chem. Eng. (ISSN: 0098-1354) 25 (11), 1647–1660. http://dx.doi.org/10.1016/S0098-1354(01)00729-3.

IBM, 2017. V12.7 IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. International Business Machines Corporation, https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf.

IEA, 2022. Total copper demand by sector and scenario, 2020–2040. https://www.iea.org/data-and-statistics/charts/total-copper-demand-by-sector-and-scenario-2020-2040. (Accessed 19 July 2022).

International Copper Association, 2017. Global mega trends and key key markets for copper. https://issuu.com/copperalliance/docs/ica-annual-report-2017-final-lowres?e=11116907/59702325. (Accessed 19 July 2022).

Jose, Rinaldo A., Ungar, Lyle H., 2000. Pricing interprocess streams using slack auctions. AIChE J. 46 (3), 575–587. http://dx.doi.org/10.1002/aic.690460316.

Kelly, Jeffrey D., Zyngier, Danielle, 2008. Hierarchical decomposition heuristic for scheduling: Coordinated reasoning for decentralized and distributed decision-making problems. Comput. Chem. Eng. (ISSN: 0098-1354) 32 (11), 2684–2705. http://dx.doi.org/10.1016/j.compchemeng.2007.08.007, Enterprise-Wide Optimization.

Lavios Villahoz, Juan José, del Olmo Martínez, Ricardo, Arauzo, Alberto Arauzo, 2010. Price-setting combinatorial auctions for coordination and control of manufacturing multiagent systems: Updating prices methods. In: Ortiz, Ángel, Franco, Rubén Darío, Gasquet, Pedro Gómez (Eds.), Balanced Automation Systems for Future Manufacturing Networks. Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-14341-0, pp. 293–300.

Liu, Jian-Hua, hua Gui, Wei, Xie, Yong-Fang, Yang, Chun-Hua, 2014. Dynamic modeling of copper flash smelting process at a smelter in China. Appl. Math. Model. (ISSN: 0307-904X) 38 (7), 2206–2213. http://dx.doi.org/10.1016/j.apm.2013.10.035.

Martí, Rubén, Sarabia, Daniel, Navia, Daniel, de Prada, César, 2013. A method to coordinate decentralized NMPC controllers in oxygen distribution networks. Comput. Chem. Eng. (ISSN: 0098-1354) 59, 122–137. http://dx.doi.org/10.1016/j.compchemeng.2013.05.023, Selected papers from ESCAPE-22 (European Symposium on Computer Aided Process Engineering - 22), 17-20 June 2012, London, UK.

Moroşan, P., Bourdais, R., Dumur, D., Buisson, J., 2010. Distributed model predictive control based on benders' decomposition applied to multisource multizone building temperature regulation. In: 49th IEEE Conference on Decision and Control. CDC, pp. 3914–3919. http://dx.doi.org/10.1109/CDC.2010.5717092.

Popa, Cristina, 2014. Application of plantwide control strategy to the catalytic cracking process. Procedia Eng. (ISSN: 1877-7058) 69, 1469–1474. http://dx.doi.org/10.1016/j.proeng.2014.03.143, 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

Pradenas, L., Fernandez, R., Parada, V., Caballero, C., Zuniga, J., 2003. A solution to the copper smelter scheduling problem. Pyrom. Copp. 4 (11), 351–357. http://dx.doi.org/10.1016/j.compchemeng.2004.05.002, The Hermann Schwarze Symposium on Copper Pyrometallurgy.

Rokhforoz, Pegah, Fink, Olga, 2021. Hierarchical multi-agent predictive maintenance scheduling for trains using price-based approach. Comput. Ind. Eng. (ISSN: 0360-8352) 159, 107475. http://dx.doi.org/10.1016/j.cie.2021.107475.

Ruben, Marti, Daniel, Sarabia, Daniel, Navia, Cesar, De Prada, 2013. Coordination of distributed model predictive controllers using price-driven coordination and sensitivity analysis. IFAC Proc. Vol. (ISSN: 1474-6670) 46 (32), 215–220. http://dx.doi.org/10.3182/20131218-3-IN-2045.00035, 10th IFAC International Symposium on Dynamics and Control of Process Systems.

Schipper, Branco W., Lin, Hsiu-Chuan, Meloni, Marco A., Wansleeben, Kjell, Heijungs, Reinout, van der Voet, Ester, 2018. Estimating global copper demand until 2100 with regression and stock dynamics. Resour. Conserv. Recy. (ISSN: 0921-3449) 132, 28–36. http://dx.doi.org/10.1016/j.resconrec.2018.01.004.

Suominen, Olli, Mörsky, Ville, 2016. Framework for optimization and scheduling of a copper production plant. In: Kravanja, Zdravko, Bogataj, Miloš (Eds.), 26th European Symposium on Computer Aided Process Engineering. In: Computer Aided Chemical Engineering, vol. 38, Elsevier, (ISSN: 1570-7946) pp. 1243–1248. http://dx.doi.org/10.1016/B978-0-444-63428-3.50212-5.

Tan, Pengfu, 2007. Applications of thermodynamic modeling in copper converting operations. Int. J. Mater. Res. 98 (10), 995–1003. http://dx.doi.org/10.3139/146.101548.

Voos, Holger, 2007. Resource allocation in continuous production using market-based multi-agent systems. In: 2007 5th IEEE International Conference on Industrial Informatics, vol. 2. (ISSN: 2378-363X) pp. 1085–1090. http://dx.doi.org/10.1109/INDIN.2007.4384926.