Continuous-time formulation and differential evolution algorithm for an integrated batching and scheduling problem in aluminium industry

This paper has been accepted for publication in International Journal of Production Research.
Qingxin Guo, Lixin Tang, Jiyin Liu & Shengnan Zhao (2020) Continuous-time formulation and differential evolution algorithm for an integrated batching and scheduling problem in aluminium industry, *International Journal of Production Research*, DOI: 10.1080/00207543.2020.1747656

Continuous-time Formulation and Differential Evolution Algorithm for an Integrated Batching and Scheduling Problem in Aluminium Industry

Qingxin Guo[a,b], Lixin Tang[a,c,*], Jiyin Liu[d] and Shengnan Zhao[c,e]

[a] *Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang, China*

[b] *Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Northeastern University, Shenyang, China*

[c] *Institute of Industrial and Systems Engineering, Northeastern University, Shenyang, China*

[d] *School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, UK*

[e] *Liaoning Key Laboratory of Manufacturing System and Logistics, Northeastern University, Shenyang, China*

Abstract

This paper investigates an integrated batching and scheduling problem of electrolysis and caster in aluminium industry. The problem is to determine the assignment and scheduling of orders considering sequence-dependent setup times caused by technological and operational constraints of electrolysis cells, and determine the batching and scheduling of orders in the following casters. A novel unit-specific event-based continuous-time mixed integer linear programming model (MILP) is proposed to describe the problem. In this model, the event point is stage specific, and lower bounds are specified to tighten the model. A hybrid pointer-based differential evolution algorithm with new individual representation scheme is designed to solve the problem of industrial scale. An improved hybrid pointer-based mutation operator and a new point-cross crossover operator are proposed to enhance the performance of the algorithm. Computational experiments show that the proposed algorithm is more efficient when compared with CPLEX for medium and large size instances. Comparisons with the lower bound demonstrate that the algorithm is effective.

Keywords: integrated batching and scheduling; aluminium industry; unit-specific event-based; continuous-time model; differential evolution.

* Corresponding author

1. Introduction

Aluminium is the most abundant metal in the earth's crust. Due to favourable properties, e.g., lightweight and corrosion resistant, aluminium is widely used in aerospace engineering and chemical engineering. As illustrated in Figure 1, the production process of aluminium roughly consists of two stages: mining raw material (alumina) and aluminium fabrication process.

Insert Figure 1 about here.

Electrolysis and casting are important operations in aluminium fabrication process. In electrolysis production, pure aluminium is extracted from alumina which is called Hall-Hėroult process. Electrolysis cells work at low voltage, like 5-6 volts, but at huge currents, 150,000 amps or more. The heating effect of these large currents keeps the cells at a temperature of about 950-980°C (Moors. 2006). Cells are operated 24 hours a day so that the molten aluminium will not solidify. Both electrolysis and casting are energy intensive and highly polluting processes since large amount of electricity, natural gas or other fuel are consumed to achieve the desired production conditions. After decades of research and development of technology, it is difficult to save energy by improving the production process or equipment. However, from another point of view, a good scheduling of electrolysis and casting is an effective approach to achieve reduction of energy consumption by reducing setup times and shorten makespan. In this paper, we investigate the integrated batching and scheduling problem for aluminium electrolysis and casting processes. Considering the practical scale, effective formulation and new algorithms are needed to obtain near optimal solutions.

There is less research on production scheduling of aluminium compared with iron and steel industry. Bowers et al. (1995) proposed a two-phase model for aluminium ingot casting to reduce misapplication/import and maintain low inventory levels. Gravel et al. (2000) presented a genetic algorithm for scheduling jobs in an aluminium foundry with sequence-dependent setup times. Gravel et al. (2002) proposed an ant colony optimization algorithm for an aluminium casting scheduling problem. They proposed a representation of a continuous horizon which considered several objectives. Schwindt and Trautmann (2003) formulated rolling ingots scheduling using prescribed time lags between operations, different kinds of resources, and sequence-dependent changeovers. A branch-

and-bound procedure was implemented to solve it. Prasad et al. (2006) presented an MILP model for the optimization of aluminium smelter cast house operations. An iterative decomposition scheme was developed to solve medium-sized to large-sized problems. Ladurantaye et al. (2007) studied the integrated homogenizing furnaces and the mill scheduling problem to minimize the idle time and penalties for soft constraint violations. Duman et al. (2008) considered the casting lines scheduling of an aluminium casting and processing plant. Their objective was to minimize setup time for a given time period while balancing workload to accommodate potential new orders. Steinrücke (2011) studied a global aluminium supply chain network, relax-and-fix heuristics were proposed. Steinrücke (2015) studied an integrated aluminium production, distribution and scheduling problem, and proposed a continuous-time MILP model and decomposition heuristic. To the best of our knowledge, no attention has been paid on integrated batching and scheduling of electrolysis and casting.

Considering time representation, the mathematical models for the scheduling problems could be classified into: precedence-based, discrete-time, and continuous-time. Floudas and Lin (2004, 2005), Mendez et al. (2006), and Harjunkoski et al. (2014) have presented extensive reviews for the formulations. Lee and Maravelias (2017, 2017) presented discrete-time MILP formulations for short-term scheduling and simultaneous batching and scheduling in multipurpose environments, respectively. Seid and Majozi (2014) proposed a formulation to optimize the use of both water and energy in multipurpose batch plants. The approaches to formulate continuous-time models include: time slots based models, global event based models and unit-specific event-based models. Comparing with other models, unit-specific event-based models (Ierapetritou and Floudas, 1998a, 1998b) requires fewer event points, leading to a smaller model size, and hence, a better computational performance. Shaik et al. (2006), Janak and Floudas (2008), Shaik and Floudas (2007, 2008, 2009), Li and Floudas (2010) proposed improved version of this formulation for short-term scheduling problems. Vooradi and Shaik (2012) proposed improvements in allocation, duration and sequencing constraints, and investigated the effect of big-M terms of unit-specific event-based models. Shaik and Vooradi (2017) proposed a reformulation of this model to reduce events, constraints and variables. There are also successful applications of this formulation on iron and steel, and crude oil scheduling problems (Li et al. 2012, 2012, and 2016).

Most unit-specific event-based models are represented as MILP or MINLP. For the small and medium scale instances, the models could be solved by standard optimization solvers such as CPLEX and ANTIGONE (Misener and Floudas, 2014). Merchan and Maravelias (2016) proposed preprocessing and tightening methods for time-indexed, discrete and continuous, MIP scheduling models which could reduce computational time or improve optimality gap. However, it may be difficult to obtain optimal solutions for some large-scale instances within short solution time. In this case, evolutionary computation algorithms could be used to obtain near-optimal solutions. Differential evolution (DE) is a population-based stochastic evolutionary algorithm proposed by Storn and Price (1995, 1997), which has exhibit remarkable performance for many optimization problems. DE variants have been proposed for both benchmark and practical optimization problems (Tang et al., 2015, Tang et al., 2014). The recent developments of DE were reviewed by Das and Suganthan (2011) and Das et al. (2016).

In this paper, we propose a continuous-time formulation and an improved DE algorithm for the aluminium integrated batching and scheduling problem. The remainder of this paper is organized as follows: Section 2 describes the problem in more details. The proposed model is formulated in Section 3. The improved DE algorithm is proposed in Section 4. Section 5 reports the experimental results. Conclusions are stated in Section 6.

2. Problem description

In a typical aluminium fabrication plant of China, there are a large number of parallel electrolysis cells in the electrolysis workshop, and parallel casters in the smelter. Pure aluminium is extracted from alumina by electrolysis cells and then casted into ingots by casters. The aluminium production is order-oriented, which means that customer orders are first contracted between the customers and the aluminium plant, and then transformed to production orders in the form of alloy composition, size and due date with the consideration of production capacity and operation constraints. Finally, the production orders are scheduled on the electrolysis cells and casters.

Insert Figure 2 about here.

Figure 2 shows a schematic of the proposed batching and scheduling problem. According to technological requirements, each order has order-specific processing time and needs to be assigned to cells according to assignment regulations. The orders at the

same cell should be sequenced considering the possible setup times, which are mainly caused by the variation of the alloy composition of orders. The setup time is sequence-dependent in the light of setup forms such as adjustment of facilities and cleaning of cells, which are both costly and time consuming.

After the electrolysis production, orders are processed at the furnace and caster stage. The basic unit of furnace and caster production is a cast, which is defined as a "batch" in the scheduling. All the orders for the same batch have to arrive at the furnace in order to be pre-processed together and then casted continuously. Before finishing the current batch, the furnace cannot start pre-processing the orders for the next batch. The pre-processing time can be considered as proportional to the casting time and so both are counted into the processing time of this stage. The batching decisions for aluminium caster involve grouping different cells to form a set of casts. The total molten aluminium for each batch must be within a certain range of capacity. In real production, each caster batch is made of up to 3 electrolysis cells, since each electrolysis cell has a capacity of 1.5-2 tons, and a caster has about 5 tons. The casters are expected to run in high or full load, in order to provide high productivity and save energy. Based on this reality, we assume that the batch capacity is fixed to three in this paper to simplify the problem. The production time of a cast is equal to the summation of pre-treatment and production time of the orders that are assigned to this cast. For the furnace and caster stage, the production structure of orders is roughly the same and the production is continuous. Therefore, we consider the furnace and caster as one stage which could simplify the problem.

As shown in Figure 2, the process of this problem includes two stages: electrolysis and casting. A number of orders are needed to be assigned to electrolysis cells and scheduled considering production constraints and sequence-dependent setups. Then the electrolyzed pure molten aluminium will be combined into different batches (casts), which are assigned and scheduled at casters.

The characteristics of this problem are summarized as follows: 1) the scheduling of production orders at the first stage must consider the requirements from the batching and scheduling decisions at the second stage; 2) the batching and scheduling at the second stage should consider the batch capacity and the release time which depend on the scheduling at the first stage. The interaction of the two stages makes it difficult to obtain the optimal batching and scheduling plan for the whole production. This problem could be described as a variant of the two-stage hybrid flow shop scheduling problem but can

be distinguished by the setup times and the batching decision. For more details of classical two-stage hybrid flow shop scheduling problem, readers are referred to Gupta et al. (1997) Lee and Kim (2004), Gong et al. (2010), Yu et al. (2017), and Hwang and Lin (2018). Due to these new features, the existing formulations and algorithms for the typical hybrid flow shop scheduling problem cannot be adopted well to this problem.

3. Mathematical Model

In this section, we formulate the considered batching and scheduling problem as a novel unit-specific event-based model. The main feature of this problem is that the production objects are orders at the first stage and batches at the second stage. To deal with this feature, different event points are proposed for the two stages respectively. Other characteristics of this model include: practical production features (setup times, batching) are considered, lower bounds are specified to tighten the model, and a strategy is presented to estimate the minimum number of event points.

The proposed formulation requires the indices, sets, parameters, and variables given as follows.

Indices:

i	orders
j	units, a unit here is equivalent to a machine in the hybrid flow shop.
n	event points

Sets:

I	set of orders
$J1$	set of units at the first stage (electrolysis cells)
$J2$	set of units at the second stage (casters)
$N1$	set of event points at the first stage
$N2$	set of event points at the second stage

Parameters:

$v(i)$	the weight of order i
B^{num}	the number of orders in a batch
$B^{\max}$	the maximum weight of a batch
$P1(i)$	the processing time of order i at the first stage
$P2(i)$	the processing time of order i at the second stage

$\tau_{ii'}$ the sequence dependent setup time between orders *i* and *i'* at the first stage

H time horizon

Binary Variables:

$w_1(i, j, n)$ assignment of order *i* to unit *j* at event point *n* at the first stage

$w_2(i, j, n)$ assignment of order *i* to unit *j* at event point *n* at the second stage

$v(j, n)$ equal to 1 if unit *j* is idle at event point *n*

Continuous Variables:

$T_1^s(i, j, n)$ start time of order *i* at unit *j* at event point *n* at the first stage

$T_1^f(i, j, n)$ finish time of order *i* at unit *j* at event point *n* at the first stage

$T_2^s(i, j, n)$ start time of order *i* at unit *j* at event point *n* at the second stage

$T_2^f(i, j, n)$ finish time of order *i* at unit *j* at event point *n* at the second stage

MS makespan

Based on the above notation, the mathematical model for this problem involves the following constraints.

Allocation Constraints.

$$\sum_{j\in J1}\sum_{n\in N1} w_1(i, j, n) = 1, \qquad \forall i \in I \tag{1}$$

Constraints (1) express the requirement that each order *i* must be allocated exactly to one event point *n* at one electrolysis cell *j* at the first stage.

$$\sum_{i\in I} w_1(i, j, n) + v(j, n) = 1, \qquad \forall j \in J1, n \in N1 \tag{2}$$

$$v(j, n-1) \le v(j, n), \qquad \forall j \in J1, n \in N1, n > 1 \tag{3}$$

Constraints (2) indicate that at most one order could be produced at unit *j* at any event point *n*. If event point *n* at unit *j* is not assigned with any order, the unit will be idle and $v(j, n)$ is equal to 1. Constraints (3) enforce all empty event points to be at the end of each unit's schedule.

$$\sum_{j\in J2}\sum_{n\in N2} w_2(i, j, n) = 1, \qquad \forall i \in I \tag{4}$$

Constraints (4) ensure that each order i must be allocated exactly to one event point n and one unit j at the second stage.

$$\sum_{i \in I} w_2(i,j,n) = B^{\text{num}} \cdot (1 - v(j,n)), \qquad \forall j \in J2, n \in N2 \tag{5}$$

Constraints (5) ensure that the number of orders assigned to every event point at the units of the second stage should be equal to the predesigned number B^{num}, as long as it is not an empty event point. For systems without such managerial guideline, this set of constraints can be removed.

$$\sum_{i \in I} w_2(i,j,n) + v(j,n) \geq 1, \qquad \forall j \in J2, n \in N2 \tag{6}$$

Constraints (6) ensure that $v(j, n)$ is equal to 1 when there is no order allocated at unit j at event point n of the second stage.

$$v(j,n-1) \leq v(j,n), \qquad \forall j \in J2, n \in N2, n > 1 \tag{7}$$

Constraints (7) enforce all empty event points to be at the end of each unit's schedule in the second stage.

Capacity Constraints.

$$\sum_{i \in I} w_2(i,j,n) \cdot v(i) \leq B^{\max}, \qquad \forall j \in J2, n \in N2 \tag{8}$$

Constraints (8) express that the total weight of the orders allocated to a batch should be less than the required maximum weight of a batch.

Duration Constraints.

$$T_1^f(i,j,n) = T_1^s(i,j,n) + w_1(i,j,n) \cdot P1(i), \qquad \forall i \in I, j \in J1, n \in N1 \tag{9}$$

Constraints (9) represent the relationship between the finishing and starting times of order i at unit j at event point n at the first stage. The finishing time of order i at event point n at unit j is equal to its starting time plus processing time.

At the second stage, the processing time of event point n at unit j is equal to the summation of the processing time of assigned orders, as represented by constraints (10).

$$pTotal(j,n) = \sum_{i \in I} (w_2(i,j,n) \cdot P2(i)), \qquad \forall j \in J2, n \in N2 \tag{10}$$

Constraints (11) require that the finishing time of order *i* at unit *j* at event point *n* at the second stage must be equal to the starting time plus the total processing time.

$$T_2^f(i,j,n) = T_2^s(i,j,n) + pTotal(j,n), \qquad \forall i \in I, j \in J2, n \in N2 \tag{11}$$

For orders allocated to the same batch at the second stage, the starting time should be the same as expressed by constraints (12) and (13).

$$\begin{gathered} T_2^s(i,j,n) \geq T_2^s(i',j,n) - H(2 - w_2(i,j,n) - w_2(i',j,n)), \\ \forall i \in I, i' \in I, i \neq i', j \in J2, n \in N2 \end{gathered} \tag{12}$$

$$\begin{gathered} T_2^s(i,j,n) \leq T_2^s(i',j,n) + H(2 - w_2(i,j,n) - w_2(i',j,n)), \\ \forall i \in I, i' \in I, i \neq i', j \in J2, n \in N2 \end{gathered} \tag{13}$$

Constraints (14) and (15) represent that the finishing time of orders allocated to the same batch should be the same.

$$\begin{gathered} T_2^f(i,j,n) \geq T_2^f(i',j,n) - H(2 - w_2(i,j,n) - w_2(i',j,n)), \\ \forall i \in I, i' \in I, i \neq i', j \in J2, n \in N2 \end{gathered} \tag{14}$$

$$\begin{gathered} T_2^f(i,j,n) \leq T_2^f(i',j,n) + H(2 - w_2(i,j,n) - w_2(i',j,n)), \\ \forall i \in I, i' \in I, i \neq i', j \in J2, n \in N2 \end{gathered} \tag{15}$$

Sequence Constraints.

One of the differences between our problem and the chemical process is that each order must be allocated exactly to one event point at one unit at the two stages, so there are no sequence constraints for the same order at the same unit. We only consider the sequence of different orders at the same unit. The following constraints (16) represent the sequence at the first stage, and constraints (17) represent the sequence at the second stage, respectively.

$$\begin{gathered} T_1^s(i',j,n) \geq T_1^f(i,j,n-1) - H \cdot (2 - w_1(i',j,n) - w_1(i,j,n-1)), \\ \forall i \in I, i' \in I, i \neq i', j \in J1, n \in N1, n > 1 \end{gathered} \tag{16}$$

$$T_2^s(i',j,n) \geq T_2^f(i,j,n-1) - H \cdot (2 - w_2(i',j,n) - w_2(i,j,n-1)), \quad \forall i \in I, i' \in I, i \neq i', j \in J2, n \in N2, n > 1 \tag{17}$$

For an order *i*, the sequence between the two stages are expressed by constraints (18) that ensure the starting time of order *i* at the second stage should be greater than or equal to its finishing time at the first stage.

$$T_2^s(i,j',n') \geq T_1^f(i,j,n), \qquad \forall i \in I, j \in J1, j' \in J2, n \in N1, n' \in N2 \tag{18}$$

Sequence Dependent Changeovers.

$$T_1^s(i',j,n) \geq T_1^f(i,j,n-1) + \tau_{ii'} - H \cdot (2 - w_1(i',j,n) - w_1(i,j,n-1)), \quad \forall i \in I, i' \in I, i \neq i', j \in J1, n \in N1, n > 1 \tag{19}$$

At the first stage, if the alloy composition of the order *i'* is different from the predecessor order *i*, there is a changeover between them.

Objective.

The objective of this problem is to minimize makespan.

Bounds.

Constraints (20)-(23) represent bounds on the starting time and the finishing time of all orders at the two stages. Constraints (24) represent that the starting time of the orders at the first event point is 0.

$$T_1^s(i,j,n) \leq MS, \qquad \forall i \in I, j \in J1, n \in N1 \tag{20}$$

$$T_1^f(i,j,n) \leq MS, \qquad \forall i \in I, j \in J1, n \in N1 \tag{21}$$

$$T_1^s(i,j,n) \leq MS, \qquad \forall i \in I, j \in J2, n \in N2 \tag{22}$$

$$T_2^f(i,j,n) \leq MS, \qquad \forall i \in I, j \in J2, n \in N2 \tag{23}$$

$$T_1^s(i,j,1) = 0, \qquad \forall i \in I, j \in J1 \tag{24}$$

Constraints (25)-(27) represent bounds on the makespan. Constraints (25) represent that the makespan should be larger than or equal to the total processing time of each order at the two stages.

$$MS \geq P1(i) + P2(i), \qquad \forall i \in I \tag{25}$$

More accurate lower bounds on the makespan are given by constraints (26) and (27). Bound (26) represents that the makespan should be larger than or equal to the average processing time of all orders at the first stage plus the minimum processing time of orders at the second stage.

$$MS \geq \frac{1}{J1}\left\{\sum_{i\in I}\left[P1(i)+\min_{i'\in I}(\tau_{ii'})\right]\right\}+\text{MinP2} \tag{26}$$

Bound (27) represents that the makespan should be larger than or equal to the minimum processing time of orders at the first stage plus the average processing time of all orders at the second stage.

$$MS \geq \text{MinP1}+\frac{1}{J2}\left[\sum_{i\in I}P2(i)\right] \tag{27}$$

There are different formulations of the minimum processing times at the first stage and the second stage. The minimum processing time at the second stage is only depended on the data scale, and the minimum processing time at the first stage is more complex, which is depended on both the problem structure and the data scale of the two stages. For the second stage, the minimum process time is the minimum of the processing times of all possible batches, which could be described by (28).

$$\text{MinP2}=\min\left[\sum_{i'=1}^{B^{\text{num}}}P2(i)\right] \tag{28}$$

While for the first stage, the minimum process time depends on the number of orders and units. The following are two examples for this formulation. If there are three units at both the first and second stages, then the minimum processing time could be described by (29). And if there are nine units in the first stage and three units in the second stage, then the minimum process time could be described by (30).

$$\text{MinP1}=\min_{i,i',i''\in I}[P1(i)+\tau_{ii'}+P1(i')+\tau_{i'i''}+P1(i'')] \tag{29}$$

$$\text{MinP1}=\max\min_{i,i',i''\in I}[P1(i),P1(i'),P1(i'')] \tag{30}$$

To this end, we complete the continuous-time MILP formulation for the aluminium batching and scheduling problem.

Minimum number of event points.

One of the key issues of the unit-specific event-based modelling is to determine the number of event points, which is used as an input parameter for the solver. In the previous studies (Li and Floudas, 2010, Seid and Majozi, 2012), the general procedure of determining the optimal number of event points is to start with a small number and gradually increase it until no improvement of the objective function can be achieved. In order to do this, valid lower and upper bounds are needed for the event points.

The lower bound for the number of event points at the first stage is $\left\lceil \frac{I}{J1} \right\rceil$, which is the average number of orders that an electrolysis cell should process.

To achieve the upper bound for the number of the event points at the first stage, orders are first sequenced based on the SPT heuristic in a single machine to obtain MS_{SPT}, which is the makespan using SPT heuristic. The upper bound for the number of event points at the first stage is just equal to the maximum number of orders, of which the finishing time is less than $\left\lceil \frac{MS_{SPT}}{J1} \right\rceil$.

The lower bound and upper bound for the number of event points at the second stage can be estimated in a similar way, where the only difference is that the orders are replaced by batches.

Then the proposed method starts with the lower bounds and gradually increases until the upper bounds are reached. The numbers of event points with the best objective function value are chosen for the solver. It has to be noted that the optimal number of event points may be different for the various instances of the same scale. This is because the optimal number is related to both the parameters and the data of the instances.

In this paper, the proposed continuous-time model is used in aluminium industry which has different problem characteristics comparing with the continuous chemical engineering production process. New constraints and lower bounds are specified considering the practical problem.

4. Solution methodology

In this paper, a hybrid pointer-based DE (HYPDE) is proposed to solve the problem under consideration. The description of HYPDE is as follows.

(1) Individual representation

In each generation g of standard DE, there is a population consisting of NP variable vectors $\mathbf{P}^g = \{x_1^g, x_2^g, \cdots, x_{NP}^g\}$ and each vector $x_r^g = \{x_{r,1}^g, x_{r,2}^g, \cdots, x_{r,D}^g\}$ represents an individual, D is the dimension of the problem.

Due to the continuous nature of DE, individuals are usually encoded by real numbers. To describe the discrete scheduling problem, an operation-based representation scheme is proposed, in which the individuals are encoded as matrixes of integers. The matrix will represent the order assignments and sequencing at the first stage of production. The individual matrix **A** is given by:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1I'} \\ a_{21} & a_{22} & \cdots & a_{2I'} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MI'} \end{bmatrix} \tag{31}$$

In the matrix, element $a_{j,i}$ is an integer number which denotes an order is the ith one to be processed at unit j, $j \in J1$ and $a_{j,i} \in I$. M is the number of units at first production stage and I' is the number of orders to be processed at unit j, $I' = \frac{|I|}{|J1|}$. Each element is unique in the matrix. A solution of the problem which is also a complete schedule can be uniquely interpreted from an individual through a decoding scheme. The complete schedule is used to evaluate the solution in the selection operation.

(2) Decoding scheme

Due to the sequentially relationship of these two stages, the batching and scheduling decisions at the second stage can be established based on the decisions at the first stage via a decoding scheme. We assume that the orders which are released earlier from the first stage have priority to be processed earlier at the second stage. For each individual, once the release times of orders at the first stage are determined, the batch

composition and scheduling at the second stage could then be established. The details of the decoding scheme are expressed as follows.

Step 1. Calculate the time of all candidate orders released from the first stage. These candidate orders form a Candidate Order set ***CO***.

Step 2. Group batches at the second stage.

Step 2.1 Sort all orders in ascending order of their release times from the first stage.

Step 2.2 Assuming *nb* is the capacity of a batch. Group the *nb* earliest released orders in ***CO*** into a batch and remove these orders from ***CO***. Add this batch to Batch set ***B.***

Step 2.3 Repeat the operations in Step 2.2 until there is no order left in ***CO.***

Step 2.4 Calculate the ready times of the batches in ***B***. The ready time of a batch equals to the maximum release time of the orders in this batch.

Step 3 Schedule the batches according to the ascending order of their ready times.

Step 3.1 Allocate the batch with the earliest ready time in ***B*** to the first available unit. If there is more than one unit being available at the same time, then choose an arbitrary one. Locate this batch at the end of the batch sequence on that unit. Remove this batch from ***B***.

Step 3.2 Update available time of the unit. The new available time is equal to the processing time of the newly assigned batch add the larger one of the batch ready time and the last available time of the unit.

Step 3.3 Loop to Step 3.1 until there is no batch left in ***B.***

(3) Population initialization

Similar to the standard DE, HYPDE needs to initialize a population of *NP* individuals before iteration. According to the feature of the proposed encoding scheme, a new initialization method is proposed by (32).

$$\begin{aligned} x_r^g &= Form(\mathbf{S}), \\ \mathbf{S} &= rands(\mathbf{DSN}) \end{aligned} \tag{32}$$

DSN is an ascending sequence of D consecutive integers initializing with 0 and the function *rands*( ) is to generate a random permutation vector of elements in **DSN**. Function *Form*() is to convert integer vector **S** to a matrix. In this paper we assume that each unit will process equal amount of orders. Therefore, the function *Form*() cuts the sequence S into M segments averagely, and then group these pieces into a two-dimensional matrix, which is an individual. An example of the generation of 9-order individual is given in Figure 3.

Insert Figure 3 about here.

(4) Mutation

In this paper, an improved mutation operation is proposed in (33) based on Dong et al. (2013). The operation will adjust the mutation strategy through iterations.

$$v_r^g = \begin{cases} x_{best}^g \oplus (x_{r2}^g \,!\; x_{r3}^g),\ r2 \neq r3, & \text{if } g < g' \text{ and } rand(r) \le F \\ x_r^g, & \text{if } g < g' \text{ and } rand(r) > F \\ Swap(x_{best}^g), & \text{if } g \ge g' \text{ and } r \le 0.5 \times NP \\ x_{r1}^g \oplus (x_{r2}^g \,!\; x_{r3}^g),\ r1 \neq r2 \neq r3, & \text{if } g \ge g' \text{ and } r > 0.5 \times NP \end{cases} \tag{33}$$

Where $g' = 0.3 \times G\max$, Gmax is the maximum generation of iterations. In the proposed mutation operation, the pointer-based operator $x_{best}^g \oplus (x_{r2}^g \,!\; x_{r3}^g)$ is used at the early stage of the evolution to keep the population in good diversity. x_{best}^g is the best individual in current population. The operator $!$ in (33) corresponds to the subtraction operation in the mutation operation of DE to generate the difference vector, which is a vector consisting of the location indexes of the elements of x_{r2}^g in x_{r3}^g, while the operator $\oplus$ corresponds to the addition operation of mutation in standard DE. With $\oplus$ operator, the elements of the current best individual x_{best}^g are permutated according to the location indexes that stored in the result of $x_{r2}^g \,!\; x_{r3}^g$. Fig. 4 is an example of $x_{best}^g \oplus (x_{r2}^g \,!\; x_{r3}^g)$ mutation operator. As the evolution continues, we employ operator $Swap(x_{best}^g)$ to enhance the exploiting ability of the algorithm. Operator $Swap(x_{best}^g)$ is to swap any two randomly selected components of x_{best}^g to seek for more promising individuals in the

neighborhoods of x_{best}^{g}. However, if all mutant individuals were generated by operator $Swap(x_{best}^{g})$, the search would be easy to be trapped in local optima. Therefore, we adopt the hybrid of operator $Swap(x_{best}^{g})$ and the pointer-based operator in the later stage of evolution to achieve efficient convergence and avoid premature convergence. Figure 5 is an example of operator $Swap(x_{best}^{g})$.

Insert Figure 5 about here.

(5) Crossover

A new crossover operation is proposed in (34), which is based on point-crossover and repetitiveness repair.

$$\begin{aligned} v'^{g}_{r} = v'^{g}_{j,i} &= \begin{cases} v^{g}_{j,i}, & \text{if } rand(j,i) < CR' \\ x^{g}_{j,i}, & \text{otherwise} \end{cases} \\ u^{g}_{r} &= \begin{cases} \text{Repair}(v'^{g}_{r}), & \text{if } rand(r) < CR \\ x^{g}_{r}, & \text{otherwise} \end{cases} \end{aligned} \tag{34}$$

Function $rand(j,i)$ is to generate a real number in the range of [0,1] for each component of the individual. Each component of v'^{g}_{r} is alternatively selected from $v^{g}_{j,i}$ or $x^{g}_{j,i}$ according to the comparison of $rand(j,i)$ and *CR'*, where *CR'* is a predetermined real number. And then the $\text{Repair}(\cdot)$ operator is applied on the individual v'^{g}_{r} to fix it to be a feasible solution. The Repair operator can be described as follows: For every component $v^{g}_{j,i}$ in v^{g}_{r}, we check if it appears in v'^{g}_{r}. If so, then continue to check the next component; otherwise, we find the first repeated component in v'^{g}_{r} and replace it with $v^{g}_{j,i}$. Figure 6 is an example of the proposed crossover operation.

Insert Figure 6 about here.

(6) Selection

In HYPDE, tournament selection is adopted as shown in (35). *Makespan* is used as the evaluation criterion in the selection operation.

$$x_r^{g+1} = \begin{cases} u_r^g, \text{ if} f(u_r^g) < f(x_r^g) \\ x_r^g, \text{ otherwise} \end{cases} \quad (35)$$

The algorithm is then obtained by incorporating all operations above. Figure 7 is the pseudo-code of HYPDE.

Insert Figure 7 about here.

5. Computational experiments

To evaluate the proposed formulation and HYPDE, 20 instances are randomly generated according to the scope of the practical production data in a Chinese aluminium fabrication company. This company takes 24 hours as a production duration. Our experiments basically refer to the actual background, the scheduling period is about one day. For ease of calculation, processing times and setup times are generated as integers in the experiments. The processing times of orders at the first and second stage are between 3 hours to 8 hours, and 2 hours to 6 hours, respectively. The setup times at the first stage are between 1 hours to 4 hours. Three parameters characterize the instances: the number of orders, the number of units at the first stage and the number of units at the second stage. Instances 1-10 are established by the following parameters: the number of orders = {12, 24, 36, 48, 60}, the number of units in first stage = {4, 6, 8, 10, 12}, and the number of units at the second stage= {2}, the combinations of these parameters leads to 10 test instances. Instances 11-20 are established by the following parameters: the number of orders = {18, 27, 36, 45, 54}, the number of units in first stage = {6, 9, 12, 15, 18} and the number of units in second stage = {3}, the combinations of these parameters leads to another 10 test instances.

HYPDE was compared with CPLEX which could also provide effective lower bounds as criterions for comparison, therefore, it is no need to compare with other DEs. All computational experiments were performed on a PC with Core i5 2.0GHz CPU and 4 GB of RAM using 64-bit Windows 7 operating system. The continuous-time formulation was solved by CPLEX 12.6.1, and the proposed HYPDE was coded in C++ language. The upper limits of CPU times for CPLEX were set as 3000 seconds for each instance.

Optimal solutions of small-scale instances were obtained by CPLEX, and then Gantt charts are generated to illustrate the production process, which could verify the

correctness of the model. When solving with CPLEX, the proposed lower bound is used to determine the number of event points. This method cannot guarantee the optimal number of event points, but it can get satisfactory results in actual production.

The parameters of HYPDE were set as follows: the population size *NP* was set to be 50, the maximum number of iterations *Gmax* was set to be 200, the scaling factor *F* and crossover probability *CR* were set to be 1 and 0.5, respectively. The HYPDE algorithm was stopped when the number of evolution iterations reached *Gmax*. The computational results are presented in Table 1 and Table 3, and the model statistics for these instances are given in Table 2 and Table 4, which describe the scale of the problem instances.

In Table 1 and Table 3, the first four columns represent the number of instances, number of orders, number of machines at the first stage, and number of machines at the second stage. The following columns report the lower bound (LB), results of CPLEX, CPU times of CPLEX, results of HYPDE, and CPU times of HYPDE. Note that bold numbers in Table 1 and Table 3 show the optimal solutions obtained by either CPLEX or HYPDE.

Insert Table 1-4 about here.

Based on the results shown in these tables, the following conclusions could be drawn.

Among the 20 instances, CPLEX could obtain 1 optimal solution and HYPDE could obtain 6 optimal solutions. While comparing with CPLEX, 16 instances (80%) out of 20 are improved by HYPDE, 1 instance (5%) out of 20 is even, and 3 instances (15%) out of 20 are inferior. Addition to this, the solution time of HYPDE is much shorter than CPLEX. For instances 7-10, 20, the results of HYPDE algorithm are equal to the underlined lower bound of the problem, which means that the HYPDE has obtained the optimal solutions for these instances. For other instances, the HYPDE algorithm is still demonstrated to be effective, since the HYPDE solution is much close to the lower bound of the problem for instances 1, 6, 1-19. For instances 3,4 & 12, due to the data of production time and setup time, it is difficult for HYPDE to obtain better makespan than CPLEX. The influence of different data instances on HYPDE could be studied in the future. Therefore, HYPDE is able to generate quick and high-quality solution in the real application environment.

Insert Figure 8 about here.

Figure 8 is an illustrative line chart of the experiment results. The abscissa and ordinate denote instances and makespan, respectively. Besides, Gantt charts of the optimal solution for instance 2 obtained by CPLEX and HYPDE are illustrated in Figure 9 and Figure 10, respectively. It could be found in the Gantt charts that CPLEX and HYPDE have obtained different but both optimal scheduling solutions for instance 2.

Insert Figure 9-10 about here.

6. Conclusions

In this paper, we formulated the integrated batching and scheduling problem of aluminium electrolysis and caster as a novel unit-specific event-based continuous-time model. An event point specific strategy was proposed for the practical two-stage batching constraint, and effective bounds were also proposed to tighten the model. For small and medium scale instance, the model could be solved by CPLEX to obtain near-optimal solutions. A HYPDE algorithm with hybrid pointer-based mutation and point-cross crossover strategies was proposed to solve the problem for industrial scale instances. The results of computational experiments demonstrated the effectiveness of the proposed continuous-time formulation and the HYPDE algorithm.

Acknowledgements

The authors thank Professor Christodoulos A. Floudas for the opportunity of Qingxin Guo to study in his lab and his helpful guidance. This research is supported by the Fund for Innovative Research Groups of the National Natural Science Foundation of China (71621061) , the National Natural Science Foundation of China (71602025), the Major International Joint Research Project of the National Natural Science Foundation of China (71520107004), and the 111 Project (B16009).

References:

Bowers, M.R., Kaplan, L.A. and Hooker, T.L. 1995. “A two-phase model for planning the production of aluminum ingot.” *European Journal of Operational Research* 81(1): 105-114.

Das, S., and Suganthan, P.N. 2011. “Differential evolution: A survey of the state-of-the-art.” *IEEE Transactions on Evolutionary Computation* 15(1): 4-31.

Das, S., Mullick, S.S., and Suganthan, P.N. 2016. “Recent advances in differential evolution – An updated survey.” *Swarm and Evolutionary Computation* 27: 1-30.

Dong, Y., Guo, Q.X., and Tang, L.X. 2013. “A pointer-based discrete differential evolution.” *2013 IEEE Congress on Evolutionary Computation (CEC)* pp. 3064-3071.

Duman, E., Yildirim, M.B. and Alkaya, A.F. 2008. “Scheduling continuous aluminium casting lines.” *International Journal of Production Research* 46(20): 5701-5718.

Floudas, C. A., and Lin, X. 2004. “Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review.” *Computers and Chemical Engineering* 28: 2109–2129.

Floudas, C. A., and Lin, X. 2005. “Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications.” *Annals of Operations Research* 139(1): 131-162.

Gravel, M., Price, W.L. and Gaqné, C. 2000. “Scheduling jobs in an Alcan aluminium foundry using a genetic algorithm.” *International Journal of Production Research* 38(13): 3031-3041.

Gravel, M., Price, W.L. and Gaqné, C. 2002. “Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic.” *European Journal of Operational Research* 143(1): 218-229.

Gong, H., Tang, L.X., Duin, C.W. 2010. “A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times.” *Computers & Operations Research* 37(5): 960-969.

Gupta, J.N.D., Hariri, A.M.A., and Potts, C.N. 1997. “Scheduling a two-stage hybrid flow shop with parallel machines at the first stage.” *Annals of Operations Research* 69: 171-191.

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., Hooker, J., Méndez, C., Sand, G., and Wassick, J. 2014. “Scope for industrial applications of production scheduling models and solution methods.” *Computers and Chemical Engineering* 62: 161-193.

Hwang, F.J., and Lin, B.M.T. 2018. “Survey and extensions of manufacturing models in two-stage flexible flow shops with dedicated machines.” *Computers & Operations Research* 98: 103-112.

Ierapetritou, M. G., and Floudas, C. A. 1998a. “Effective continuous-time formulation for short-term scheduling: 1. Multipurpose batch processes.” *Industrial and Engineering Chemistry Research* 37: 4341–4359.

Ierapetritou, M. G., and Floudas, C. A. 1998b. “Effective continuous-time formulation for short-term scheduling: 2. Continuous and semi-continuous processes.” *Industrial and Engineering Chemistry Research* 37: 4360–4374.

Janak, S. L., and Floudas, C. A. 2008. “Improving unit-specific event based continuous-time approaches for batch processes: Integrality gap and task splitting.” *Computers and Chemical Engineering* 32(4–5): 913–955.

Ladurantaye, D., Gendreau, M. and Potvin, J.Y. 2007. “Scheduling a hot rolling mill.” *Journal of the Operational Research Society* 58: 288-300.

Lee, H., and Maravelias, C. T. 2017. “Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments.” *Computers and Chemical Engineering* 107: 171-183.

Lee, G.C., and Kim, Y.D. 2004. “A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness.” *International Journal of Production Research* 42(22): 4731-4743.

Lee, H., and Maravelias, C. T. 2017. “Mixed-integer programming models for simultaneous batching and scheduling in multipurpose batch plants.” *Computers and Chemical Engineering* 106: 621-644.

Li, J., and Floudas, C. A. 2010. “Optimal event point determination for short-term scheduling of multipurpose batch plants via unit-specific event-based continuous-time approaches.” *Industrial and Engineering Chemistry Research* 49(16): 7446-7469.

Li, J., Misener, R., and Floudas, C. A. 2012. “Continuous-time modeling and global optimization approach for scheduling of crude oil operations.” *AIChE Journal* 58(1): 205-226.

Li, J., Xiao, X., and Floudas, C. A. 2016. “Integrated gasoline blending and order delivery operations: Part I. short-term scheduling and global optimization for single and multi-period operations.” *AIChE Journal* 62(6): 2043-2070.

Li, J., Xiao, X., Tang, Q. H., and Floudas, C. A. 2012. “Production scheduling of a large-scale steelmaking continuous casting process via unit-specific event-based

continuous-time models: short-term and medium-term scheduling." *Industrial and Engineering Chemistry Research* 51: 7300−7319.

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., and Fahl, M. 2006. "State-of-the-art review of optimization methods for short-term scheduling of batch processes." *Computers and Chemical Engineering* 30: 913–946.

Merchan, A. F., Maravelias, C. T. 2016. "Preprocessing and tightening methods for time-indexed MIP chemical production scheduling models." *Computers and Chemical Engineering* 84: 516-535

Misener, R., and Floudas, C. A. 2014. "ANTIGONE: Algorithms for coNTinuous / integer global optimization of nonlinear equations." *Journal of Global Optimization* 59(2-3): 503-526.

Moors, E.H.M. 2006. "Technology strategies for sustainable metals production systems: a case study of primary aluminium production in the Netherlands and Norway." *Journal of Cleaner Production* 14: 1121-1138.

Prasad, P., Maravelias, C.T. and Kelly, J. 2006. "Optimization of aluminum smelter casthouse operations." *Industrial and Engineering Chemistry Research* 45: 7603-7617.

Schwindt, C. and Trautmann, N. 2003. "Scheduling the production of rolling ingots: Industrial context, model and solution method." *International Transactions in Operational Research* 10: 547-563.

Seid, E. R.; and Majozi, T. 2014. "Optimization of energy and water use in multipurpose batch plants using an improved mathematical formulation." *Chemical Engineering Science* 111: 335-349.

Seid, R., and Majozi, T. 2012. "A novel technique for prediction of time points for scheduling of multipurpose batch plants." *Chemical Engineering Science* 68(1): 54-71.

Shaik, M. A., Janak, S. L., and Floudas, C. A. 2006. "Continuous-Time Models for Short-Term Scheduling of Multipurpose Batch Plants: A Comparative Study." *Industrial & Engineering Chemistry Research* 45 (18), 6190-6209.

Shaik, M. A., and Floudas, C. A. 2007. "An Improved Unit-Specific-Event based Continuous-Time Model for Short-Term Scheduling of Continuous Processes: Rigorous Treatment of Storage Requirements." *Industrial & Engineering Chemistry Research* 46, 1764-1779.

Shaik, M. A. and Floudas, C. A. 2008. “Unit-Specific-Event based Continuous-Time Approach for Short-Term Scheduling of Batch Plants using RTN Framework.” *Computers and Chemical Engineering* 32, 260-274.

Shaik, M. A., and Floudas, C. A. 2009. “Novel Unified Modeling Approach for Short-term Scheduling.” *Industrial & Engineering Chemistry Research* 48, 2947-2964.

Shaik, M. A., and Vooradi, R. 2017. “Short-Term Scheduling of Batch Plants: Reformulation for Handling Material Transfer at the Same Event.” *Industrial and Engineering Chemistry Research* 56(39): 11175-11185.

Steinrücke, M. 2011. “An approach to integrate production-transportation planning and scheduling in an aluminium supply chain network.” *International Journal of Production Research* 49(21): 6559–6583.

Steinrücke, M. 2015. “Integrated production, distribution and scheduling in the aluminium industry: a continuous-time MILP model and decomposition method.” *International Journal of Production Research* 53(19): 5912-5930.

Storn, R., and Price, K. 1995. “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces.” Berkeley: ICSI, Available: http://icsi.berkeley.edu/~storn/litera.html.

Storn, R., and Price, K. 1997. “Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces.” *Journal of Global Optimization* 11(4): 341-359.

Tang, L.X., Dong, Y., and Liu, J.Y. 2015. “Differential evolution with an individual-dependent mechanism.” *IEEE Transactions on Evolutionary Computation* 19(4): 560-574.

Tang, L.X., Zhao, Y., and Liu, J.Y. 2014. “An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production.” *IEEE Transactions on Evolutionary Computation* 18(2): 209-225.

Vooradi, R., and Shaik, M. A. 2012. “Improved three-index unit-specific event-based model for short-term scheduling of batch plants.” *Computers and Chemical Engineering* 43: 148-172.

Yu, J.M., Huang, R., and Lee, D.H. 2017. “Iterative algorithms for batching and scheduling to minimise the total job tardiness in two-stage hybrid flow shops.” *International Journal of Production Research* 55(11): 3266-3282.

Table 1. The computational results of instance 1-10.

Problem				LB	CPLEX		HYPDE	
nos	number of orders	number of machines at stage 1	number of machines at stage 2		makespan	CPU time	makespan	CPU time
1	12	4	2	30	33	3000	32	20.129
2	12	6	2	28.7	**29**	23.7	**29**	19.317
3	24	4	2	46	55	3000	57	38.392
4	24	6	2	46	49	3000	54	42.954
5	36	4	2	77	106	3000	83	97.005
6	36	6	2	77	85	3000	78	129.15
7	48	8	2	91	104	3000	**91**	152.596
8	48	12	2	91	100	3000	**91**	176.714
9	60	10	2	124	134	3000	**124**	213.823
10	60	12	2	124	145	3000	**124**	238.131

Table 2. Model statistics for instance 1-10 from model *F*.

problem	number of binary variables	number of continuous variables	number of constraints	number of nonzeros
1	208	598	5745	20972
2	208	598	5239	18880
3	800	2346	50181	188736
4	800	2346	48019	179948
5	1776	5246	174777	662260
6	1776	5246	169807	642168
7	3136	9298	403145	1528512
8	3136	9298	385285	1456504
9	4880	14502	788211	2992920
10	4880	14502	774169	2936396

Table 3. The computational results of instance 11-20.

Problem				LB	CPLEX		HYPDE	
nos	number of orders	number of machines at stage 1	number of machines at stage 2		makespan	CPU time	makespan	CPU time
11	18	6	3	30	35	3000	34	56.069
12	18	9	3	30	31	3000	32	36.803
13	27	6	3	39	48	3000	40	43.64
14	27	9	3	39	42	3000	41	57.612
15	36	9	3	53	56	3000	55	96.81
16	36	12	3	53	57	3000	54	114.62
17	45	9	3	63	82	3000	64	168.606
18	45	15	3	63	70	3000	64	172.911
19	54	9	3	72	91	3000	73	297.154
20	54	18	3	72	77	3000	**72**	456.502

Table 4. Model statistics for instance 11-20 from model *F*.

problem	number of binary variables	number of continuous variables	number of constraints	number of nonzeros
11	456	1328	19038	70480
12	456	1328	17253	63184
13	1092	3209	73926	277294
14	1092	3209	64602	241477
15	1776	5246	161091	606988
16	1776	5246	153636	576850
17	2760	8177	324216	1226785
18	2760	8177	300720	1132003
19	3960	11756	571473	2167936
20	3960	11756	520434	1962340

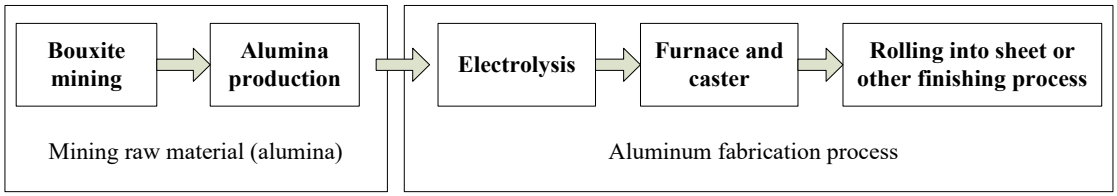

Figure 1. Aluminium production process.

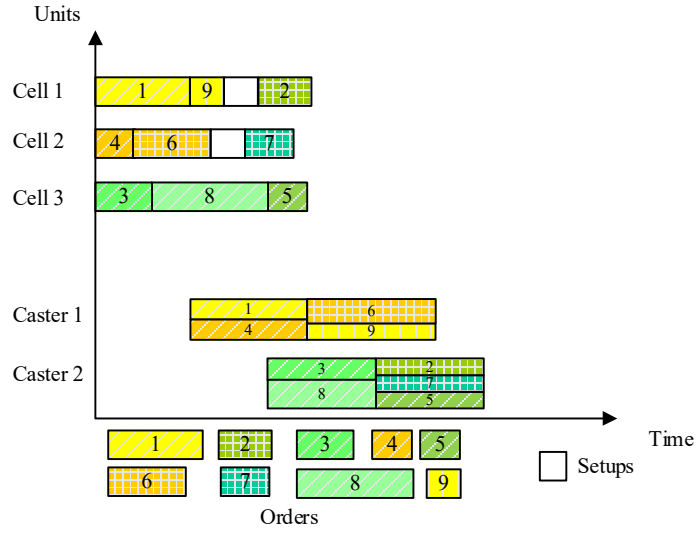

Figure 2. Schematic of batching and scheduling in aluminium electrolysis and cast production.

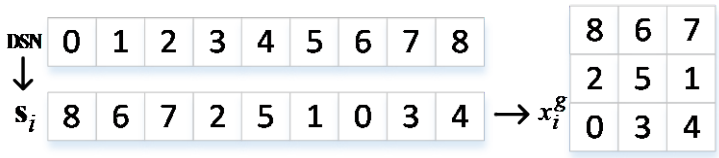

Figure 3. An example of the generation of 9-job individual.

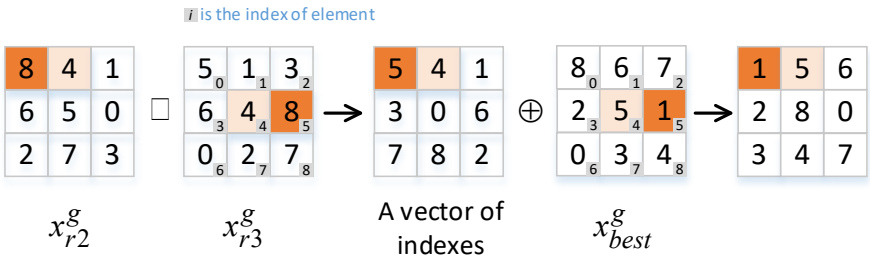

Figure 4. An example of $x_{best}^g \oplus (x_{r2}^g \,!\, x_{r3}^g)$ mutation operator

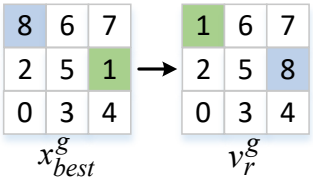

Figure 5. An example of operator $Swap(x_{best}^g)$.

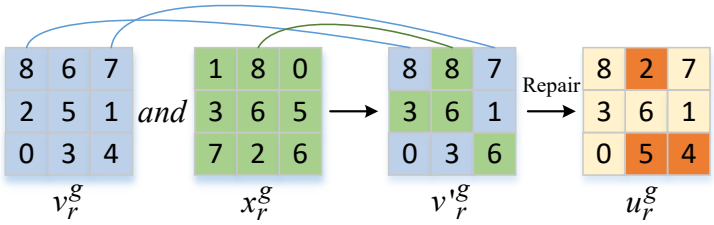

Figure 6. An example of crossover operation.

Step1: Initialization
Set the Maximum generation number *G*max, generation index $g = 0$, scaling factor $F(0,1)$, crossover rate factor $CR(0,1)$, and initialize a population of *NP* individuals
$\mathbf{P}^0 = \{x_1^0, x_2^0, ..., x_r^0 \ldots, x_{NP}^0\}$ with $x_r^0 = Form(\mathbf{S}), \mathbf{S} = rands(\mathbf{DSN})$.
Step2: Evolutionary iteration
WHILE the iteration criterion is not satisfied
Do
Step2.1: Mutation operation
FOR *r*=1 to *NP*
Generate mutant individualaccording to the proposed mutation strategy given by Equation (33)
END FOR
Step2.2: Crossover operation
FOR *r*=1 to *NP*
Generate a trial individualaccording to the crossover strategy given by Equation (34)
END FOR
Step2.3: Selection operation
Evaluate the trial individuals
FOR *r*=1 to *NP*
Select the better individuals from trial population and current population according to Equation (35)
END FOR
Step2.4: Increase the generation count
g=*g*+1
END WHILE

Figure 7. The pseudo-code of HYPDE.

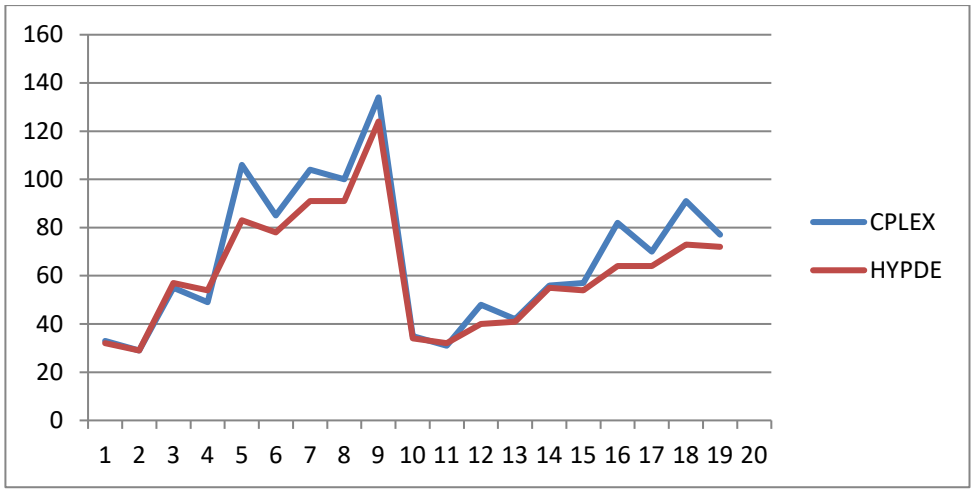

Figure 8. Line chart of experiment results.

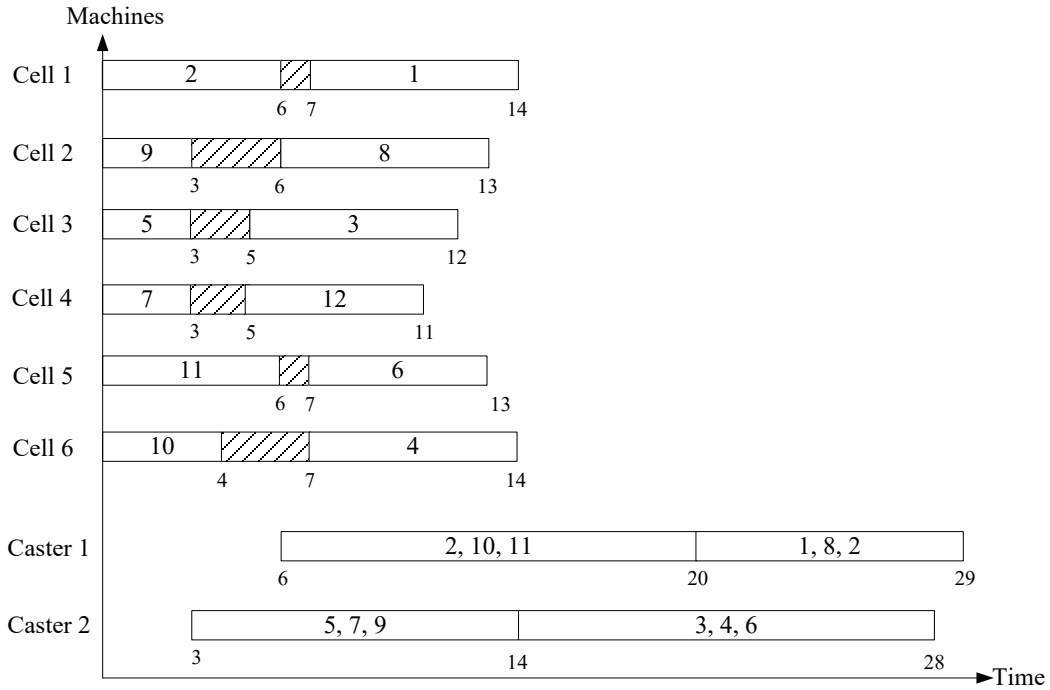

Figure 9. Gantt chart of the optimal solution for instance 2 by CPLEX.

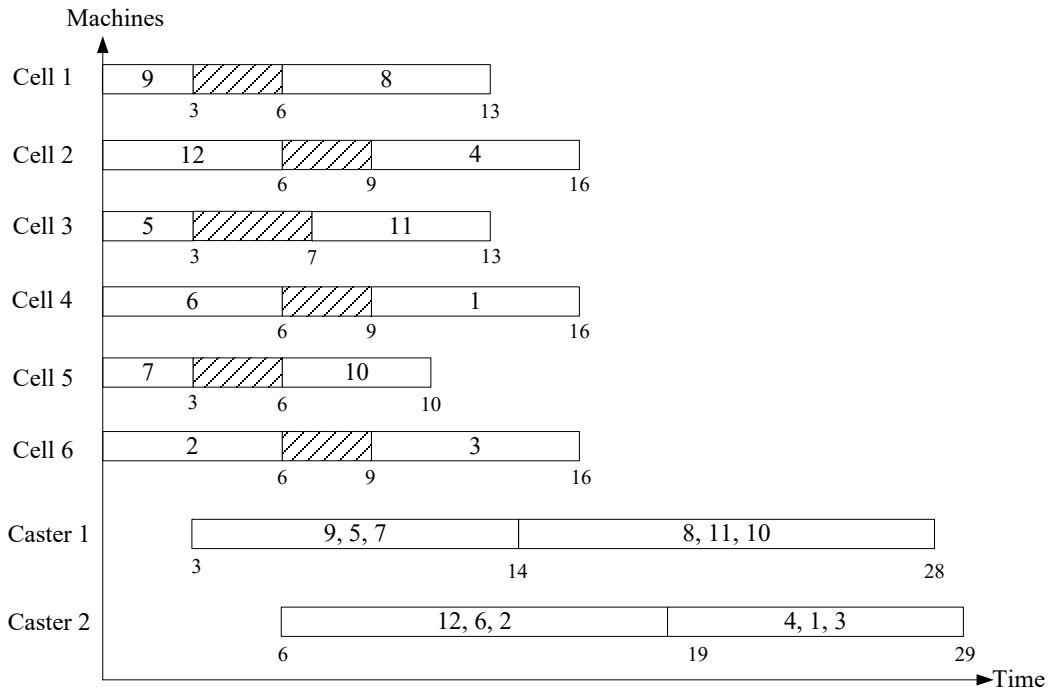

Figure 10. Gantt chart of the optimal solution for instance 2 by HYPDE.