



Price-based coordination strategy for copper smelting process

Hussain Ahmed*, Matti Vilkkö

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

ARTICLE INFO

Keywords:

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

ABSTRACT

Rigorous scheduling that uses hard heuristics and allows forceful shutdown of process units, whether intentionally or unintentionally, in the copper smelting process always harms the process operation. This way of process scheduling often results in sub-optimal process operation. Although a flexible scheduling framework that empowers process units to make independent operational decisions is desirable, the inter-dependencies among these units continue to pose a challenge in its design. This study presents a hierarchical scheduling framework for the copper smelting process, which uses price-based heuristics to resolve process inter-dependencies among the process units. These inter-dependencies arise due to the bottlenecks present in the process. Taking advantage of market theory, the coordinator uses PI controllers to find the appropriate prices to address the inter-dependencies. Two case studies are presented to demonstrate that the proposed framework effectively resolves all the process inter-dependencies, indicating its potential for application in other large-scale scheduling applications.

Nomenclature

FSF:

Set:

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

Variables:

FSF_{FR}^h (ton/min) = concentrate feed rate at time h

FSF_p^h (ton/min) = FSF matte production at time h

FSF_M^h (ton) = mass of FSF at time h

Parameters:

mg (percentage) = matte grade

$feed_{max}$ (percentage) = maximum input concentrates feed rate

$feed_{min}$ (percentage) = minimum input concentrates feed rate

FSF_{UL}^h (ton) = FSF upper capacity limit at time h

FSF_{LL}^h (ton) = FSF lower capacity limit at time h

$FSF_M^{h=0}$ (ton) = FSF initial inventory level

PSC:

Sets:

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

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

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

Variables:

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

$$s_{z_i}^t = \begin{cases} 1 & \text{operation } b_{z_i}^{t'} \text{ is completed at time } t' \quad \forall t \geq t' \\ 0 & \forall t < t' \end{cases}$$

$$BP_{idle}^t = \begin{cases} 1 & b_{z_i}^t = 1 \\ 0 & \text{otherwise} \end{cases}$$

BP_{mass}^t (ton) = matte in PSC unit at time t

BP_{ele}^t (ton) = content of element ele in the PSC matte at time t . This ele can be iron (Fe) or sulphur (S).

BP_{MD}^t (ton) = matte required by batch problem at time t

BP_{Cu}^t (ton) = minimum copper loss point at time t . We calculated these points using the methodology outlined in our previous work (Ahmed et al., 2021).

BP_{blow}^t (ton) = slag or copper-blowing happens at time t

Parameters:

$Cu_{loss_{z_i}}$ (ton/min) = The rate of copper loss during operation z_i

* Corresponding author.

E-mail addresses: hussain.ahmed@tuni.fi (H. Ahmed), matti.vilkkö@tuni.fi (M. Vilkkö).

r_{ele} (ton/min) = oxidation rate of element ele
 n_{z_i} = the position of operation z_i within the set Z
 max_{z_i} (min) = maximum processing duration of operation z_i
 min_{z_i} (min) = minimum processing duration of operation z_i
 fix_{z_i} (min) = fixed processing duration of operation z_i
 MT_{fix} (ton) = amount of matte transferred from FSF to a PSC unit
 PLB_{PSC}^t = price value to resolve logistical inter-dependencies at time t
 PFB_{PSC}^t = price value to resolve flow inter-dependencies at time t
 PLB_{FSF}^t = price value to resolve FSF lower-level inter-dependencies at time t
 LT_{BP} (min) = earliest time for a batch problem to begin loading operation without generating logistical inter-dependencies
 FT_{BP} (min) = earliest time for a batch problem to begin blowing operation without generating flow inter-dependencies
 FT_{FSF} (min) = earliest time for a PSC batch to perform loading operation without generating FSF lower-level inter-dependencies
 $MDOB_{FSF}^t$ (ton) = matte demand by batch problems on other PSC units at FSF lower limit violating time t
 LLB_{FSF}^t (ton) = FSF lower capacity limit at time t

$BP_{load}^{n,b,j}$ (min) = $\begin{cases} t & \text{loading time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$
 $BP_{blow}^{n,b,j}$ (min) = $\begin{cases} t & \text{slag or copper blowing time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$
 $BP_{start}^{n,b}$ (min) = $\begin{cases} t & \text{start time of batch problem on PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$
 $BP_{end}^{n,b}$ (min) = $\begin{cases} t & \text{end time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$
 $BP_{MD}^{n,b,j}$ (ton) = $\begin{cases} \mathbb{Z}^+ & \text{matte demand of PSC unit } n \text{ during batch } b \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$
 $BP_{load_1}^{n,b}$ (min) = $\begin{cases} t & \text{first loading time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$
 $BP_{load_l}^{n,b}$ (min) = $\begin{cases} t & \text{last loading time of PSC unit } n \text{ during batch } b \\ 0 & \text{otherwise} \end{cases}$

Coordinator:

Set:

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

Parameters:

$start_{PSC}^n$ (min) = start time of PSC unit n
 $prior_{PSC}^n$ = priority of PSC unit n
 LG = foremost group that has logistical inter-dependencies
 FG = foremost group that has flow inter-dependencies
 PSC_{conf}^n = PSC unit n is the source of inter-dependencies
 PSC_{LL} = lowest priority PSC unit having logistical inter-dependencies
 PSC_{LU} = highest priority PSC unit having logistical inter-dependencies
 PSC_{FL} = lowest priority PSC unit having flow inter-dependencies
 PSC_{FU} = highest priority PSC unit having flow inter-dependencies
 $time_{LA}$ (min) = earliest available time for a batch problem to begin its loading operation without generating logistical inter-dependencies
 $time_{FA}$ (min) = earliest available time for a batch problem to begin its blowing operation without generating flow inter-dependencies
 $time_{LDF}^{n,b,h}$ (min) = difference between the available and demanded loading times on PSC unit n in the group b having logistical inter-dependencies at time h
 $time_{EDF}^{n,b,h}$ (min) = difference between the available and demanded blowing times on PSC unit n in group b having flow inter-dependencies at time h
 FSF_{ULLV}^h (min) = FSF upper limit violation exists at time h
 FSF_{LLV}^h (min) = FSF lower limit violation exists at time h
 FSF_{TS} = total number of time instances in a time slot

FSF_{TSS} = portion of FSF_{TS}
 min_{TS} = minimum number of time instances in a time slot
 PSC_{MD}^h (ton) = matte demanded by PSC units at time h
 FSF_{MDO}^h (ton) = matte demanded by all other PSC units except the unit that is the source of FSF lower-level inter-dependencies
 FSF_{ULD}^h (ton) = difference between the available and demanded matte at time h having FSF upper-level inter-dependencies
 $FSF_{ULD}^{h,k}$ (ton) = difference between the available and demanded matte at time h during iteration k having FSF upper-level inter-dependencies
 FSF_{ULT}^h = FSF upper-level inter-dependencies exist at time h ($\forall h \leq FSF_{TSS}$)
 FSF_{LT} (min) = foremost time at which FSF lower-level inter-dependencies exist
 FSF_{LLD}^h (ton) = difference between the available and demanded matte at time h having FSF lower-level inter-dependencies
 $FSF_{LLD}^{h,k}$ (ton) = difference between the available and demanded matte at time h during iteration k having FSF lower-level inter-dependencies
 $load_{price}^{n,b,h}$ = price value for the logistical inter-dependencies on PSC unit n in group b at time h
 $flow_{price}^{n,b,h}$ = price value for the flow inter-dependencies on PSC unit n in group b at time h
 UL_{price}^h = price value for FSF upper-level inter-dependencies at time h
 LL_{price}^h = price value for FSF lower-level inter-dependencies at time h
 P_{load} = PI controller proportional gain value to calculate prices for the logistical inter-dependencies
 I_{load} = PI controller integral gain value to calculate prices for the logistical inter-dependencies
 P_{flow} = PI controller proportional gain value to calculate prices for the flow inter-dependencies
 I_{flow} = PI controller integral gain value to calculate prices for the flow inter-dependencies
 P_{UL} = PI controller proportional gain value to calculate prices for the FSF upper-level inter-dependencies
 I_{UL} = PI controller integral gain value to calculate prices for the FSF upper-level inter-dependencies
 P_{LL} = PI controller proportional gain value to calculate prices for the FSF lower-level inter-dependencies
 I_{LL} = PI controller integral gain value to calculate prices for the FSF lower-level inter-dependencies

$PSC_{LID}^{n,b}$ = $\begin{cases} 1 & \text{logistical inter-dependencies exist on PSC unit } n \text{ in group } b \\ 0 & \text{otherwise} \end{cases}$
 $PSC_{LBG}^{n,b}$ = $\begin{cases} 1 & \text{logistical inter-dependencies exist on PSC unit } n \text{ between group } b \\ & \text{and preceding group} \\ 0 & \text{otherwise} \end{cases}$
 $PSC_{FA}^{n,b,h}$ = $\begin{cases} 1 & \text{slag or copper blowing is made to PSC unit } n \text{ of group } b \text{ at time } h \\ 0 & \text{otherwise} \end{cases}$
 $PSC_{FID}^{n,b}$ = $\begin{cases} 1 & \text{flow inter-dependencies exist on PSC unit } n \text{ in group } b \\ 0 & \text{otherwise} \end{cases}$
 $PSC_{FBG}^{n,b}$ = $\begin{cases} 1 & \text{flow inter-dependencies exist on PSC unit } n \text{ between group } b \\ & \text{and preceding group} \\ 0 & \text{otherwise} \end{cases}$
 $PSC_{MD}^{n,b}$ = $\begin{cases} 1 & \text{logistical inter-dependency exists at time } h \\ 0 & \text{otherwise} \end{cases}$
 GP_{load}^b = $\begin{cases} 1 & \text{logistical inter-dependencies exist in group } b \\ 0 & \text{otherwise} \end{cases}$
 GP_{flow}^b = $\begin{cases} 1 & \text{flow inter-dependencies exist in group } b \\ 0 & \text{otherwise} \end{cases}$
 is_{PSC} = $\begin{cases} 1 & \text{PSC inter-dependencies exist in the schedule} \\ 0 & \text{otherwise} \end{cases}$
 $time_{LD}^{n,b,h}$ (min) = $\begin{cases} h & \text{time difference between the start time and last loading operation} \\ & \text{on PSC unit } n \text{ in group } b \text{ at time } h \\ 0 & \text{otherwise} \end{cases}$

$$\begin{aligned}
 \text{time}_{LD}^{n,b,h,k}(\text{min}) &= \begin{cases} h & \text{difference between the available and demanded} \\ & \text{loading times by PSC unit } n \text{ in group } b \text{ at time } h \\ & \text{during iteration } k \\ 0 & \text{otherwise} \end{cases} \\
 \text{time}_{FD}^{n,b,h}(\text{min}) &= \begin{cases} h & \text{flow inter-dependencies exist in PSC unit } n \text{ in} \\ & \text{group } b \text{ at time } h \\ 0 & \text{otherwise} \end{cases} \\
 \text{time}_{FD}^{n,b,h,k}(\text{min}) &= \begin{cases} h & \text{difference between the available and demanded} \\ & \text{blowing times by PSC unit } n \text{ in group } b \text{ at time } h \\ & \text{during iteration } k \\ 0 & \text{otherwise} \end{cases} \\
 \text{FSF}_{BP LL}^{n,b} &= \begin{cases} 1 & \text{PSC unit } n \text{ in group } b \text{ is the source of FSF lower-level} \\ & \text{inter-dependencies} \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

1. Introduction

The current copper industry faces a strong need to increase its production, reduce operating costs, and its environmental footprint (Li et al., 2017). In addition, this industry is struggling to neutralize the growing market demand considering depletion in the high-quality copper concentrates and increasing utilization of low-quality concentrates (Li et al., 2017; Schipper et al., 2018). The low-quality concentrate utilization leads to higher operational costs in terms of copper losses that adversely affect the process throughput. Given the rising demand for copper, the stakeholders are interested in finding new scheduling techniques that improve the process throughput, reduce copper losses and improve the overall process performance.

In the copper industry, copper smelters are used for copper production. The copper smelting process involves various units, and its performance depends heavily on the scheduling of these units, which have strong inter-dependencies (Ahmed et al., 2021, 2022). If these units are scheduled without considering the inter-dependencies among them, the process's throughput will be negatively impacted, potentially resulting in severe consequences for the smelting process.

The copper smelting process is considered a large-scale process. Traditionally, experienced process personnel has been responsible for scheduling a large-scale process. Although many scheduling solutions for the copper smelting process exist in the market, the process personnel is reluctant to use those available solutions as they neglect the real industrial practices, ignore inter-dependencies between the process units, and the complexity associated with those solutions (Björklund et al., 2021; Harjunkoski and Grossmann, 2001; Ahmed et al., 2022).

In the copper smelting process, Flash smelting furnace (FSF) and Peirce-Smith converter (PSC) contribute much to the removal of unwanted elements from the input concentrates and to the copper losses during the process operation (Davenport et al., 2002). Both units are subject to a variety of storage, logistical, and operational constraints; hence, strong inter-dependencies between the FSF and PSC units. Ahmed et al. (2022). If these units are operated independently, it will present a significant challenge for process personnel to operate a unit at the optimal operating point and actively respond to any emerging situation (Ahmed et al., 2022). Furthermore, this way of process operation will require a strong collaboration between process personnel to maintain a common understanding of the entire process. As copper smelting processes are constantly upgraded to meet the growing demand, achieving common ground and expertise of all process personnel is sometimes impossible. Hence, modern scheduling solutions are required that have the potential to resolve process inter-dependencies and enable process personnel to operate a processing unit independently without a thorough understanding of the process and foresee the consequences of the operational decisions on the entire smelting process.

For designing a scheduling framework for the copper smelting process, one approach is to select a coordination variable and then use that variable to address the inter-dependencies of the process. Choosing a coordination variable requires a thorough understanding of the process, and it may require continuous redefinition if the process dynamics or its objective changes continuously. Although this approach may be suitable for simple scheduling tasks, it becomes time-consuming and challenging to implement when dealing with complex scheduling applications such as the copper smelting process.

An alternative approach is to use heuristics (Sobeyko and Mönch, 2015; Adams and Seider, 2009; Zhang et al., 2011; Abreu et al., 2022; Panek et al., 2005). Heuristics describe the procedure of unit interaction, information sharing, and methods to resolve conflicts among the process units. Generally, they are application-dependent, easy to troubled shoot, and open to adjustment, and they can be designed based on industrial practices, derived from a well-known technique, or based on rigorous decisions to resolve inter-dependencies among the process units (Ahmed et al., 2022; Ahmed and Vilkkö, 2023). Considering that a forceful interruption of a unit operation is not fostered in the copper smelting process, scheduling approaches based on rigorous heuristics are unsuitable for the copper smelting process. Thus, innovative heuristics-based scheduling solutions are required for the copper smelting process that allows the process units to make their own choices to resolve process inter-dependencies.

In previous work, we presented hierarchical scheduling for a copper smelting process that resolved process inter-dependencies using hard heuristics (Ahmed et al., 2022). Alternative to the previous approach, this study focuses on a scheduling framework that first determines the bottlenecks that are the source of inter-dependencies generation and then resolves these inter-dependencies by finding suitable prices using proportional-integral (PI) controllers. The proposed scheduling framework considers a copper smelting process that consists of one FSF and multiple PSC units, and it is designed using discrete-time mixed integer linear programming (MILP) techniques. The objective is to formulate a scheduling framework that maximizes the FSF throughput while respecting the FSF storage capacity limits, minimizes the copper losses, and allows the process units to resolve inter-dependencies without being dictated by any external entity.

2. Theoretical background

For formulating a scheduling framework for a large-scale industrial process, two well-known approaches are centralized and hierarchical, which are used in numerous industrial applications (Ahmed and Vilkkö, 2023; Harjunkoski and Grossmann, 2001; Cheng et al., 2006, 2007, 2004; Popa, 2014). In the centralized approach, all the information is collected in one place, and then a single large scheduling problem is designed. Generally, this approach requires high computational resources, challenges in maintaining, and has a single-point failure and data-sharing issues if units are manufactured and assembled by different vendors; therefore, it is suitable for small-scale scheduling applications (Cheng et al., 2004; Rokhforoz and Fink, 2021; Gao et al., 2018). In our previous work, we developed a centralized scheduling framework for the copper smelting process, and we have demonstrated that the centralized approach is not the best option for the copper smelting process (Ahmed et al., 2022).

An alternative to the centralized approach is to use the hierarchical approach. Here, a large-scale optimization problem is decomposed into several unit-level scheduling problems, which are solved independently. To enable unit-level communication and provide an optimal operation of the process, a top-level entity is used, which is typically referred to as the coordinator. The coordinator is responsible for all data exchange between the unit-level scheduling problems and resolving process inter-dependencies if they arise during the process operation. Furthermore, it can employ heuristics or other coordination mechanisms to achieve a plant-wide solution.

Heuristics based on the price-updating scheme are derived from market theory. Typically, various scarce resources can exist in a market, and market forces determine prices for these resources. In an open market, the price rises when the demand for a scarce resource exceeds its supply and vice versa. Price-based coordination relies first on identifying the scarce resources in the scheduling application and then utilizing their prices to coordinate the allocation of these resources to provide a plant-wide optimal solution.

For designing a scheduling framework for the copper smelting process, multiple solutions are presented. Pradenas et al. (2003) presented a heuristic-based scheduling framework for the copper smelting process that maximizes the process throughput. This framework produces optimal schedules for the smelting process, which has numerous operational constraints. Rojas et al. (2006) developed a mathematical framework for the copper smelting process that maximizes the production considering metallurgical, operational, environmental, and other constraints. An industrial case study is presented to demonstrate the effectiveness of the proposed framework. Harjunkoski et al. (2006) developed an interesting scheduling framework for the copper smelting process using continuous-time MILP techniques where the objective is to maximize the process throughput. Suominen and M (2016) developed an innovative continuous-time scheduling framework for a copper smelter. This framework maximizes smelting production and provides an optimal schedule. Björklund et al. (2021) proposed an automatic control solution for the FSF. This solution has online feed rate and matte level estimators to continuously estimate and monitor the feed rate and matte level in the FSF. In our previous work (Ahmed et al., 2022), we presented a simple and effective scheduling framework using discrete-time MILP techniques where the coordinator uses rigorous heuristics to resolve process inter-dependencies and provide a feasible schedule.

Price-based coordination is also used in multiple industrial applications. Ruben et al. (2013) used price-based heuristics to coordinate four unique control problems to provide a plant-wide solution. For price-updating, this framework used Newton's method. Lavios Villahoz et al. (2010) used a price-updating scheme for a multi-agent scheduling process where agents use combinatorial auctions for negotiation and achieve a plant-wide solution. Sarabia et al. (2013) used price-based coordination to distribute a scarce resource to various process units. The problem is divided into multiple unit-level control problems, and the coordinator is used to calculate prices for resolving the process inter-dependencies. Several case studies are presented to demonstrate that the proposed framework improves the process operation and provides a near-centralized solution. In previous work (Ahmed and Vilkkö, 2023), we presented a scheduling framework for the copper smelting process that uses price-based coordination to resolve part of the process inter-dependencies. We showed that the proposed framework is suitable for a copper smelting process in which a forced shutdown or rigorous scheduling of units results in severe consequences. Two cases are presented to demonstrate its effectiveness.

3. Copper smelting process

A schematic flow diagram of the copper smelting process is shown in Fig. 1. The input concentrates with a variable feed rate are fed to the FSF, which removes undesired elements from the concentrates in the presence of oxygen to produce slag, gases, and matte. The slag is periodically removed and brought to the slag processing unit, while the gases are moved to the gas processing unit. The FSF produces matte of a specific grade, which is usually defined by the process personnel beforehand.

Matte is moved periodically to the PSC for further processing. The matte transfer is carried by a crane, which is placed adjacent to the FSF and PSC units. Usually, several PSC units operate in parallel. During the PSC operation, the matte passes through multiple slag-blowing stages to oxidize the iron and sulphur from the matte in the presence of

oxygen (Davenport et al., 2002). Each slag-blowing stage produces slag, which is removed from the PSC and shifted to the slag processing unit at the end of the slag-blowing stage. Following the slag removal, more matte is added to the PSC. After oxidizing the maximum amount of iron from the matte, the copper-blowing stage begins, and it continues until the maximum amount of sulphur remains in the matte. The final matte, which is generally known as white metal, is transferred to the anode furnace for further processing. At this movement, the PSC is ready to produce the next batch.

During the slag-blowing stage, copper is lost to the slag. As the iron content in the matte reaches a zero level, this copper loss starts to increase exponentially (Tan, 2007). Furthermore, slag-blowing operations are highly exothermic reactions; therefore, the temperature inside the PSC must remain within its limits (Davenport et al., 2002; Harjunkoski et al., 2008). Consequently, the maximum duration of slag-blowing operations is pre-defined. Nevertheless, the final slag-blowing operation persists until the maximum amount of iron is oxidized from the matte. For maintaining the temperature within its operational limits during the final slag-blowing operation, some coolants (e.g., scrape from the last batch) are added to the PSC (Davenport et al., 2002; Ahmed et al., 2021).

4. Problem statement

This study considers a relatively simple copper smelting process, as shown in Fig. 2. The real copper smelter has more units and complex dynamics; however, the proposed formulation still reflects the important aspects of a copper smelting process.

Given: A copper smelting process consists of one FSF and multiple PSC units only. The FSF processes the same type of concentrates continuously without any break. It is assumed that the FSF never shut down, and any such attempt leads to fatal consequences. The FSF has a predefined limited storage capacity. Matte is transferred from the FSF to a PSC unit using the installed crane. The matte inside a PSC unit follows a predefined scheduling recipe.

Goal: The aim is to develop a hierarchical scheduling framework for copper smelting to operate the FSF at the optimal operating point and provides a PSC schedule with a shorter batch time and minimal copper losses. The motivation is to show that price-based coordination has the potential to provide an optimal operation of the process by allowing the process units to make their own decisions for resolving process inter-dependencies.

4.1. Inter-dependencies

In this study, FSF and PSC units give rise to the inter-dependencies. These inter-dependencies arise as a result of violations in storage, operational, and logistics constraints that commonly occur during the copper smelting process. The following section provides a detailed discussion of these inter-dependencies.

FSF inter-dependencies:

The FSF inter-dependencies arise due to its storage capacity violations. If the matte production in the FSF is high due to lower matte grade selection or higher input concentrates feed rate, and the PSC units do not process the matte as speedily due to their slow functioning or some other technical issue, the matte in the FSF reaches to its maximum value which generates inter-dependencies. These inter-dependencies will be referred to as the FSF upper-level inter-dependencies. On the other hand, if PSC units are performing faster or the FSF matte production is low, matte in the FSF will fall below its lower level. As the matte in the FSF must remain above the minimum threshold due to operational concerns, inter-dependencies will arise if the matte in the FSF falls below its minimum threshold. These inter-dependencies will be referred to as FSF lower-level inter-dependencies.

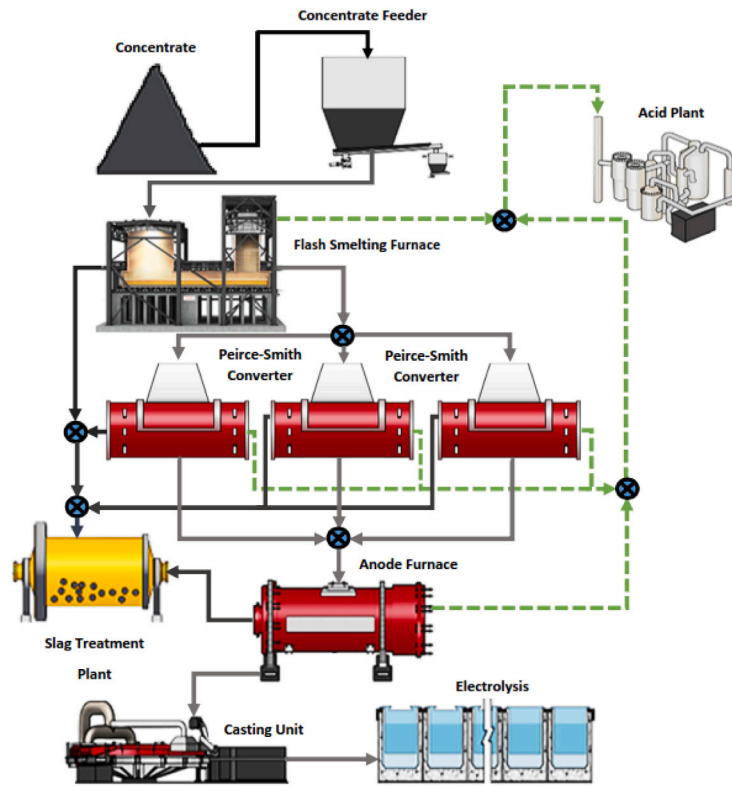


Fig. 1. Copper smelting process
 Source: adapted from Ahmed et al. (2022).

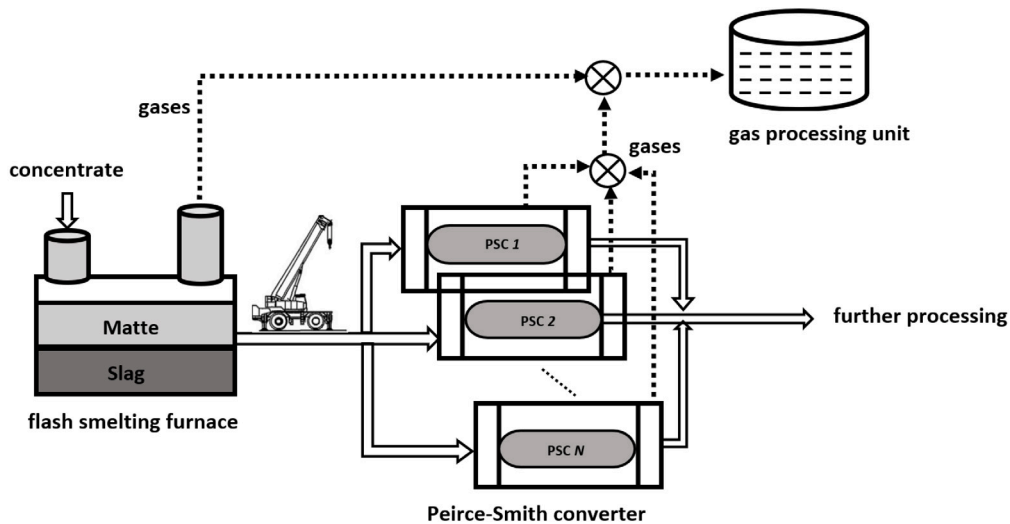


Fig. 2. Simplified copper smelting process
 Source: Adapted from Ahmed and Vilkkö (2023).

PSC inter-dependencies:

PSC inter-dependencies arise due to the parallel operation of the PSC units. During the copper smelting operation, matte is transferred from the FSF to PSC units using the installed crane. This crane can shift only a fixed amount of matte from the FSF to a PSC unit in a given time. Since all PSC units operate independently, two or more PSC units can simultaneously request matte loading, which generates PSC inter-dependencies. In this study, such inter-dependencies are identified as logistical inter-dependencies.

PSC units are prioritized based on their availability in terms of time. The unit that becomes available first is assigned the highest priority,

followed by the next earliest available unit, and the one that becomes available latest is assigned the lowest priority. For keeping the blister copper batch production synchronized, matte loading onto a PSC unit is carried out only after all loading operations on the high-priority PSC units have been successfully completed. Consequently, logistical inter-dependencies arise when a PSC unit requests matte loading before a high-priority PSC unit

The FSF and PSC units produce gases during their operation. The FSF produces gases continuously since it operates uninterruptedly, while PSC units produce gases during the slag-blowing and copper-blowing stages. All the gases are transferred to the gas processing

unit using the installed pipelines, which have limited transfer capacity. Since the FSF is operating continuously and stopping the FSF could result in serious consequences, PSC units are scheduled accordingly to respect the gas pipeline's capacity constraint. Therefore, FSF flow inter-dependencies arise when two or more PSC units are in the slag and copper blow stage in parallel.

4.2. Assumptions

The copper smelting process has strong thermodynamics; therefore, thermodynamical factors affect its operation. As the objective is not to study this process thermodynamics, several assumptions are made in the study, which are listed below.

- The FSF always processes the same type of concentrates.
- This study does not focus on any particular FSF; therefore, the FSF storage capacity limits are selected arbitrarily.
- The FSF never shut down, and any such action will result in serious consequences.
- The matte grade is defined by the process personnel beforehand and remains unchanged during the process operation.
- FSF matte production depends on the matte grade and input concentrates feed rate.
- All PSC units are the same in dimensions, follow the same scheduling recipe, and do not require maintenance breaks.
- All the PSC units have a priority number. The PSC unit available earliest will have the highest priority, and the latest PSC unit will have the lowest priority. The batch on the highest priority in the PSC units will begin its operation first, followed by the batch on the second-highest priority, and so on.
- To resolve logistical inter-dependencies, low-priority PSC units must wait until all the loading operations to all the high-priority PSC units are executed successfully.
- If logistical and flow inter-dependencies exist simultaneously, logistical inter-dependencies are addressed prior to the flow inter-dependencies.
- To resolve flow inter-dependencies, only one PSC unit must be in the slag or copper blow stage.
- Oxygen affects the oxidation rate of the unwanted elements in the copper smelting process. As the objective of this study is not to study the thermodynamics of the process, this study assumes that the oxygen supply remains unchanged during process operation. Therefore, the oxidation rate of unwanted elements remains the same.
- The temperature inside the PSC remains within its operating limits by adjusting the oxygen flow. As the oxygen supply does not change during the process operation, the temperature does not affect the FSF or PSC performance.
- It is assumed that the temperature always remains within its operating limits; therefore, no coolants are added to the PSC.
- PSC units always produce the same type of slag.
- Slag treatment and acid processing units have unlimited storage capacity; therefore, they are ignored.
- All the units are functioning independently with no interaction between them.

This study focuses mainly on designing a scheduling framework of the copper smelting process which is independent of thermodynamic and other operational factors; therefore, the proposed assumptions do not affect the applicability of the proposed framework.

5. Framework formulation

The schematic diagram of the proposed hierarchical framework is shown in Fig. 3. The hierarchical scheduling is based on discrete-time linear MILP techniques and is composed of the FSF model, the PSC

model, and a coordinator. These models are discussed in more detail here.

5.1. FSF model

The input concentrates are fed to the FSF and converted to slag, gases, and matte with a specific grade. Typically, this matte grade is defined by process personnel beforehand. The FSF operates continuously, and its storage capacity limits are defined in advance.

FSF matte production depends on the matte grade mg and input concentrate feed rate FSF_{FR}^h . This production decreases with the increase in the mg as more time is required to remove the unwanted elements from the matte, while it increases with the increase in the input feed rate as more concentrate is available. The FSF matte production is given in Eq. (1), and the amount of matte in the FSF is given by Eq. (2). In the FSF, the typical objective is to increase its matte production. As the matte grade remains unchanged during the process operation, maximization of the input concentrate is the obvious objective of the FSF, as given in Eq. (3).

$$FSF_P^h = (a \times mg + c) + (b \times FSF_{FR}^h + d) \quad \forall h \in FSF_{ULT}^h, \{a, b, c, d\} \in R^+ \quad (1)$$

$$FSF_M^h = FSF_M^{h=0} + FSF_M^{h-1} + FSF_P^h \quad \forall h \in FSF_{ULT}^h \quad (2)$$

$$\begin{aligned} \max_{FSF_{FR}} \sum_{h=1}^H FSF_{FR}^h + \sum_{h=FSF_{ULT}^h}^H [UL_{price}^h \times (PSC_{MD}^h + FSF_{UL}^h - FSF_M^h)] \quad (3) \\ feed_{min} \leq FSF_{FR}^h \leq feed_{max} \end{aligned}$$

During the simulation, the FSF model receives time instances at which FSF upper-level inter-dependencies exist FSF_{ULT}^h , price UL_{price}^h , total matte demand by the PSC units PSC_{MD}^h , and the FSF upper capacity limit UL_{FSF}^h from the coordinator. However, as no FSF upper-level inter-dependencies exist at the beginning of the simulation, the framework initializes the FSF_{ULT}^h with all the time instances ($FSF_{ULT}^h = h$). If the framework finds FSF upper capacity limit violations during an iteration k ($k \neq 1$), it solves the FSF model only for the time instances that are stored in FSF_{ULT}^h . However, if no FSF upper limit violations exist in the schedule during an iteration k ($k \neq 1$), the framework will use its latest available solution. After solving the FSF model, it returns the FSF matte FSF_M^h and feed rate FSF_{FR}^h to the coordinator. At this point, the FSF model is waiting for further guidance from the coordinator.

5.2. Batch problem

This study considers N independent PSC units producing B number of batches; therefore, the framework solves a maximum of $N \times B$ independent PSC scheduling problems during an iteration. For simplicity, this scheduling problem is referred to as *batch problem*.

In a batch problem, only one operation is carried out at a time, as given in Eqs. (4). For scheduling operations in a batch problem, this study uses two variable sets. The first variable set, $b_{z_i}^t$, is used to identify the occurrence of an internal operation, while the second variable set, $s_{z_i}^t$, ensures that operations inside a batch problem are performed according to the predefined scheduling recipe. For scheduling of internal operations, Eqs. (5)–(6) are used. Here, n_{z_i} represents the position of operation z_i in set Z ($\forall z_i \in Z$). The processing time of an operation is either fixed, or its optimal value is computed by the batch problem as given in Eqs. (7)–(8).

$$\sum_{z_i \in operation} b_{z_i}^t \leq 1 \quad (4)$$

$$s_{z_i}^t = s_{z_i}^{t-1} + b_{z_i}^t \quad (5)$$

$$\sum_{z_i \in operation} \sum_{t=1}^T s_{z_i}^t \geq (n_{z_i} \times b_{z_i}^t) \quad z_i^t = \text{operation preceding } z_i \quad (6)$$

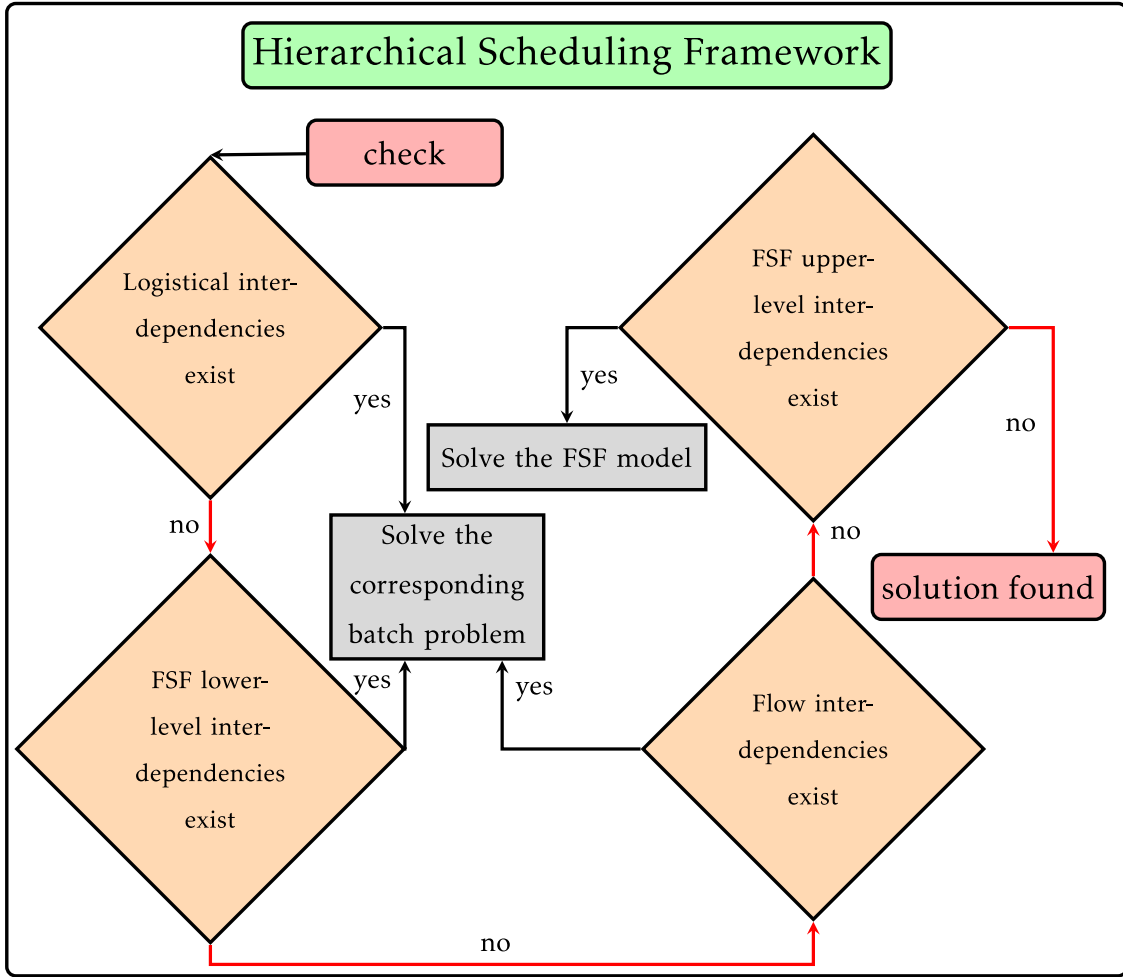


Fig. 3. Hierarchical scheduling framework.

$$\sum_{t=1}^T b_{z_i}^t = fix_{z_i} \quad \forall z_i = ML_i, SK_i \quad (7)$$

$$\min_{z_i} \leq \sum_{t=1}^T b_{z_i}^t \leq \max_{z_i} \quad \forall z_i = SB_i, CB \quad (8)$$

During the PSC operation, the amount of matte present in the PSC and the impurities in this matte are calculated using Eqs. (9) and (10). During the PSC operation, copper loses to the slag, and its amount in the slag is calculated using Eq. (11). To minimize copper losses, this study uses a simple and efficient scheme, which has been proposed in our previous work [for details, check (Ahmed et al., 2021)].

The slag-blowing and copper-blowing timings are stored in BP_{blow}^t , as given in Eq. (12). The batch problem demand for the matte is calculated using Eq. (13). Furthermore, the batch problem operation times are stored in BP_{idle}^t using Eq. (14). This BP_{idle}^t is used as a penalty term to reduce unnecessary idle times in the schedule.

During the simulation, batch problems receive three sets of information from the coordinator. The first set consists of the earliest available time $time_{LA}$ to perform loading operations and the price $load_{price}^{n,b,h}$ to address PSC logistical inter-dependencies. The second set consists of the earliest available time $time_{FA}$ to execute slag or copper-blowing operations and the price $flow_{price}^{n,b,h}$ to address the PSC flow inter-dependencies. The last set includes the earliest available time FSF_{LT} to perform the loading operation without generating FSF lower-level inter-dependencies, the matte requested by the batch problems on

all other PSC units FSF_{MDO}^h and the price LL_{price}^h to resolve the FSF lower-level inter-dependencies at a batch level.

$$BP_{mass}^t = BP_{mass}^{t-1} + \sum_{z_i \in operation} MT_{fix} \times b_{z_i}^t - \sum_{z_i \in operation} (Cu_{loss_{z_i}} + r_{ele}) \times b_{z_i}^t \quad (9)$$

$$\forall z_i \in ML_i, z_i \in \{SB_i, CB\}$$

$$BP_{elm}^t = BP_{elm}^{t-1} + \sum_{z_i \in operation} f[(mg)] \times b_{z_i}^t - \sum_{z_i \in operation} (Cu_{loss_{z_i}} + r_{ele}) \times b_{z_i}^t \quad (10)$$

$$\forall z_i \in ML_i, z_i \in \{SB_i, CB\}$$

$$BP_{Cu}^t = BP_{Cu}^{t-1} + \sum_{z_i \in operation} Cu_{loss_{z_i}} \times b_{z_i}^t \quad \forall z_i \in SB_i \quad (11)$$

$$BP_{blow}^t = \sum_{z_i \in operation} b_{z_i}^t \quad \forall z_i \in \{SB_i, CB\} \quad (12)$$

$$BP_{MD}^t = BP_{MD}^{t-1} + \sum_{z_i \in operation} MT_{fix} \times b_{z_i}^t \quad \forall z_i \in ML_i \quad (13)$$

$$BP_{idle}^t = \sum_{z_i \in operation} b_{z_i}^t \quad \forall z_i \in Z \quad (14)$$

For addressing inter-dependencies, each batch problem uses Eqs. (15)–(22) to retrieve the corresponding prices, available times to perform loading, slag, or copper-blowing operations, the relevant matte demand of the batch problems on other PSC units, and the

corresponding FSF lower capacity limits from the information that it received from the coordinator. How the coordinator calculates the given information is discussed in Section 5.3.

The objective of the batch problem is to minimize the copper losses, impurities in the matte, idle time in the schedule, and resolve inter-dependencies, as given in Eq. (23). The batch problem treats all the prices as penalty terms. Based on the calculated price values by the coordinator, the batch problem reschedules its loading times, slag-blowing, or copper-blowing times to resolve the process inter-dependencies.

$$PLB_{PSC}^t = \begin{cases} load_{price}^{n,b,h} & t = (h - BP_{start}^{n,b}) \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

$$PFB_{PSC}^t = \begin{cases} flow_{price}^{n,b,h} & t = (h - BP_{start}^{n,b}) \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

$$PLB_{FSF}^t = \begin{cases} LL_{price}^h & t = (h - BP_{start}^{n,b}) \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

$$LT_{BP} = time_{LA} - BP_{start}^{n,b} \quad (18)$$

$$FT_{BP} = time_{FA} - BP_{start}^{n,b} \quad (19)$$

$$LT_{FSF} = FSF_{LT} - BP_{start}^{n,b} \quad (20)$$

$$MDOB_{FSF}^t = \begin{cases} FSF_{MDO}^h & t = (h - BP_{start}^{n,b}) \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

$$LLB_{FSF}^t = \begin{cases} FSF_{LL}^h & t = (h - BP_{start}^{n,b}) \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

$$\begin{aligned} \min_{BP_{Cu}, BP_{ele}, Curatio, BP_{idle}} & \sum_{t=1}^T BP_{Cu}^t + \sum_{t=1}^T BP_{ele}^t + \sum_{t=1}^T BP_{Cu}^t + \sum_{t=1}^T BP_{idle}^t \\ & + \sum_{t \leq LT_{FSF}} PBL_{FSF}^t \times (MDOB_{FSF}^t + BP_{MD}^t - LLB_{FSF}^t) \\ & + \sum_{t \leq LT_{BP}} PLB_{PSC}^t \times [(LT_{BP} \times s_{ML_1}^t) - LT_{BP}] \\ & + \sum_{t \leq FT_{BP}} PFB_{PSC}^t \times [(FT_{BP} \times BP_{blow}^t) - FT_{BP}] \end{aligned} \quad (23)$$

At the end of the iteration, each problem uses Eqs. (24)–(28) to determine loading times $BP_{load_1}^{n,b}$, $BP_{load_I}^{n,b}$, slag and copper-blowing times $BP_{blow}^{n,b,t}$, the batch end time $BP_{end}^{n,b}$, and matte demand $BP_{MD}^{n,b,t}$, and send them to the coordinator. After this, the batch problem is waiting for further instruction from the coordinator.

$$BP_{load_1}^{n,b} = \begin{cases} t & b_{z_i}^{n,b,t} = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall z_i = ML_1 \quad (24)$$

$$BP_{load_I}^{n,b} = \begin{cases} t & b_{z_i}^{n,b,t} = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall z_i = ML_I \quad (25)$$

$$BP_{blow}^{n,b,t} = \begin{cases} t & b_{z_i}^{n,b,t} = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall z_i = \{SB_1, CB\} \quad (26)$$

$$BP_{end}^{n,b} = \begin{cases} t + BP_{start}^{n,b} & b_{z_i}^{n,b,t} = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall z_i = BE \quad (27)$$

$$BP_{MD}^{n,b,t} = BP_{MD}^t \quad (28)$$

5.3. Coordinator

The heart of this study is the coordination mechanism between the FSF and the PSC units. The coordinator is responsible for solving all inter-dependencies between the process units and providing a feasible schedule using a price updating scheme.

In the proposed scheduling framework, the FSF and PSC units are the bottleneck for the process. During the process operation, matte is produced by the FSF and consumed by the PSC units. If the matte in the FSF violates its upper or lower storage limits, FSF inter-dependencies

will arise. Therefore, this study considers the matte as a scarce resource that must be distributed optimally to the PSC units to guarantee equilibrium between supply and demand. PSC units are the bottleneck due to the timing of PSC operations. As discussed in Section 4.1, PSC inter-dependencies exist in the schedule when a unit requests matte loading or blowing at a time that conflicts with the scheduling policy. Therefore, the operation time of the PSC unit is a scarce resource, which generates PSC inter-dependencies. By considering PSC operation time as a scarce resource, this study will guarantee that the batch problems execute their operations according to the scheduling recipe.

For the optimal distribution of shared resources, this study utilizes the principles of free market theory. In an open market, prices of commodities are adjusted continuously to maintain an equilibrium between supply and demand. Ideally, the supply and demand of a scarce resource should be equal. If so, the price of the resource remains unchanged. But in actual markets, the supply and demand of a scarce resource keep fluctuating, and so do their prices. If a scarce resource is in excess, market forces will reduce its price to increase market demand. On the contrary, if demand exceeds the supply, the price is raised to lower its demand. This study leverages market phenomena by employing a price adjustment scheme for the optimal distribution of scarce resources and provides an optimal schedule.

At the start of the simulation, the coordinator initializes the FSF_{ULT}^h with all the scheduling time instances ($FSF_{ULT}^h = h$) and sends it the FSF model. The FSF model solves the scheduling problem and returns the FSF matte and feed rate trajectories to the coordinator. Furthermore, it assigns priorities to the PSC unit using Eq. (29). After each iteration, the coordinator receives loading timings, blowing timings, and start and end times from all the batch problems.

$$prior_{PSC}^n = \begin{cases} highest & \text{unit } n \text{ has the lowest } start_{PSC}^n \\ highest - 1 & \text{unit } n \text{ has the second lowest } start_{PSC}^n \\ \vdots & \\ lowest & \text{unit } n \text{ has the highest } start_{PSC}^n \end{cases} \quad (29)$$

To assist in the inter-dependencies management and to reduce the computational demand, the coordinator divides all the batch problems on all the PSC units into different groups. The grouping is carried out such that the first batch problem on each PSC unit is assigned to Group 1, the second batch on each PSC unit to Group 2, and the final batch problem on each PSC unit to Group B , as shown in Fig. 4. However, if the start time of the batch problem is higher than the maximum end time of the batch problem within a group, the batch problem is not added to the given group but to the following one if the batch start time is less than the maximum end time of the batch in the next group. The grouping of the batch problems allows the framework to examine and solves the inter-dependencies in ascending order of the group number and produces the batches in a synchronized manner.

Logistical inter-dependencies exist in a group if a batch problem on a low-priority PSC unit begins its operation before the batch problems on the preceding high-priority PSC units. Thus, the coordinator uses the $BP_{load_1}^{n,b}$ and $BP_{load_I}^{n,b}$ to determine batch problems that are the source of the logistical inter-dependencies, as given in Eqs. (30). It assigns a value of 1 to batch problem b on the PSC unit n if loading operations are not performed according to the scheduling recipe. As PSC inter-dependencies can exist between the groups, the coordinator uses Eq. (31) to determine if logistical inter-dependencies present between the batch problems of the current and the following group. Similarly, the coordinator uses Eq. (32) to calculate the blowing times for the entire scheduling horizon and then identifies batch problems that generate flow inter-dependencies within a group or between the groups, as given in Eqs. (33)–(34). In addition, the coordinator rearranges the matte demand of the batch problems over the entire scheduling horizon using Eq. (35). Moreover, the coordinator updates the start time of each batch problem using Eq. (36). At this point, the first step of the

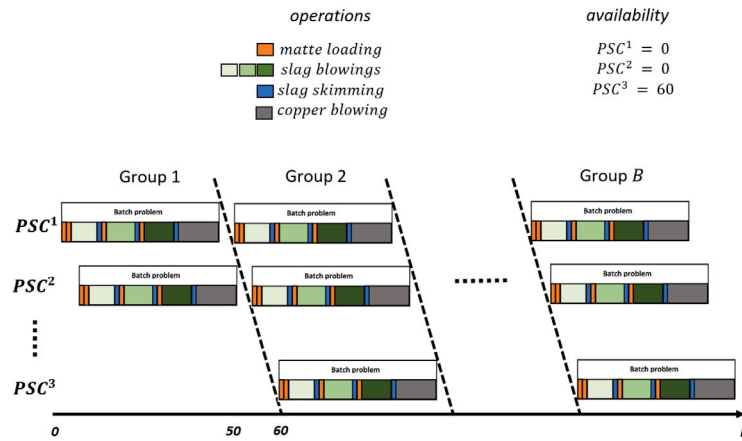


Fig. 4. PSC units availability
Source: Adapted from Ahmed and Vilkkö (2023).

coordination task is completed.

$$PSC_{LID}^{n,b} = \begin{cases} 1 & (BP_{start}^{\bar{n},b} + BP_{load_1}^{\bar{n},b}) \geq (BP_{start}^{n,b} + BP_{load_1}^{n,b}) \\ & \forall \{\bar{n}, \bar{b}\} \subseteq n, prior_{PSC}^{\bar{n}} < prior_{PSC}^{\bar{n}} \\ 0 & \text{otherwise} \end{cases} \quad (30)$$

$$PSC_{LBG}^{n,b} = \begin{cases} 1 & (BP_{start}^{\bar{n},b} + BP_{load_1}^{\bar{n},b}) \geq (BP_{start}^{n,b+1} + BP_{load_1}^{n,b+1}) \\ & \forall \sum_{n=1}^N PSC_{LID}^{n,b} = 0, prior_{PSC}^{\bar{n}} = \text{lowest} \\ 0 & \text{otherwise} \end{cases} \quad (31)$$

$$PSC_{FA}^{n,b,h} = \begin{cases} 1 & h = (BP_{start}^{n,b} + BP_{blow}^{n,b,t}) \quad \forall BP_{blow}^{n,b,t} \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (32)$$

$$PSC_{FID}^{n,b} = \begin{cases} 1 & (PSC_{FA}^{\bar{n},b,h} + PSC_{FA}^{\bar{n},b,h}) > 1 \\ & \forall \{\bar{n}, \bar{b}\} \subseteq n, prior_{PSC}^{\bar{n}} < prior_{PSC}^{\bar{n}} \\ 0 & \text{otherwise} \end{cases} \quad (33)$$

$$PSC_{FBG}^{n,b} = \begin{cases} 1 & (PSC_{FA}^{\bar{n},b,h} + PSC_{FA}^{n,b+1,h}) > 1 \\ & \forall BP_{start}^{n,b} \leq h \leq BP_{end}^{n,b+1}, prior_{PSC}^{\bar{n}} = \text{lowest} \\ 0 & \text{otherwise} \end{cases} \quad (34)$$

$$PSC_{MD}^h = \sum_{n=1}^N \sum_{b=1}^B BP_{MD}^{n,b,t} \quad \forall h = BP_{start}^{n,b} + t \quad (35)$$

$$BP_{start}^{n,b} = BP_{end}^{n,b-1} + 1 \quad (36)$$

The second step of the coordinator dealt with identifying the type of inter-dependencies and taking the necessary measures to solve them. How the coordinator manages these inter-dependencies is discussed in turn.

5.4. PSC-inter-dependencies

PSC inter-dependencies exist in the schedule due to the violation of logistical or flow constraints. Therefore, the coordinator uses Eqs. (37) and (38) to see if logistical or flow inter-dependencies exist in the schedule. If such inter-dependencies present in the schedule, the coordinator assigns a value of 1 to GP_{load}^b or GP_{flow}^b for each conflicting group b .

$$GP_{load}^b = \begin{cases} 1 & \sum_{n=1}^N PSC_{LID}^{n,b} > 1 \text{ or } \sum_{n=1}^N PSC_{LBG}^{n,b} > 1 \\ 0 & \text{otherwise} \end{cases} \quad (37)$$

$$GP_{flow}^b = \begin{cases} 1 & \sum_{n=1}^N PSC_{FID}^{n,b} > 1 \text{ or } \sum_{n=1}^N PSC_{FBG}^{n,b} > 1 \\ 0 & \text{otherwise} \end{cases} \quad (38)$$

The coordinator assigns a value of 1 to is_{PSC} if PSC inter-dependencies exist in the schedule; otherwise, it will assign a value of 0, as given in Eq. (39). A value of 0 means that the coordinator will not examine the schedule against PSC inter-dependencies and will check for FSF inter-dependencies only.

The coordinator addresses PSC inter-dependencies in ascending order of the group number, and it uses Eqs. (40) and (41) to determine the foremost group number that has the PSC inter-dependencies. When a group has both logistical and flow inter-dependencies, it is advisable for the coordinator to first address the logistical inter-dependencies. The reason for this is that solving logistical inter-dependencies can potentially resolve some of the flow inter-dependencies, thus reducing the computational costs of the framework.

$$is_{PSC} = \begin{cases} 1 & \sum_{b=1}^B GP_{load}^b \geq 1 \text{ or } \sum_{b=1}^B GP_{flow}^b \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (39)$$

$$LG = \begin{cases} b & GP_{load}^b = 1 \text{ and } LG = 0 \\ 0 & \text{otherwise} \end{cases} \quad (40)$$

$$FG = \begin{cases} b & GP_{flow}^b = 1 \text{ and } FG = 0 \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

Logistical inter-dependencies

Logistical inter-dependencies could exist in the schedule at multiple points. If a group has multiple logistical inter-dependencies, resolving all of them at once means solving several batch problems simultaneously. This simultaneous solution of the batch problems may not guarantee a feasible solution as it ignores the multilateral effects while solving the batch problems. Therefore, addressing all the logistical inter-dependencies could lock up the framework between two local solutions that could prevent it to return a feasible solution. For this reason, this study intends to solve logistical inter-dependencies affiliated with a single batch problem during a single iteration.

To address logistical inter-dependencies in a group, the coordinator uses Algorithm 1 that consists of three steps. In the first step, the coordinator determines all the conflicting PSC units, as represented by PSC_{conf}^n . From these units, it identifies the highest priority and the lowest priority PSC unit and stores them in PSC_{LU} and PSC_{LL} . As priorities of the PSC units determine the order of their operation, logistical inter-dependencies in a group are addressed by rescheduling the batch problem on the lower priority PSC unit, as mentioned in Section 4.2.

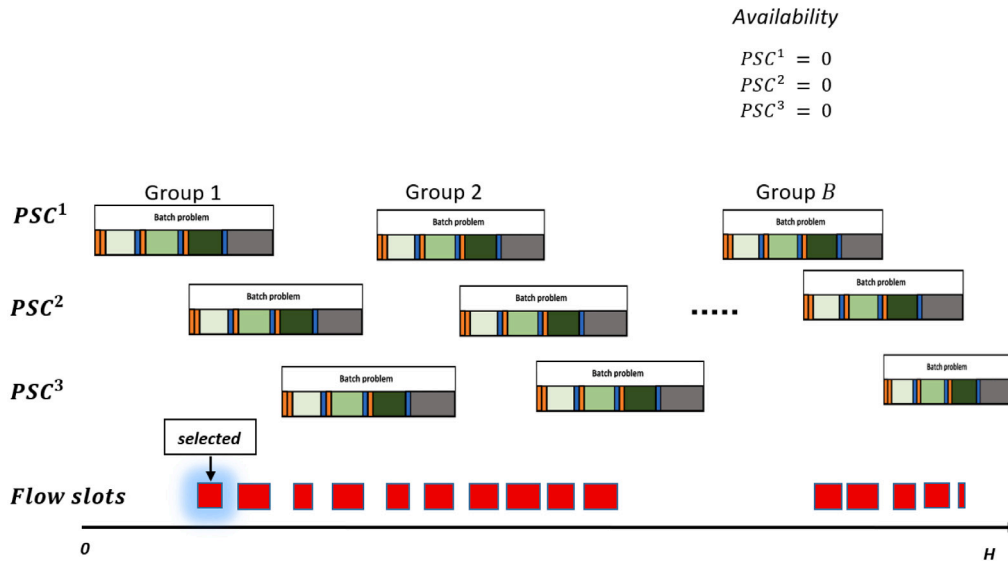


Fig. 5. Flow slots formulation.

Logistical inter-dependencies arise due to the wrong time of the loading operations of the batch problem on the PSC unit PSC_{LL} . Therefore, the coordinator uses the second step of the Algorithm 1 to calculate the time between the start and the last loading operation of the batch problem on the PSC unit PSC_{LU} and store them in $time_{LD}^{n,b,h}$ ($n = PSC_{LL}$). The $time_{LD}^{n,b,h}$ contains the time instances at which the batch problem on the PSC unit PSC_{LL} must not execute loading operations to resolve the logistical inter-dependencies between PSC units. In addition, the coordinator calculates the earliest available time at which the batch problem on the PSC unit PSC_{LL} can start its loading operation without generating logistical inter-dependencies. This earliest available time is always the time instant next to the last loading operation of the batch problem on the PSC unit PSC_{LU} , and it is stored in $time_{LA}$. As PSC operation time is a scarce resource, the coordinator calculates the difference between the available and the demanded loading times for the batch problem on the PSC unit PSC_{LL} and stores it in $time_{LDF}^{n,b,h}$. At this point, the second step is completed.

Logistical inter-dependencies exist between two groups when a batch problem in a succeeding group begins its loading operation before the loading operations to all the batch problems in Group LG . As the framework objective is to produce PSC batches in a synchronized manner, the batch problems in the succeeding groups must wait until the loading operations of the batch problems in the Group LG are executed successfully. Therefore, after addressing all the logistical inter-dependencies in Group LG , the coordinator uses the third step of Algorithm 1 to learn if inter-group logistical inter-dependencies exist in the schedule.

To address inter-group logistical inter-dependencies, the coordinator calculates all the conflicting time instances between the Group LG and the succeeding group and saves it in $time_{LD}^{n,b+1,h}$, if exist any. If inter-group logistical inter-dependencies exist, it calculates the earliest available time at which the batch problems in the succeeding group can freely begin their loading operations without generating logistical inter-dependencies. After that, the coordinator calculates the difference between the available and demanded time for the loading operations of the batch problems in the succeeding group, which is stored in $time_{LDF}^{n,b+1,h}$.

For monitoring logistical inter-dependencies, the coordinator saves the conflicting time instances $time_{LD}^{n,b,h,k}$ during the iteration k , as given in Eq. (42). Addressing logistical inter-dependencies means rescheduling a batch problem from the beginning. Therefore, if a batch problem in the group LG is the source of the logistical inter-dependencies

which was also the source of flow inter-dependencies during a previous iteration, the coordinator will reset to zero the flow inter-dependencies parameters ($flow_{price}^{n,b,h} = 0, time_{FA} = 0 \forall b \in FG$).

$$time_{LDF}^{n,b,h,k} = \begin{cases} time_{LD}^{n,b,h} & k + + \\ time_{LD}^{n,b,h,k} & \text{otherwise} \end{cases} \quad (42)$$

To address the logistical inter-dependencies, the final task of the coordinator is to use a PI controller to calculate the prices. This controller considers both the present and past conflicting time instances, as given in Eq. (43). Using the PI controller, the coordinator can more effectively manage the price updating mechanism and ensure that the optimal prices are founded with less computational demand.

$$load_{price}^{n,b,h} = (P_{load} \times time_{LDF}^{n,b,h}) + (I_{load} \times \sum_{k=1}^{k-1} time_{LDF}^{n,b,h,k}) \quad \forall b = LG \quad (43)$$

The coordinator sends the $load_{price}^{n,b,h}$ and the $time_{LA}$ to the batch problems. The corresponding batch problem solves its optimization problems and returns the solution to the coordinator. If logistical inter-dependencies still exist in a group LG , the coordinator uses the same procedure to calculate the new price $load_{price}^{n,b,h}$ and performs a new iteration. Otherwise, it examines the flow inter-dependencies or FSF inter-dependencies in the group.

Flow inter-dependencies

Flow inter-dependencies exist in the schedule when two or more batch problems are in the slag-blowing or copper-blowing stage simultaneously. Flow inter-dependencies can exist within a group or between consecutive groups. Resolving all the flow inter-dependencies simultaneously ignores the interconnection between the conflicting batch problems that affect the quality of the solution. Therefore, the coordinator will only address the flow inter-dependencies of the batch problem on the highest priority PSC unit among the conflicting units within a group during a single iteration. However, for inter-group flow inter-dependencies, the conflicting batch problems of the succeeding group are solved simultaneously to address the flow inter-dependencies between the current and the succeeding group. The approach that the coordinator uses to resolve flow inter-dependencies is presented in Algorithm 2.

The coordinator divides all the flow inter-dependencies into different flow slots. These flow slots are made such that each flow slot contains consecutive time instances with flow inter-dependencies, as shown in Fig. 5. The framework has the potential to tackle single or

Algorithm 1: Logistical inter-dependencies**Begin****First step**

1: Initialization

 $PSC_{conf}^n = 0$ $PSC_{LL}^n = 0$ $PSC_{LU}^n = 0$ 2: for $b = 1$ to B do for $n = 1$ to N do if ($b = LG$ and $PSC_{LID}^{n,b} = 1$) then $PSC_{conf}^n = 1$

else

 $PSC_{conf}^n = 0$

end

end

end

3: for $n = 1$ to N do if ($PSC_{conf}^n = 1$) then if $prior_{PSC}^n > prior_{PSC}^{n-1}$ then $PSC_{LU}^n = n$

end

end

end

4: for $n = 1$ to N do if ($PSC_{conf}^n = 1$ and $PSC_{conf}^n \neq PSC_{LU}^n$) then if $prior_{PSC}^n < prior_{PSC}^{n-1}$ then $PSC_{LL}^n = n$

end

end

end

Second step1: for $b = 1$ to B do for $\bar{n} = 1$ to N do for $\bar{n} = 1$ to N do if ($b = LG$, $\bar{n} = PSC_{LU}$ and $\bar{n} = PSC_{LL}$) then for $h = 1$ to H do if ($BP_{start}^{n,b} \leq h \leq (BP_{start}^{\bar{n},b} + BP_{load_1}^{\bar{n},b})$) then $time_{LD}^{n,b,h} = h$

else

 $time_{LD}^{n,b,h} = 0$

end

end

end

end

end

end

2: for $b = 1$ to B do for $n = 1$ to N do if ($b = LG$ and $n = PSC_{LU}$) then $time_{LA} = BP_{start}^{n,b} + BP_{load_1}^{n,b} + 1$

else

 $time_{LA} = 0$

end

end

end

3: for $b = 1$ to B do for $n = 1$ to N do if ($b = LG$ and $n = PSC_{LU}$) then $time_{LDF}^{n,b,h} = time_{LA} - time_{LD}^{n,b,h}$

end

end

end

Third step1: for $b = 1$ to B do for $n = 1$ to N do if ($b = LG$ and $PSC_{LID}^{n,b} = 1$) then for $h = 1$ to H do if ($\bar{n} = \min(prior_{PSC}^n)$, $\bar{n} = \max(prior_{PSC}^n)$) then

if

 ($(BP_{start}^{n,b} + BP_{load_1}^{n,b}) \leq h \leq (BP_{start}^{\bar{n},b+1} + BP_{load_1}^{\bar{n},b+1})$)

then

 $time_{LD}^{n,b+1,h} = h$

else

 $time_{LD}^{n,b+1,h} = 0$

end

end

end

end

end

end

2: for $b = 1$ to B do for $\bar{n} = 1$ to N do if ($b = LG$ and $\bar{n} = \min(prior_{PSC}^n)$) then $time_{LA} = BP_{start}^{n,b} + BP_{load_1}^{n,b} + 1$

else

 $time_{LA} = 0$

end

end

end

3: for $b = 1$ to B do for $n = 1$ to N do if ($b = LG$) then $time_{LDF}^{n,b+1,h} = time_{LA} - time_{LD}^{n,b+1,h}$

end

end

end

end

multiple flow slots of a batch problem at the same time. Considering only one flow slot could increase the number of iterations, which the framework requires to find a feasible solution. However, it will ensure that the schedule has the minimum batch time, thus maintaining solution quality and preventing the framework from blocking between two local sub-optimal solutions. Therefore, this study will address only the foremost flow slot of a batch problem during a single iteration.

To tackle the flow inter-dependencies in the group, the coordinator uses the first step of the Algorithm 2 to find PSC units that generate flow inter-dependencies. Next, it finds out the lowest and the highest priority PSC units from the PSC_{conf}^n , which are stored in PSC_{FL} and PSC_{FU} . Subsequently, the coordinator uses the second step of the Algorithm 2 to calculate the foremost flow slot of the batch problem on the PSC_{FL} .

The $time_{FD}^{n,b,h}$ consists of time instances during which the batch problem PSC_{FU} performs its blowing operation. For resolving flow inter-dependencies, the batch problem on the unit PSC_{FL} must reschedule its blow operation such that it does not perform blowing operation at time instances that are stored in $time_{FD}^{n,b,h}$. The earliest available time at which the batch problem PSC_{FL} can begin its blowing operation is the time instant after the batch problem on the unit PSC_{FU} has completed its blowing operation. Therefore, the coordinator calculates this earliest available time and stores it in $time_{FA}$. The $time_{FA}$ is the time instant at which the batch problem on the unit PSC_{FU} ends its blowing

operation, and the batch problem PSC_{FL} can begin its blowing without generating inter-dependencies.

Similar to logistical inter-dependencies, the coordinator treats PSC blowing time as a scarce resource to resolve flow inter-dependencies. The batch problem on the unit PSC_{LU} demands time instances $time_{FD}^{n,b,h}$ for the blowing operation, but the coordinator will offer it the time $time_{FA}$ to begin its blowing operation. The coordinator calculates the difference between the available and demanded time and stores it in $time_{FDF}^{n,b,h}$.

Like the logistical inter-dependencies, flow inter-dependencies could exist between consecutive groups. Resolving inter-group logistical inter-dependencies can potentially address the flow inter-dependencies between the groups. However, when a profusion of unnecessary idle times is present in a group, dealing with only logistical inter-dependencies may not address the inter-group flow inter-dependencies. Thus, the coordinator checks for the inter-group flow inter-dependencies after resolving inter-dependencies in Group LG .

To address inter-group flow inter-dependencies, the coordinator uses the third step of the Algorithm 2 to calculate all the conflicting time instances between the Group FG and the preceding group, which are saved in $time_{FD}^{n,b+1,h}$. Next, it determines the earliest available time for the batch problems in the preceding group to begin their blowing operation without generating inter-dependencies, as represented by $time_{FA}$. For inter-group flow inter-dependencies, $time_{FA}$ is always the time next to the end time of the batch problem on the PSC unit PSC_{FL} . Furthermore, it calculates the difference between the available and the demanded time.

To monitor flow inter-dependencies during the simulation, the coordinator uses Eq. (44) to save the conflicting time instances during each iteration. Like logistical inter-dependencies, the coordinator uses the PI controller to determine optimal price, as given in Eq. (45). However, a batch problem with flow inter-dependencies may be the source of logistical inter-dependencies during a previous iteration; therefore, the coordinator shares the flow and logistical inter-dependencies parameters $flow_{price}^{n,b,h}$, $time_{FA}$, $load_{price}^{n,b,h}$, and the $time_{LA}$ with the batch problems, and performs a new iteration. The conflicting batch problem solves the scheduling problem and returns the relevant information to the coordinator. The coordinator formulates the schedule and investigates the presence of flow inter-dependencies. If such inter-dependencies exist in the schedule, it performs the same actions. Otherwise, it looks for the FSF inter-dependencies or logistical inter-dependencies in the schedule.

$$time_{FDF}^{n,b,h,k} = \begin{cases} time_{FD}^{n,b,h} & k++ \\ time_{FD}^{n,b,h,k} & \text{otherwise} \end{cases} \quad (44)$$

$$flow_{price}^{n,b,h} = (P_{flow} \times time_{FDF}^{n,b,h}) + (I_{flow} \times \sum_{k=1}^{k-1} time_{FDF}^{n,b,h,k}) \quad \forall b = FG \quad (45)$$

5.5. FSF inter-dependencies

FSF inter-dependencies arise during the simulation when the matte in the FSF violates the storage capacity limits. The FSF produces the matte, and PSC units consume it for batch production. If matte production is high due to a higher feed rate or lower matte grade selection

Algorithm 2: Flow inter-dependencies

```

Begin
  First step
  1: Initialization
  PSCnconf = 0
  PSCFL = 0
  PSCFU = 0
  stop1 = false
  stop2 = false
  
```

```

2: for b = 1 to B do
  for n = 1 to N do
    if (b = LG and PSCn,bFID = 1) then
      PSCnconf = 1
    else
      PSCnconf = 0
    end
  end
end
3: for n = 1 to N do
  if (PSCnconf = 1) then
    if (priornPSC > priorn-1PSC) then
      PSCFU = n
    end
  end
end
4: for n = 1 to N do
  if (PSCnconf = 1 and PSCnconf ≠ PSCFU) then
    if (priornPSC < priorn-1PSC) then
      PSCFL = n
    end
  end
end

Second step
1: for b = 1 to B do
  for n̄ = 1 to N do
    for n = 1 to N do
      if (b = LG, n̄ = PSCFU and n = PSCFL) then
        for h = 1 to H do
          if (PSCn̄,b,hFA + PSCn,b,h+1FA) > 1 then
            timen̄,b,hFD = h
          else
            timen̄,b,hFD = 0
          end
        end
      end
    end
  end
end
2: for b = 1 to B do
  for n = 1 to N do
    if (b = LG and n = PSCFL) then
      for h = 1 to H do
        while stop1 = false do
          if ((timen,b,hFD + timen,b,h+1FD) ≥ 1 and
              timen,b,h+1FD = 0) then
            timeFA = h + 1
            stop1 = true
          else
            timeFA = 0
          end
        end
      end
    end
  end
end
3: for b = 1 to B do
  for n = 1 to N do
    if (b = LG, and n = PSCFL) then
      timen,b,hFDF = timeFA - timen,b,hFD
    end
  end
end
end
  
```

```

Third step
1: for  $b = 1$  to  $B$  do
    for  $n = 1$  to  $N$  do
        if ( $b = LG$  and  $n = \min(\text{prior}_{PSC}^n)$ ) then
            for  $h = 1$  to  $H$  do
                if ( $BP_{start}^{n,b} \leq h \leq BP_{end}^{n,b}$ ) then
                    for  $n = 1$  to  $N$  do
                        if ( $PSC_{FA}^{n,b,h} + PSC_{FA}^{n,b+1,h} > 1$ ) then
                             $time_{FD}^{n,b+1,h} = h$ 
                        else
                             $time_{FD}^{n,b+1,h} = 0$ 
                        end
                    end
                end
            end
        end
    end
end

2: for  $b = 1$  to  $B$  do
    for  $n = 1$  to  $N$  do
        if ( $b = LG$  and  $n = \min(\text{prior}_{PSC}^n)$ ) then
            for  $h = 1$  to  $H$  do
                while  $stop_2 = \text{false}$  do
                    if ( $(time_{FD}^{n,b,h} + time_{FD}^{n,b,h+1}) \geq 1$  and  $time_{FD}^{n,b,h+1} = 0$ ) then
                         $time_{FA} = h + 1$ 
                         $stop_2 = \text{true}$ 
                    else
                         $time_{FA} = 0$ 
                    end
                end
            end
        end
    end
end

3: for  $b = 1$  to  $B$  do
    for  $n = 1$  to  $N$  do
        if ( $b = LG$ ) then
             $time_{FD}^{n,b+1,h} = time_{FA} - time_{FD}^{n,b+1,h}$ 
        end
    end
end
end
    
```

and the PSC units are not fast enough to consume the matte, the FSF upper capacity limit will reach. To address the FSF upper capacity limit violations, temporarily shutting down the FSF unit can be proposed as a possible solution. However, due to the potentially serious consequences of shutting down the FSF unit, this approach cannot be employed as a viable method of process operation. Hence, the FSF must readjust its feed rate to ensure an optimal process operation. On the other hand, if matte production is low, the FSF may not fulfill the demands of the PSC units; consequently, the FSF's lower capacity limit will be violated which will make the PSC units the bottleneck for the entire process. If mechanical and operational conditions prevent the FSF unit from increasing production, the only viable option for maintaining process operation is to reschedule the PSC units more effectively to mitigate FSF lower capacity violations.

To check the FSF inter-dependencies in the schedule, the coordinator uses the matte available and demanded trajectories (FSF_M^h and PSC_{MD}^h) and assigns a value of 1 to each capacity violating time instance, as given in Eqs. (46) and (47). In this study, the coordinator addresses the FSF upper inter-dependencies before the FSF lower inter-dependencies.

$$FSF_{ULV}^h = \begin{cases} 1 & \forall PSC_{MD}^h < (FSF_M^h - FSF_{UL}^h) \\ 0 & \text{otherwise} \end{cases} \quad (46)$$

$$FSF_{LLV}^h = \begin{cases} 1 & \forall PSC_{MD}^h > (FSF_M^h - FSF_{LL}^h) \\ 0 & \text{otherwise} \end{cases} \quad (47)$$

FSF upper-level inter-dependencies

FSF upper-level inter-dependencies arise due to its upper storage limit violation. As the FSF operates continuously, FSF upper-level inter-dependencies may exist at several time instances in the schedule. The coordinator is free to handle several or part of such inter-dependencies during a single iteration. Simultaneous handling of multiple FSF upper-level inter-dependencies has the advantage that the coordinator will address all such inter-dependencies quickly and with lower computational demand. Nevertheless, this approach may result in occasional unnecessary reductions in the FSF feed rate, thereby decreasing overall throughput. In addition, handling multiple FSF upper-level inter-dependencies simultaneously could sandwich the coordinator between two local solutions; thus, the framework will terminate without returning a feasible solution. Therefore, this study intends to address only part of the FSF upper-level inter-dependencies during a single iteration to maintain a good solution quality. Similarly to flow inter-dependencies, upper-level inter-dependencies in the FSF are also segregated into distinct time slots. The division is done to ensure that each time slot includes one or more successive time instances where conflicts arise. The coordinator will address only the first time slot from the given time slots during a single iteration.

A time slot may encompass multiple conflicting time instances. Since the FSF operates as an integrator, addressing even a portion of a time slot may suffice to resolve the entire time slot or shorten its span. While handling a fraction of a time slot during a single iteration increases the computational requirements of the framework, this approach guarantees high-quality solutions by sustaining optimal FSF performance.

To address the inter-dependencies at the upper level of the FSF, the coordinator employs a two-step approach outlined in Algorithm 3. In the first step, the coordinator uses the FSF_{ULV}^h to determine the earliest available time slot, denoted as FSF_{TS} . The coordinator then calculates a portion of the time slot, represented by FSF_{TSS} , that will be handled during the next iteration. Here, $part_{TS}$ refers to the portion of FSF_{TS} being considered. Its value is predetermined and influenced by factors such as the available computational power, the expertise of process personnel, and the optimal gap of the solver. If a time slot contains only a few time instances that are shorter than the predetermined minimum time min_{TS} , the coordinator will utilize the entire time slot FSF_{TS} . After determining the number of conflicting time instances FSF_{TSS} , the coordinator determines the times at which the FSF upper-level inter-dependencies exist, which are stored in FSF_{ULT}^h . At this point, the first step is completed.

The framework resolves the FSF upper-level inter-dependencies by adjusting the FSF feed rate. In situations, where the FSF matte production is high, reducing the feed rate only at the conflicting time instances may not address the inter-dependencies; and the coordinator must consider the time instances prior to conflicting times to resolve the FSF upper-level inter-dependencies. If so, the second step of the coordinator becomes active, where the coordinator examines the feed rate FSF_{FR}^h . If the FSF_{FR}^h at the conflicting time instances FSF_{ULT}^h is already set to its minimum value $feed_{min}$ during a previous iteration, the coordinator adds one more time to the FSF_{ULT}^h . This new time

instance always precedes the first conflicting time instance. At this point, the second stage is completed.

Algorithm 3: FSF upper-level inter-dependencies

```

Begin
First Step
1: Initialization
 $FSF_{TS} = 0$ 
 $stop_1 = false$ 
 $stop_2 = false$ 
 $extra = false$ 
2: for  $h = 1$  to  $H$  do
  while  $stop_1 = false$  do
    if  $((FSF_{ULV}^h + FSF_{ULV}^{h+1}) > 1$  and  $FSF_{ULV}^{h+1} \neq 0$ ) then
      |  $FSF_{TS} = FSF_{TS} + 1$ 
    else if  $(FSF_{TS} \neq 0)$  then
      |  $stop_1 = true$ 
    else
      | Goto step 2
    end
  end
end
3: if  $(FSF_{TS} \gg min_{TS})$  then
  |  $FSF_{TSS} = (\frac{part_{TS}}{100} \times FSF_{TS})$ 
else
  |  $FSF_{TSS} = FSF_{TS}$ 
end
4: for  $h = 1$  to  $H$  do
  while  $stop_2 = false$  do
    if  $(FSF_{ULV}^h \neq 0$  and  $FSF_{TSS} \neq 0)$  then
      |  $FSF_{ULT}^h = h$ 
      |  $FSF_{TSS} = FSF_{TSS} - 1$ 
    end
    else if  $(FSF_{TSS} = 0)$  then
      |  $stop_2 = true$ 
    else
      | Goto step 4
    end
  end
end
Second Step
1: for  $h = 1$  to  $H$  do
  if  $(FSF_{ULT}^h \neq 0, feed_{FSF}^h = feed_{min}$  and  $extra = false)$  then
    |  $FSF_{ULT}^{h-1} = h - 1$ 
    |  $extra = true$ 
  end
end
end

```

After determining the conflicting time instances, the coordinator calculates the difference between the available and demand matte, as given in Eq. (48). Like PSC inter-dependencies, the coordinator uses the PI controller to calculate the prices. As the FSF inter-dependencies could appear at the same conflicting time instances during different iterations, the coordinator records the FSF_{ULT}^h using Eq. (49). The PI controller uses the current and past conflicting time instances to calculate the prices for the FSF model, as given in Eq. (50).

The coordinator sends the UL_{price}^h , PSC_{MD}^h , and the matte upper limit FSF_{UL}^h to the FSF model. The FSF model solves the optimization problem and returns the mass and input feed rate trajectories to the coordinator. The coordinator investigates the schedule against the FSF

upper-level limit violation. If a violation is found, the coordinator executes the same actions and performs a new iteration. This exchange of information continues until all the FSF upper-level inter-dependencies are addressed in the schedule.

$$FSF_{ULD}^h = \begin{cases} FSF_M^h - PSC_{MD}^h - FSF_{UL}^h & \forall FSF_{ULT}^h \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (48)$$

$$FSF_{ULD}^{h,k} = \begin{cases} FSF_{ULD}^h & k++ \\ FSF_{ULT}^{h,k} & \text{otherwise} \end{cases} \quad (49)$$

$$UL_{price}^h = (P_{UL} \times FSF_{ULD}^h) + (I_{UL} \times \sum_{k=1}^{k-1} FSF_{ULD}^{h,k}) \quad (50)$$

FSF lower-level inter-dependencies

FSF inter-dependencies arise if the FSF matte level falls below its lower capacity limit. It could occur when the FSF produces the matte at a lower rate and the PSC units demand faster matte loading. Therefore, the coordinator must instruct the PSC units to reduce their matte demand without adding unnecessary idle times to the schedule.

FSF lower-level inter-dependencies could exist in the schedule at various time instances. Like the FSF upper-level inter-dependencies, the framework has the potential to consider all or part of these inter-dependencies during a single iteration. Considering all the FSF lower-level inter-dependencies means simulating multiple batch problems in parallel. Consequently, two or more batch problems may start producing the same solution during two or more consecutive iterations. If this happens, the coordinator will not resolve the FSF lower-level inter-dependencies, and the framework may terminate without returning a feasible solution. In an ideal situation where the coordinator may find a viable solution, the solution quality will be lower because of the unnecessary idle times in the schedule. Therefore, this study addresses only the first FSF lower-level violations during an iteration to ensure that the framework produces a feasible solution with a high level of quality. In scenarios where the demand for matte is lower or fewer batches are produced, resolving the foremost lower limit violation may potentially resolve the subsequent FSF lower limit violations in the schedule.

To handle the inter-dependencies due to FSF lower-level violations, the coordinator uses Algorithm 4. Initially, it identifies the first instance where the FSF lower storage limit is violated. After that, it finds the batch problem that has requested for the matte loading at the conflicting time FSF_{TL} during the previous iteration. For this, the coordinator uses the batch problem start time ($BF_{start}^{n,b}$) and loading times (B_{load_1}, B_{load_1}) and assigns a value of 1 to the batch problem b on PSC unit n if the conflicting time instance FSF_{LT} belongs to it; otherwise, it assigns a value of 0.

The coordinator uses Eq. (51) to store the matte demand at the conflicting time instance FSF_{LT} . It helps the coordinator to react promptly if the FSF lower-level violation starts recurring at the time FSF_{LT} in the schedule. Lastly, the coordinator uses the PI controller to calculate the prices, as given in Eq. (52). If the prices are optimal, it will enable the conflicting batch problem ($FSF_{BP_{LL}}^{n,b} = 1$) to readjust its matte demand at the time instance FSF_{LT} .

$$FSF_{LLD}^{h,k} = \begin{cases} FSF_{LLD}^h & \forall h = FSF_{LT}, k++ \\ FSF_{LLD}^{h,k} & \text{otherwise} \end{cases} \quad (51)$$

$$LL_{price}^h = (P_{LL} \times FSF_{LLD}^h) + (I_{LL} \times \sum_{k=1}^{k-1} FSF_{LLD}^{h,k}) \quad \forall h = FSF_{TL} \quad (52)$$

The coordinator sends the prices LL_{price}^h , matte demand of other batch problems FSF_{MDO}^h , FSF_{LT} , and the FSF lower limit FSF_{LL}^h to the conflicting batch problem ($\forall FSF_{BP_{LL}}^{n,b} = 1$). After solving the scheduling problem, the conflicting batch problem reports its solution to the coordinator. The coordinator verifies the schedule and examines whether any FSF lower-level inter-dependencies exist in the schedule. If such inter-dependencies are present, the coordinator initiates a

Algorithm 4: FSF lower-level inter-dependencies

```

Begin
1: Initialization
    $stop_1 = false$ 
    $FSF_{LT} = 0$ 
    $FSF_{BP_{LL}}^{n,b} = 0$ 
    $FSF_{MD} = 0$ 
    $FSF_{MDO}^h = 0$ 
2: for  $h = 1$  to  $H$  do
   if ( $FSF_{LLV}^h \neq 0$  and  $stop_1 = false$ ) then
      $FSF_{LT} = h$ 
      $stop_1 = true$ 
   end
end
3: for  $b = 1$  to  $B$  do
   for  $n = 1$  to  $N$  do
     if ( $(BP_{start}^{n,b} + BP_{load_1}^{n,b}) \leq FSF_{LT} \leq (BP_{start}^{n,b} + BP_{load_1}^{n,b})$ ) then
        $FSF_{BP_{LL}}^{n,b} = 1$ 
     end
   end
end
4: for  $b = 1$  to  $B$  do
   for  $n = 1$  to  $N$  do
     if ( $h = FSF_{LT}$ ) then
        $FSF_{LLD}^h = PSC_{MD}^h - FSF_M^h + FSF_{LL}^h$ 
     else
        $FSF_{MD} = 0$ 
     end
   end
end
5: for  $h = 1$  to  $H$  do
   if ( $h = FSF_{LT}$ ) then
     for  $b = 1$  to  $B$  do
       for  $n = 1$  to  $N$  do
         if ( $FSF_{BP_{LL}}^{n,b} = 0$ ) then
           for  $t = 1$  to  $T$  do
             if ( $t = (FSF_{LT} - BP_{start}^{n,b})$ ) then
                $FSF_{MDO}^h = \sum_{n=1}^N \sum_{b=1}^B BP_{MD}^{n,b,t}$ 
             else
                $FSF_{MDO}^h = FSF_{MDO}^{h-1}$ 
             end
           end
         end
       end
     end
   end
end

```

new iteration of the same procedure. However, if no FSF lower-level inter-dependencies exist in the schedule, it searches for the other inter-dependencies. If it finds anything against the scheduling policy, it reacts accordingly and performs a new iteration. However, if the schedule is free of all FSF and PSC inter-dependencies, the coordinator returns the schedule, and the framework terminates.

6. Case studies

In order to showcase the effectiveness of the proposed scheduling framework, this study presents two case studies. The FSF matte grade

Table 1
Framework parameters.

Parameter	Description	Value
mg	matte grade	64 (percent)
$feed_{min}$	minimum feed rate	10 (percentage)
$feed_{max}$	maximum feed rate	100 (percentage)
FSF_{UL}^h	minimum FSF capacity limit	70 (ton)
FSF_{LL}^h	maximum FSF capacity limit	180 (ton)
$part_{TS}$	step size to solve FSF inter-dependencies	10 (percentage)
min_{TS}	minimum time instances in a given slot	1
MT_{ix}	matte transferred from FSF to PSC	20 (ton/min)
r_{Fe}	iron oxidation rate	0.240 (ton/min)
r_S	sulphur oxidation rate	0.0330 (ton/min)
max_{SB_1}	slag blow 1 maximum length	50 (min)
min_{SB_1}	slag blow 1 minimum length	5 (min)
max_{SB_2}	slag blow 2 maximum length	60 (min)
min_{SB_2}	slag blow 2 minimum length	5 (min)
fix_{SK_i}	slag removal time	1 (min)
fix_{ML_i}	loading time	1 (min)
Cu_{SB_1}	Copper loss rate during slag blow 1	0.0103 (ton/min)
Cu_{SB_2}	Copper loss rate during slag blow 2	0.052 (ton/min)
Cu_{SB_3}	Copper loss rate during slag blow 3	0.1 (ton/min)

and its maximum and minimum input feed rates remain constant. The composition of input concentrates plays a key role in the copper smelting process. While the proposed framework is capable of providing a viable solution for all types of input concentrate, a commonly utilized input concentrate is selected to provide a more practical solution. This study does not focus on a specific FSF; therefore, the FSF storage capacity limits are chosen arbitrarily.

Each PSC operation begins with two loading operations, followed by a slag blow operation and then a slag removal operation. Production of a batch of blister copper is assumed to involve four loading operations, three slag blow operations, three slag removal operations, and a single copper blow operation.; therefore, $I = 3$. The blister copper batch is ready when the iron and sulphur content of the matte drops below 2 percent (98 percent copper). In addition, each loading operation from the FSF to a PSC unit is limited to a maximum transfer of 20 tonnes of matte by the crane. The process parameters are given in Table 1, which remain unchanged. Both case studies are solved with GAMS using the CPLEX 12.7 solver, and the default optimality gap (10 percent) is used during the simulations (Bussieck and Meeraus, 2004; IBM, 2017).

CPLEX uses the branch and cut algorithm to address discrete-time optimization problems. With this algorithm, a well-chosen initial starting point can guide the solver toward the optimal branch, potentially minimizing computational costs. As the PSC model is a major contributor to the total computational expense, the framework initializes each batch problem with the solution computed by it in the previous iteration. Applying this warm start technique can potentially decrease the required load demand, especially when dealing with a higher number of PSC units or when producing more batches per unit.

In a schedule, multiple inter-dependencies can occur simultaneously. When a single type of inter-dependency is present, the coordinator can address it swimmingly. However, if different inter-dependencies exist in the schedule, the coordinator must determine the order in which to resolve them. Theoretically, the coordinator can select any order. However, the order of inter-dependencies handling affects the quality of the solution. For example, if logistical and the FSF lower-level inter-dependencies appear in the schedule simultaneously, solving the logistical inter-dependencies first can potentially resolve the FSF lower limit inter-dependencies. Likewise, if both logistical inter-dependencies and flow inter-dependencies are found simultaneously, logistical inter-dependencies should be solved before the flow inter-dependencies to provide a feasible solution with minimal computational resource requirement. The order in which the coordinator handles the inter-dependencies is shown in Fig. 6.

This study uses PI controllers to compute prices. The performance of a PI controller typically relies on the values of its proportional and

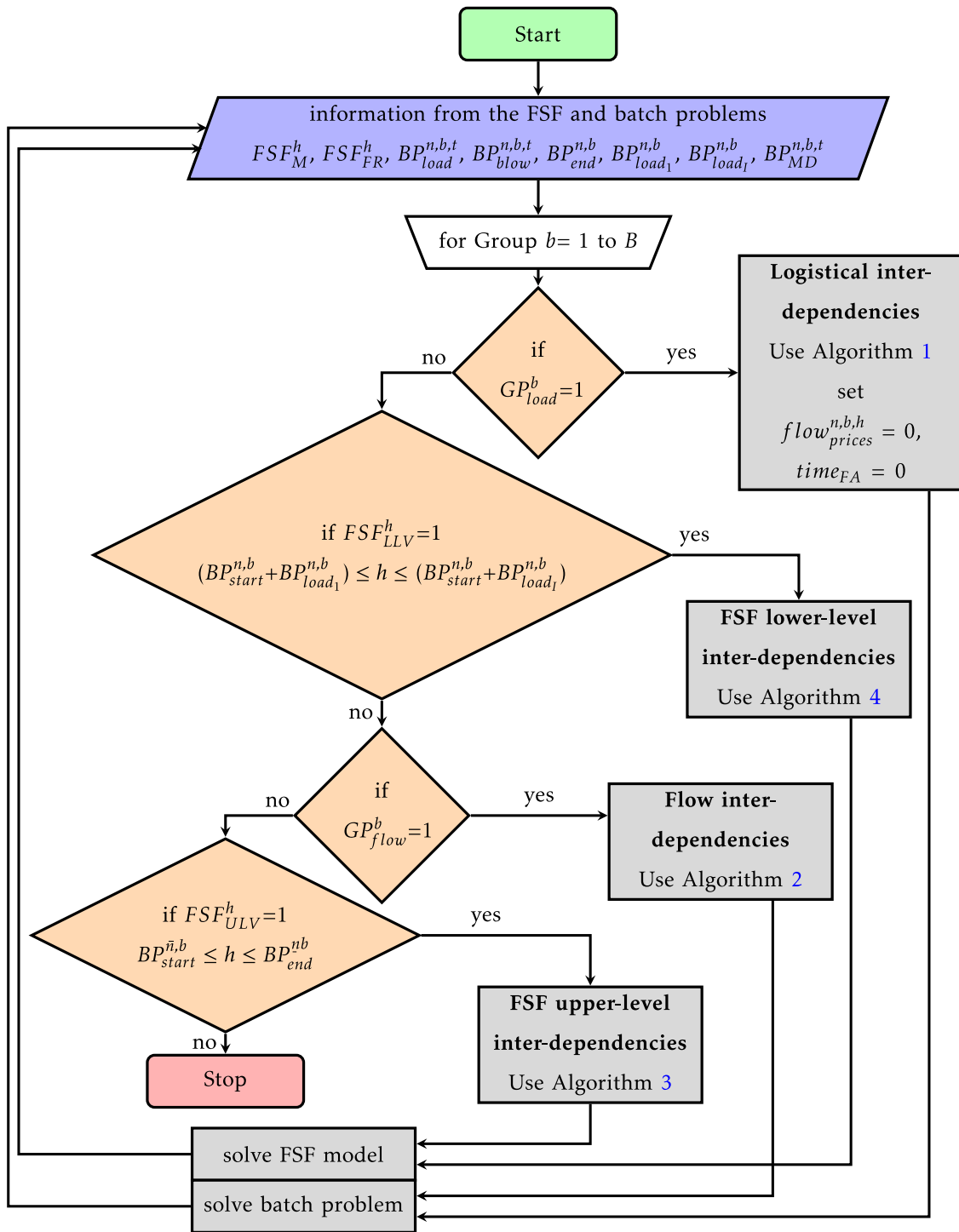


Fig. 6. Order of handling inter-dependencies.

integral gains, which affect the coordinator’s performance (Ahmed and Vilkkö, 2023). Usually, these gain values are calculated from the step response of the process model. While computing the step response is relatively straightforward for a process with simple dynamics or a simple input–output model, finding the step response for a process with complex process dynamics and structure is always challenging.

An alternative approach to computing PI controller gain values is the trial and error method. This method involves selecting arbitrary gain values at the outset and then fine-tuning them through trial and error, which can be time-consuming. However, for complex

processes such as the copper smelting process, where finding the step response can be challenging, the trial-and-error method remains the most suitable option.

In the trial-and-error method, the selection of high gain values means that the controller will react aggressively if there are inter-dependencies in the schedule. This aggressiveness of the coordinator seems pleasant since it will reduce the number of iterations necessary to find a feasible solution. However, the choice of excessively high values reduces the quality of the solution. On the contrary, opting for extremely low values of the controller gain leads to an increase

Table 2
PI controller gain values.

Set	FSF upper-level inter-dependencies		PSC and FSF lower-level inter-dependencies					
	P_{UL}	I_{UL}	P_{load}	I_{load}	P_{flow}	I_{flow}	P_{LL}	I_{LL}
1	100	10	10	1	10	1	10	1
2	1000	100	100	10	10	1	10	1
3	10000	1000	1000	100	10	1	10	1

Table 3
Framework parameters-Case study 1.

Parameter	Description	Value
$FSF_M^{h=0}$	FSF initial mass	150 (ton)
$start_{PSC}^n$	PSC starting time	{0,0,0} (min)
H	scheduling horizon	1200 (min)
T	batch scheduling horizon	220 (min)
K	maximum number of iterations	1000

in the number of iterations, but it also improves the quality of the solution. The FSF and PSC models treat the calculated prices as a penalty term in the objective function. As a result, extremely high gain values can adversely affect the quality solution and computational costs. Hence, suitable values must be selected considering the available computational power and size of the scheduling problem.

In order to demonstrate the effect of gain values on framework performance, both case studies are resolved for three different controller gain values, as outlined in Table 2. The motivation is to demonstrate that the proposed scheduling framework has the potential to provide a feasible schedule for any gain value. However, the quality of the solution varies according to the gain values used. As the FSF does not contribute to the number of idle times in the schedule, the controller gain values for the FSF high-level inter-dependencies are selected higher to reduce computational demands.

6.1. Case study 1

The first case study involves one FSF and three PSC units, with each PSC unit producing three batches, resulting in a total of nine PSC batches. Table 3 provides the FSF initial inventory level and the availability of the PSC units. As both PSC units are available concurrently, the coordinator assigns the highest priority to PSC unit 1 and the lowest priority to PSC unit 2.

Case study 1 is simulated for the PI controller gain values indicated in Table 2, and their corresponding FSF matte, FSF feed rate, and PSC schedule are presented in Figs. 7–9. As sufficient matte is available at the beginning of the simulation and the matte demand is not high, matte in the FSF remains above the FSF lower level, as shown in Figs. 7(a)–9(a). Thus, no FSF lower-level inter-dependencies exist in this case study. In each iteration, the framework collects information from batch problems and sequentially verifies the presence of FSF or PSC inter-dependencies in ascending order of the group number. In each group, the coordinator solves the logistical inter-dependencies first, followed by flow inter-dependencies. It addresses the FSF upper-level inter-dependencies as they appear in the schedule.

In this case study, the framework divides all batch problems into two groups. Group 1 consists of the first batch problem on each PSC unit, Group 2 consists of the second batch problem, and Group 3 has the third batch problem on each PSC unit. During each iteration, the coordinator starts with the batch problems in Group 1 and checks for PSC inter-dependencies. If inter-dependencies exist in Group 1, they are resolved first. After settling the inter-dependencies in Group 1, the coordinator checks for the inter-dependencies between Group 1 and Group 2. If inter-group inter-dependencies exist, the coordinator resolves them by rescheduling the batch problems in Group 2. Otherwise, the coordinator checks for the FSF upper-level inter-dependencies and address them accordingly. Next, the coordinator examines Group 2

Table 4
Case study 1.

Set	Iterations	Computational time (min)	Batch time (min)	Copper loss (ton)
1	538	42.8	452	8.60
2	288	30.2	518	8.66
3	204	25.9	622	8.92

and then Group 3 and performs the same actions if inter-dependencies exist between the batch problems. Once all the inter-dependencies are resolved, the framework terminates and returns the final schedule.

Figs. 7(b)–9(b) show that the FSF is utilizing its maximum feed rate; thus, it is operating at the optimal operating points. However, the framework adjust the FSF feed rate to address the FSF upper-level inter-dependencies. The PSC units are producing blister copper batches in a synchronized manner, as shown in Figs. 7(c)–9(c). These figures show that the framework resolves all the logistical and flow inter-dependencies and provides feasible schedules.

The computational time for Case Study 1 is given in Table 4. The PI controller gain values for Set 1 are lower than Sets 2 and 3; therefore, the framework required more iterations to solve inter-dependencies. Thus, the computational time is high. For Set 1, the prices do not change exponentially during iterations if identical inter-dependencies appear in the schedule repeatedly. Therefore, the price calculation strategy moves steadily, and the framework is able to compute the optimal price. As a result, the quality of the solution is superior to the Set 2 and 3 gain values. For Sets 2 and 3, the number of iterations is lower than in Set 1, so the computational time is lower. This is because the coordinator can quickly determine the optimal prices, thereby minimizing the computational costs of the framework. Nonetheless, the objective functions of the FSF and PSC units are penalized by a higher penalty term. Such actions add unnecessary idle time to the schedule, making the quality of the solutions inferior. Even with varying PI controller gain values, the framework can effectively resolve all scheduling inter-dependencies and produce a feasible solution without any hindrances.

The amount of copper losses during the framework simulation depends on the quality of the solution, which is determined by the batch time of the schedule. A shorter batch time indicates that the framework has optimally selected the duration of slag blows, resulting in lower copper losses. For the Set 1 gain values, the slag-blowing durations are computed optimally, leading to lower copper losses compared to Sets 2 and 3, where the PSC model objective function is penalized with a higher penalty term. As a result, the slag-blowing durations are sub-optimal, leading to higher copper losses. Principally, process personnel seeks solutions that have minimum batch time and copper losses. Therefore, solutions using Set 1 gain values are preferable for process operation.

6.2. Case study 2

In Case Study 2, there is one FSF and three PSC units, with each PSC unit producing five batches, resulting in a total of 15 PSC batches. This case study considers a lower value for the FSF initial inventory as compared to Case Study 1, as given in Table 5. The PSC units are available simultaneously; hence, PSC unit 1 has the highest priority,

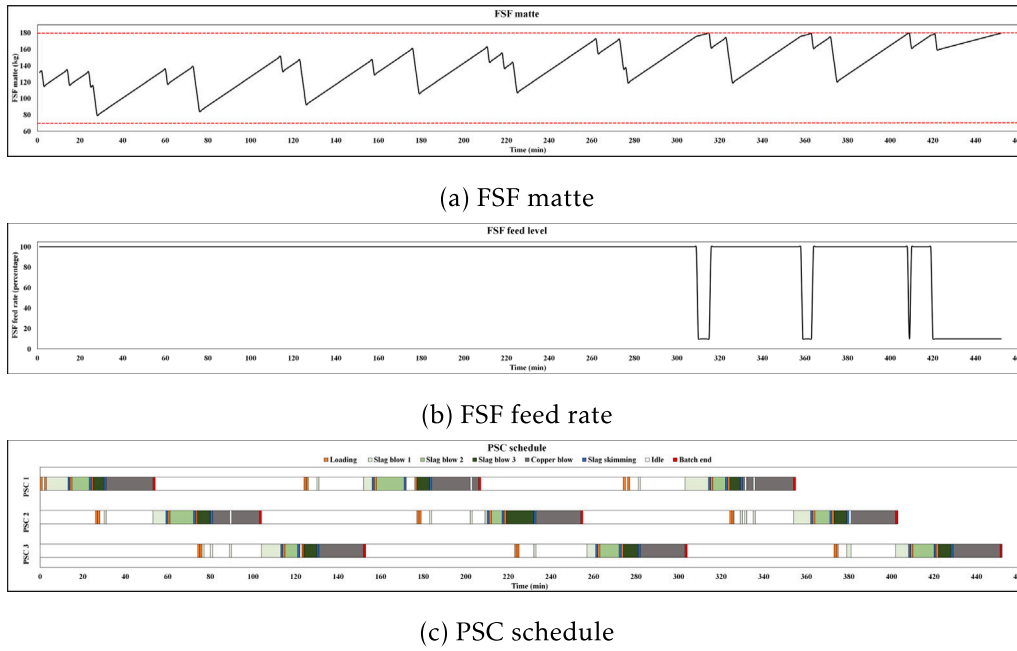


Fig. 7. Case study 1 - Set 1.

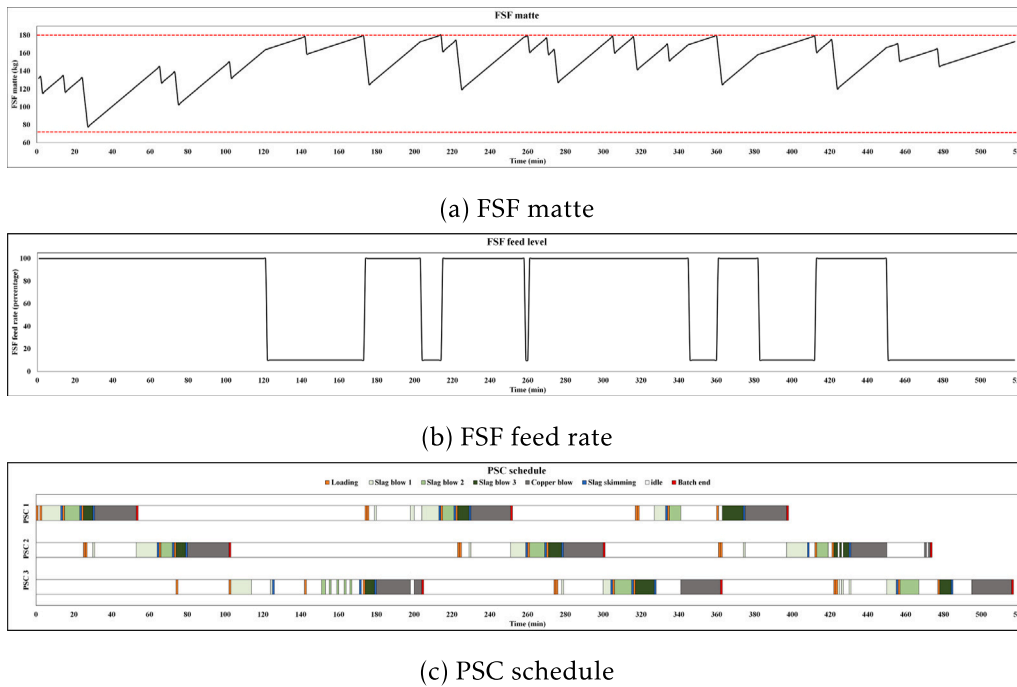


Fig. 8. Case study 1 - Set 2.

and PSC unit 3 has the lowest priority. As more batches are produced in this case study, this case study will have a higher number of iterations, higher computational demand, and longer batch time.

Case study 2 is simulated for the gain values as given in Table 2, and the FSF matte, FSF feed rate, and the corresponding PSC schedule are shown in Figs. 10–12. As insufficient FSF initial inventory is available at the beginning of the process, FSF lower-level inter-dependencies exist at time $t = 30$ min, as shown in Figs. 10(a)–12(a). During each iteration, the coordinator gathers all the relevant information from all the batch problems and checks for the inter-dependencies in

Table 5
Framework parameters-Case study 2.

Parameter	Description	Value
$FSF_M^{h=0}$	FSF initial mass	130 (ton)
$start_{PSC}^c$	PSC starting time	{0,0,0} (min)
H	scheduling horizon	1200 (min)
T	batch scheduling horizon	220 (min)
K	maximum number of iterations	1000

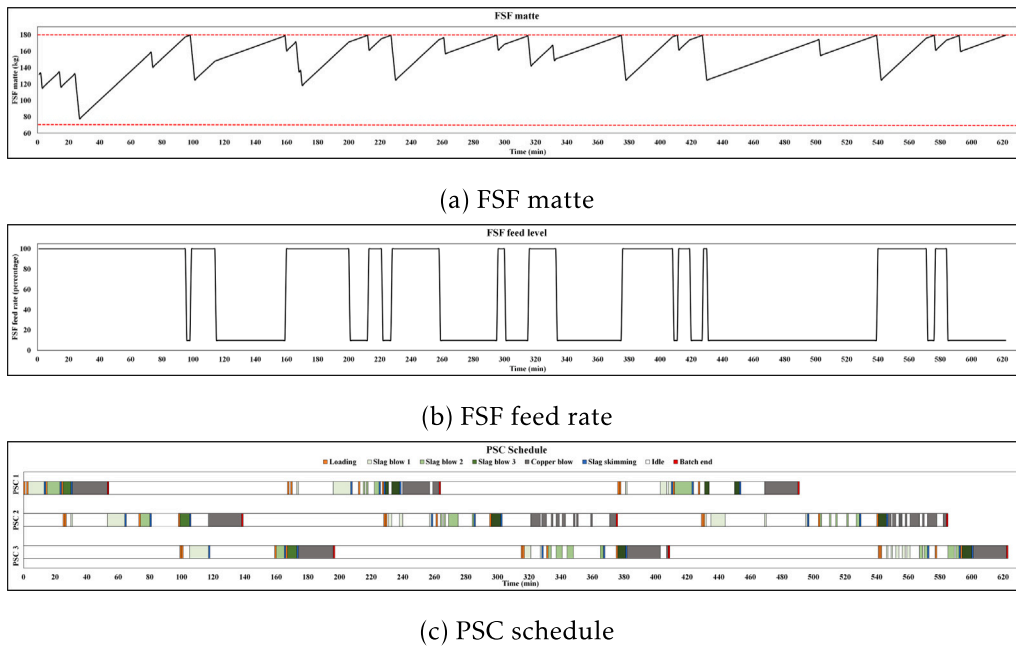


Fig. 9. Case study 1 - Set 3.

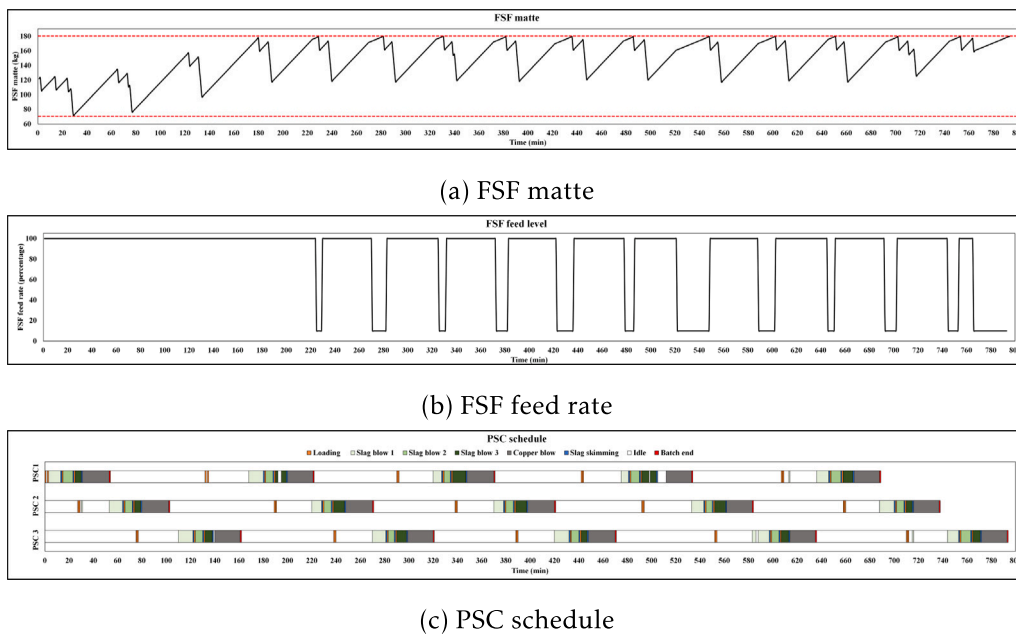


Fig. 10. Case study 2 - Set 1.

ascending order of the group number. For each group, the coordinator first resolves logistical inter-dependencies, followed by the FSF lower-level inter-dependencies, and then the PSC flow inter-dependencies. As in Case Study 1, FSF upper-level inter-dependencies are handled as they appear in the group by adjusting the FSF feed rate as shown in Figs. 10(b)–12(b). The framework effectively addresses all logistical and flow inter-dependencies, ensuring synchronized and feasible schedules, as depicted in the accompanying Figs. 10(c)–12(c).

The coordinator divides all the batch problems into five different 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 simulation, the coordinator reviews Group 1 first and resolves the FSF and PSC inter-dependencies, if found in the schedule. If Group 1 is free of all inter-dependencies, it examines inter-group

inter-dependencies between Group 1 and Group 2. If inter-group inter-dependencies exist, the coordinator resolves them in the same manner as discussed in Case Study 1; otherwise, it begins to examine Group 2 and performs a new iteration. This process continues until all the inter-dependencies are addressed successfully. When this happens, the framework stops and returns the final solution (see Figs. 10(c)–12(c)).

The computational time and copper losses for Case Study 2 is given in Table 6. For Set 1 gain values, the PSC schedule has the minimum batch time and copper losses; therefore, the solution is classified as optimal. For low gain values, the framework determines the optimal duration of the slag blow that leads to minimal copper losses. Concerning computational costs, this solution is considered inefficient since it demands more computational time to generate a feasible schedule. On the other hand, for Set 3 gain values, the number of iterations

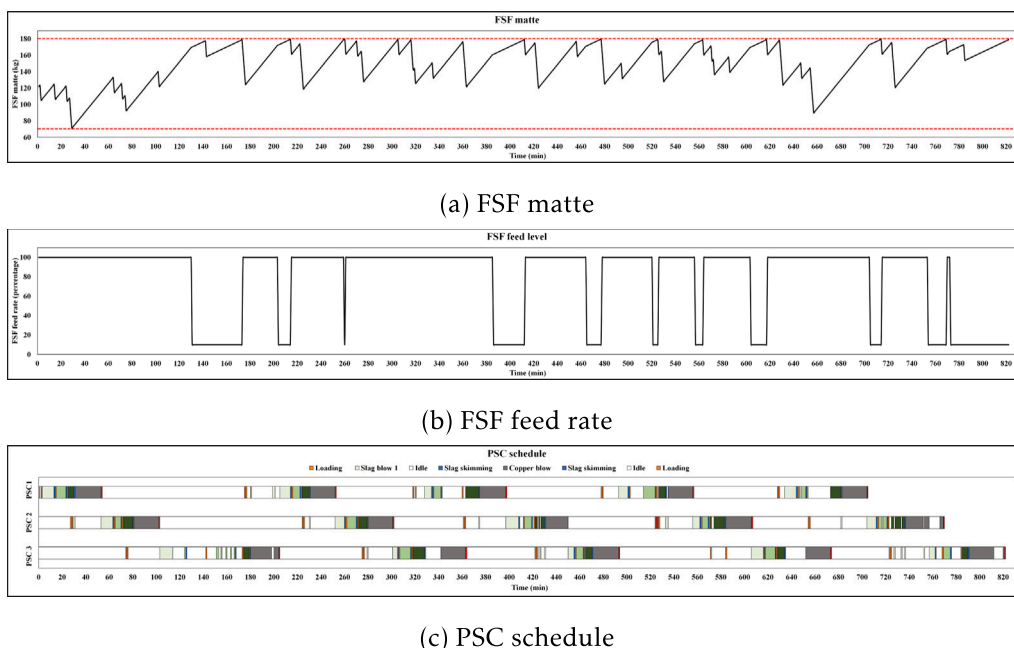


Fig. 11. Case study 2 - Set 2.

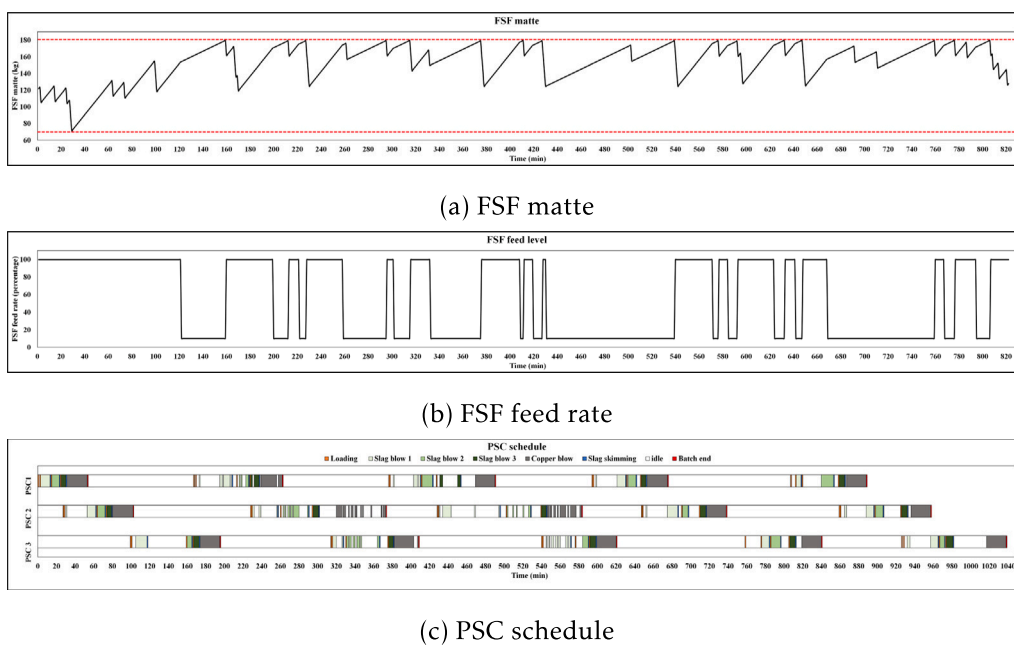


Fig. 12. Case study 2 - Set 3.

required by the framework is lower, so the computational demand is lower. Therefore, this solution is classified as optimal in terms of computational costs.

For Set 3 gain values, the coordinator penalizes the FSF and PSC objective functions with a higher penalty term that adds unnecessary idle times to the schedule, which increases the batch time. Consequently, in terms of the solution quality, such a solution is termed as poor. For Set 2 gain values, the computational time is lower than the batch time for the Set 3 gain values, while the batch time is higher than for Set 1; thus, this solution is classified as sub-optimal. Because process personnel generally prefer shorter batch time and minimal copper losses, process personnel will use the solution for the Set 1 gain values.

Table 6

Case study 2.

Set	Iterations	Computational time (min)	Batch time (min)	Copper loss (ton)
1	522	125.2	793	13.29
2	415	55.9	822	14.75
3	236	40.9	1041	14.96

The gain values presented in Table 2 are selected arbitrarily to illustrate the impact of different gain values on the performance of the framework and the advantages and disadvantages associated with

their selection. However, the framework can produce a viable solution for any given gain values. The choice of gain values depends on the available computational power, the size of the scheduling problem, the optimality gap, and the scheduling application in hand (Ahmed and Vilkkö, 2023).

Besides the gain values, the framework performance depends on the batch problem horizon T . If a high value of T is selected, the simulation time of the PSC model increases; consequently, the computational time of the framework increases. On the contrary, if a too-small value is selected, the PSC model might not be able to find a feasible solution, especially when a batch problem is the source of both FSF lower-level inter-dependencies and logistical inter-dependencies. Therefore, a suitable value for the T must be selected considering the duration of the PSC internal actions, the number of PSC units employed, and the PSC maintenance break if considered during the framework formulation.

PI controller tuning parameters influences the framework performance. Consequently, it is crucial to establish more stringent guidelines for selecting these parameters, considering factors such as the available FSF capacity levels, computational power, and maximum batch time of the schedule. Nonetheless, the findings of this study highlight that the suggested scheduling framework generates a viable schedule for a wide range of tuning parameters, indicating the potential for implementing price-based coordination in various other large-scale scheduling applications.

7. Conclusion

This study proposes a hierarchical framework for scheduling the copper smelting process based on MILP. This framework provides an optimal schedule for the PSC units considering the continuous operation of the FSF, FSF capacity constraints, and batch synchronization. The proposed hierarchical framework takes advantage of the market theory and uses prices to resolve the FSF and PSC inter-dependencies. The price-based coordination has the potential to provide optimal coordination between independent FSF and PSC units, and it can enable the process personnel to take decisions without a deep understanding of the process.

In this work, PI controllers are employed to determine the optimal prices, and it is demonstrated that the choice of controller gain values can significantly affect the quality of the solution. While the framework can provide a feasible solution for any gain value, using high-gain values often leads to poor-quality solutions. On the other hand, selecting lower-gain values requires more computational time, but it results in better-quality solutions. If the proposed framework is used as an offline tool, where computational time is less of a concern, process personnel would be inclined to use lower-gain values to achieve minimum copper losses. Additionally, this framework can serve as a valuable tool for process personnel to anticipate the consequences of their decisions and enhance their understanding of the process.

The study presented only deterministic case studies to demonstrate the ability of the proposed price-based coordination mechanism to resolve process operations and operate the copper smelting process at or near optimal operation points. Future work will involve developing techniques to calculate optimal controller gain values and identifying other computationally efficient price-updating mechanisms for the coordinator, which can improve even better solution quality.

CRedit authorship contribution statement

Hussain Ahmed: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Visualization. **Matti Vilkkö:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision, Visualization, 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 a relationship with Tampere University that includes: employment.

Data availability

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

References

- Abreu, Levi R., Prata, Bruno A., Framinan, Jose M., Nagano, Marcelo S., 2022. New efficient heuristics for scheduling open shops with makespan minimization. *Comput. Oper. Res.* (ISSN: 0305-0548) 142, 105744. <http://dx.doi.org/10.1016/j.cor.2022.105744>, URL <https://www.sciencedirect.com/science/article/pii/S0305054822000429>.
- Adams, Thomas A., Seider, Warren D., 2009. Design heuristics for semicontinuous separation processes with chemical reactions. *Chem. Eng. Res. Des.* (ISSN: 0263-8762) 87 (3), 263–270. <http://dx.doi.org/10.1016/j.cherd.2008.09.008>, URL <https://www.sciencedirect.com/science/article/pii/S0263876208002931>.
- Ahmed, Hussain, Ricardez-Sandoval, Luis, Vilkkö, 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>, URL <https://www.mdpi.com/2227-9717/9/11/2004>.
- Ahmed, Hussain, Ricardez-Sandoval, Luis A., Vilkkö, 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>, URL <https://www.sciencedirect.com/science/article/pii/S0098135422002022>.
- Ahmed, Hussain, Vilkkö, Matti, 2023. Coordination strategy based on hard-heuristics and price-updating scheme for copper smelting process. *Comput. Chem. Eng.* (ISSN: 0098-1354) 173, 108198. <http://dx.doi.org/10.1016/j.compchemeng.2023.108198>, URL <https://www.sciencedirect.com/science/article/pii/S0098135423000674>.
- Björklund, Peter, Korpi, Mikko, Grimsey, David, Marjakoski, Miikka, 2021. Continuous improvement of process advisor optimizing furnace model. In: Anderson, Corby, Goodall, Graeme, Gostu, Sumedh, Gregurek, Dean, Lundström, Mari, Meskers, Christina, Nicol, Stuart, Peuraniemi, Esa, Tesfaye, Fiseha, Tripathy, Prabhath K., Wang, Shijie, Zhang, Yuanbo (Eds.), *Ni-Co 2021: The 5th International Symposium on Nickel and Cobalt*. Springer International Publishing, Cham, ISBN: 978-3-030-65647-8, pp. 259–270.
- Busisiek, 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>, URL <http://www.sciencedirect.com/science/article/pii/S1474667016354581>. 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>, URL <http://www.sciencedirect.com/science/article/pii/S0959152406000540>. Selected papers presented at the 2005 IFAC World Congress.
- Cheng, Ruoyu, Fraser Forbes, J., San Yip, W., 2004. 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](http://dx.doi.org/10.1016/S1474-6670(17)31931-6), URL <http://www.sciencedirect.com/science/article/pii/S1474667017319316>. 7th IFAC Symposium on Dynamics and Control of Process Systems 2004 (DYCOPS -7), Cambridge, USA, 5–7 July, 2004.
- Davenport, W.G.I., King, M., Schlesinger, M.E., Biswas, A.K., 2002. *Extractive Metallurgy of Copper*. Elsevier Science, ISBN: 9780080531526, URL <https://books.google.fi/books?id=VdILLpCu9ooC>.
- 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](http://dx.doi.org/10.1016/S1570-7946(06)80209-9), URL <http://www.sciencedirect.com/science/article/pii/S1570794606802099>.

- Harjunkoski, Iiro, Fahl, Marco, Borchers, Hans Werner, 2008. Scheduling and optimization of a copper production process. In: *Logistic Optimization of Chemical Production Processes*. John Wiley & Sons, Ltd, ISBN: 9783527622771, pp. 93–109. <http://dx.doi.org/10.1002/9783527622771.ch5>, URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/9783527622771.ch5>.
- 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](http://dx.doi.org/10.1016/S0098-1354(01)00729-3), URL <https://www.sciencedirect.com/science/article/pii/S0098135401007293>.
- IBM, 2017. V12.7 IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. International Business Machines Corporation, URL https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/uscplex.pdf.
- 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.
- Li, Liquan, Pan, De'an, Li, Bin, Wu, Yufeng, Wang, Huaidong, Gu, Yifan, Zuo, Tiejong, 2017. Patterns and challenges in the copper industry in China. *Resour. Conserv. Recy.* (ISSN: 0921-3449) 127, 1–7. <http://dx.doi.org/10.1016/j.resconrec.2017.07.046>, URL <https://www.sciencedirect.com/science/article/pii/S0921344917302392>.
- Panek, Sebastian, Engell, Sebastian, Lessner, Cathrin, 2005. Scheduling of a pipeless multi-product batch plant using mixed-integer programming combined with heuristics. In: Puigjaner, Luis, Espuña, Antonio (Eds.), *European Symposium on Computer-Aided Process Engineering-15*, 38th European Symposium of the Working Party on Computer Aided Process Engineering. In: *Computer Aided Chemical Engineering*, vol. 20, Elsevier, (ISSN: 1570-7946) pp. 1033–1038. [http://dx.doi.org/10.1016/S1570-7946\(05\)80014-8](http://dx.doi.org/10.1016/S1570-7946(05)80014-8), URL <https://www.sciencedirect.com/science/article/pii/S1570794605800148>.
- 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>, URL <http://www.sciencedirect.com/science/article/pii/S1877705814003890>. 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, *The Hermann Schwarze Symposium on Copper Pyrometallurgy*.
- Rojas, Lorena, Zúñiga, Jorge, Parada, Victor, 2006. CODELCO, Chile programs its copper-smelting operations. *Interfaces* 36, <http://dx.doi.org/10.1287/inte.1060.0207>.
- 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>, URL <https://www.sciencedirect.com/science/article/pii/S036083522100379X>.
- 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>, URL <https://www.sciencedirect.com/science/article/pii/S1474667015382604>. 10th IFAC International Symposium on Dynamics and Control of Process Systems.
- Sarabia, Rubén Martiand 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>, URL <http://www.sciencedirect.com/science/article/pii/S0098135413001877>. Selected papers from ESCAPE-22 (European Symposium on Computer Aided Process Engineering - 22), 17–20 June 2012, London, UK.
- 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>, URL <https://www.sciencedirect.com/science/article/pii/S0921344918300041>.
- Sobeyko, Oleh, Mönch, Lars, 2015. Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Comput. Oper. Res.* 68, <http://dx.doi.org/10.1016/j.cor.2015.11.004>.
- Suominen, Olli, M, 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>, URL <http://www.sciencedirect.com/science/article/pii/B9780444634283502125>.
- 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>.
- Zhang, Lei, Tian, Fengchun, Kadri, Chaibou, Pei, Guangshu, Li, Hongjuan, Pan, Lina, 2011. Gases concentration estimation using heuristics and bio-inspired optimization models for experimental chemical electronic nose. *Sensors Actuators B* (ISSN: 0925-4005) 160 (1), 760–770. <http://dx.doi.org/10.1016/j.snb.2011.08.060>, URL <https://www.sciencedirect.com/science/article/pii/S0925400511007830>.