The Batch Machine Scheduling of Continuous Caster with Flexible Jobs to Minimize Setup Costs
Dong Hong Yu and Huang Min
*Concurrent Engineering* published online 17 October 2011
DOI: 10.1177/1063293X11425192

The online version of this article can be found at:
http://cer.sagepub.com/content/early/2011/10/14/1063293X11425192
The Batch Machine Scheduling of Continuous Caster with Flexible Jobs to Minimize Setup Costs

Dong Hong Yu1,2,* and Huang Min3

1National Iron & Steel Making Plant Integration Research Center, Chongqing, 40013, China
2CISDI Chongqing Iron & Steel Making Plant Integration Co., Ltd., Chongqing, 40013, China
3College of Information Science and Engineering, Northeastern University; State Key Laboratory of Synthetical Automation for Process Industries (Northeastern University), Shenyang, Liaoning, 110819, China

Abstract: The batch machine scheduling of continuous caster with flexible jobs is first proposed based on its character. The difference with classical problem is first summarized. The preprocess of the problem based on rule is proposed, and then the mathematical model with minimizing setup costs is proposed. To improve the efficiency of local search, the particle swarm optimization (PSO) is introduced, and then the PSO combined with heuristic is embedded into the iterated local search (ILS) to solve the problem. The solution space is divided into many subspaces based on the charge information, and then the subspaces are integrated after local search. The maximum number of iteration is introduced as stopping condition. Then, the proposed hybrid algorithm and ILS are compared, and the changing of number of generation, learning factors, and the maximum number of iteration are also tested, respectively. Finally, the simulation results show that the proposed algorithm can solve the problem efficiently.

Key Words: flexible jobs, batch machine scheduling, setup costs, particle swarm optimization, iterated local search.

1. Introduction

With the increasing of challenge and competition between steel companies, more researchers pay attention to the optimization of production management within company. High consumption and high costs are the main characters in steel industry. The hybrid of modeling and optimization is one of the effective methods to solve these problems.

Most researchers focus on the scheduling problems [1]. This article focuses on batch machine scheduling of continuous caster with flexible jobs, denoted by BMSCCFJ. The objective is to minimize the setup cost based on the flexible jobs. The flexible job conception in steel industry was first proposed by Balakrishnan et al. They proposed a mathematical model to assign the orders into slabs and combine the heuristic and branch and bound method to solve the model. They obtained 7% profit [2]. Tang et al. proposed that the integrated planning problem should consider all the constraint in each processes [3]. Lee et al. proposed the continuous caster scheduling problem based on rigid jobs [4]. Chang et al. also proposed the continuous caster grouping problems based on rigid jobs [5]. Dong et al. proposed the integrated charge planning based on flexible jobs [6]. Box and Herbe proposed the continuous caster scheduling problem with fix jobs but did not give the exact mathematical model [7]. From the literature, no researchers pay attention to the continuous caster scheduling based on flexible jobs.

In this article, the BMSCCFJ with single strand is proposed, this problem can be treated as a single batch machine scheduling problem, the continuous caster is treated as machine and slab is treated as job. The difference with classical problem is as follows:

1. The jobs are flexible (e.g. jobs with time windows).
2. Jobs with the similar steel grade should be grouped to process, and different steel grade of jobs (called transform jobs) will result in setup cost. Jobs can be left.
3. The jobs should be produced with the width decreasing. Adjust width will also result in additional operation cost.
4. Each job specifies the qualification including flexible weight (in tons), the product type (steel grade) and the dimensions (length range, width, and thickness), priority (based on the due date and customer grade), surface index, rolling width and so on. The process width of each job is flexible. If the process width closer to the rolling width, the more energy will be saved. If the more weight of jobs grouped, the better utilization of continuous caster.
The classical problem is to treat continuous casting scheduling based on rigid jobs. However, this article is first focused on the flexible jobs scheduling problem of continuous casting batch machine according to the practice. The objective is to minimize above cost based on constraints.

This article considers above characters and divides this problem into three sections, first, proposed to the problem based on rules. Second, proposed the mathematical model with minimizing setup costs, and design a new hybrid method based on PSO. Finally, the analyses of the proposed algorithm are given. The results show that the proposed model and hybrid algorithm are effective.

2. Description of the BMSCCFJ Processes

2.1. BMSCCFJ

A flow of liquid steel is maintained by the oxygen converter. The liquid steel is brought into the continuous caster. First, molten steel is drained into a receptacle (tundish) at the top of the machine. A tundish is expensive and has a limited useful life: about six charges in a typical life span, so one should group charges into tundish as many as possible. Steel from the tundish flows into a long strand, and solidifies as a fully rectangle solid shape (slab). In a continuous caster, width transitions along the strand must occur gradually. For some grades of steel, width changes are limited to ‘narrowing in’ only, otherwise the casting could cause an event known as a breakout. This restriction makes it necessary to terminate the casting. The BMSCCFJ with one strand is described as shown in Figure 1.

Based on the research of Dong et al. [6,8], the problem in the first level can be solved. Here, the CAST_LOT (charges with the same steel grades can be classified into one CAST_LOT) sort should be done based on the rule and tundish planning.

2.2. Preprocess the Problem

The CAST_LOT sort is to group charges with the same steel grade and thickness with the width in decreasing order. The continuous casting scheduling problem is based on the charge planning and tundish planning. Thus, the relationship among tundish, charge, and jobs should be unchangeable. The rules are consisted with hard rules and soft rules:

Hard rules:
Rule 1: The jobs with the same steel grade (the same CC_CODE) should be processed together.
Rule 2: The CAST_LOT sorting based on tundish is based on Rule 1; one tundish can consist of several CAST_LOTs.

Rule 3: Based on Rule 2, charge is the basic unit of CAST_LOT sort.

Rule 4: Jobs should be processed in the width decreasing order, and the width jump should be less than 200 mm.

Soft rules:
Rule 5: Do not adjust width as much as possible.

The detail steps are as follows:

Step 1: Based on Rule 3, calculate the intersection of charge width as follows:

\[ WW_k = \left[ \max \left\{ W_{\text{min}}^k, \frac{W_{\text{roll}}^k}{k} \right\}, \min \left\{ W_{\text{max}}^k, \frac{W_{\text{roll}}^k}{k} + WD \right\} \right] \]

- \( WW_k \): Intersection of charge width;
- \( W_{\text{min}}^k \): minimize width of jobs in the charge \( k \);
- \( W_{\text{roll}}^k \): rolling width of jobs in the charge \( k \);
- \( W_{\text{max}}^k \): maximize width of jobs in the charge \( k \); and
- \( WD \): width jump limit 200 mm.

Step 2: Assign the CAST_LOT number sequentially based on the steel grade of charges in the tundish one by one. Charges with the same steel grade will be assigned an integer number.

Step 3: If the charges of one CAST_LOT number did not satisfy Rule 4, then, do the CAST_LOT sort again according to Rule 4 and go to step 4. Otherwise, stop the CAST_LOT sort.

Step 4: The principle of CAST_LOT sort is to reduce hybrid jobs and number of adjustment width, also match the production width as close as lower bound of charges, according to the priority. First, the relationship between charges and tundish need to be adjusted based on the experience. If the results did not satisfy Step 3, find the bottleneck jobs and delete jobs or instead jobs based on the throughput of each flow. The number of deleted or instead jobs should be less than 2 once a loop. If the results did not satisfy Step 3, the hybrid jobs need to be produced and do the CAST_LOT sort, adjust the number of CAST_LOT one-by-one. Determine whether or not Step 3 is satisfied.

3. Mathematical Model

3.1. Model Description

The following symbols are used for defining the problem parameters and variables.

- \( n \): Number of jobs (slabs)
- \( m \): Number of flow code
- \( w_{\text{min}}^k \): Lower bound of weight in flow \( m \)
- \( \Gamma = \{ J_1, \ldots, J_k \} \): Job set \( \Gamma \subseteq J \)
- \( J_l = \{ C_{g_l}^1, \ldots, C_{g_l}^{n_l} \} \): Batch \( l \) contains \( n_l \) charges
- \( C_{g_l}^j = \{ \text{job}^j_1, \ldots, \text{job}^j_{n_l} \} \): Charge \( j \) of batch \( l \) contains \( n_l \) jobs
- \( R \): Number of batch (match one CAST_LOT)
- \( n_j \): Total number of jobs within charge \( j \)
- \( n_l \): Total number of jobs within batch \( l \)
- \( n_{l}^{g_l} \): Total number of charges (job set) within batch \( l \)
- \( d_{l,j} \): Due date of job \( i \)
- \( p_{l,j} \): Processing time of the \( i \)th job in the batch \( l \)
- \( w_{l,j} \): The weight of job \( i \)
- \( w_{d_{l,j}} \): The width of job \( i \)
- \( w_{d_{l,j}} \): The width of job \( i \)
- \( C_{g_l}^j \): The width of job \( i \) in charge \( j \) and batch \( l \), related with the width of jobs in charge \( j \)
- \( t_{l,j} \): The thickness of job \( i \)
- \( l_{g_l,j} \): The length of job \( i \)
- \( c_{l,j} \): Completion time
- \( c_{l,j+1} = p_{l,j} + c_{l,j} + p_{l,j}^{\text{open}} \)
- \( p_{l,j}^{\text{open}} \): Process time with no customer order
- \( r_{l,j} \): Wait time of \( J_j \), equal to the time of final completion time of \( J_{l-1} \)
3.2. The Optimization Model

Based on the above parameters and variables, the BMSCCFJ model, denoted by BMSCCFJM, is defined as follows:

\[ \min \text{G}(\pi) \]  
\[ \text{s.t.} \]
\[ 0 \leq \omega(t_i) \leq \omega(t_i+1) \leq \sigma, \quad i = 1, \ldots, (n-1), \quad l = 1, \ldots, R \]  
\[ 0 \leq \omega(t_i) - \omega(t_i+1) \leq \sigma, \quad i = 1, \ldots, (n-1), \quad l = 1, \ldots, R \]  
\[ \omega_{\text{min}} \leq \sum_{i=1}^{R} \sum_{i=1}^{n} \omega(t_i) \omega_{\text{avg}}(t_i, l), \quad i = 1, \ldots, n, \quad l = 1, \ldots, R \]  
\[ \sum_{i=1}^{R} \omega(t_i) \leq 1, \quad i = 1, \ldots, n \]  
\[ \omega(t_i) = \{0, 1\} \]  

Here,
\[ \omega(t_i) = \begin{cases} 1 & \text{job i of batch l processed} \\ 0 & \text{otherwise} \end{cases} \]  
\[ \omega_{\text{avg}}(t_i, l) = \begin{cases} 1 & \text{job i of batch l belongs to flowm} \\ 0 & \text{otherwise} \end{cases} \]

The objective function (1) minimizes the setup costs, the constraint (2) means jobs in the same batch should be processed by width decreasing order. The constraint (3) defines neighborhood batches should be processed by the width decreasing. The constraint (4) defines the weight lower bound of different flows. The constraint (5) defines one job can be processed in only one batch at most. 
\[ \pi = \{\pi_0, \ldots, \pi_n\} \] is a schedule solution, here, 
\[ \pi_j = [\pi^{\text{eq}}_1, \ldots, \pi^{\text{eq}}_{n_j}] \] defines the batch schedule solution of \(n_j\) \(\omega\) charges in batch \(l\). 
\[ \pi_{\text{eq}}^j = \{\text{job}_{j_1}^l, \ldots, \text{job}_{j_n}^l\} \] defines the solution of \(n_j\) \(\omega\) jobs in charge \(j\).

\[ G(\pi) = w_1 f_1(\pi) + w_2 f_2(\pi) + w_3 f_3(\pi) + w_4 f_4(\pi), \]
\[ \sum_{i=1}^{4} w_i = 1. \]

Setup cost of transition job: 
\[ f_1(\pi) = \eta_1 \sum_{i=1}^{n} (R - 1) \omega(t_i). \]

The transition job is formed in the continuous caster with continuous casting of different steel grade jobs. The steel grade of transition job should be determined again. Mostly, the good steel grade of transition job is assigned to a low steel grade. This condition is not required, using cost to measure transition job; the unit is defined as 100.

Setup cost to adjust job width:
\[ f_2(\pi) = \eta_2 \sum_{i=1}^{n} \omega(t_i) \sum_{i=1}^{R - 1} \left( \omega(t_{i+1}) - \omega(t_{i}) \right). \]

To ensure the continuous casting and throughput, jobs with same grade can be continuously processed in width decreasing order. However, the width adjusted result to additional operation cost, using cost to measure adjusting width job, the unit is defined as 10.

Setup cost of throughput:
\[ f_3(\pi) = \eta_3 \sum_{i=1}^{R} \sum_{j=1}^{n} \omega_{\text{avg}}(t_i, l) \left( \sum_{i=1}^{n} \omega(t_i) \omega_{\text{avg}}(t_i, l) \cdot 300 \right). \]

The charge planning and tundish planning is to group jobs into batches. The weight of jobs in charges are flexible, if the weight of jobs with customer order in one charge is less than the capacity of charge, the loss will come. Using cost to measure adjusting width job, the unit is defined as 10.

Setup cost of job rolling width:
\[ f_4(\pi) = \eta_4 \sum_{i=1}^{n} (\omega(t_i) - \omega(t_i+1)) \omega(t_i). \]
Each job has a flexible process width. If the process width is closer to the rolling width, the more energy will be saved. Using cost to measure adjusting width job, the unit is defined as 1.

The total setup cost of scheduling solution \( \pi \) is based on the expert experience, using the weighted-sum method to deal with the above four setup cost.

Based on the three-field problem classification [9], the BMSCCFJ with single strand can be denoted as 1/s-batch, \( f_{sc} \) \( x_{c} \) \( G (\pi) \). Perceptibly, the BMSCCFJ with single-strand problem is non-deterministic polynomial-time hard problem.

### 4. Solution Methodology

#### 4.1. Particle Swarm Optimization

Kennedy and Eberhart (1995) proposed a new algorithm through simulating social behavior, which is called Particle Swarm Optimization (PSO) [10]. Different with genetic operators, these individuals are ‘evolved’ by cooperation and competition among the individuals themselves through generations. Each particle adjusts its flying according to its own flying experience and its companions’ flying experience. Each individual is named as a ‘particle’ which, in fact, represents a potential solution to a problem. Each particle is treated as a point in dimensional space. Shi and Eberhart introduced a new parameter called inertia weight [11]. The proposed PSO is as follows:

\[
v_{id}^{k+1} = w v_{id}^k + c_1 \xi_1 (p_{id}^k - x_{id}^k) + c_2 \xi_2 (g_{id}^k - x_{id}^k) \tag{7}
\]

\[
x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1} \tag{8}
\]

Here, \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}), \ v_i = (v_{i1}, v_{i2}, \ldots, v_{iD}), \) \( -v_{\text{max}} \leq v_{id} \leq v_{\text{max}}, 1 \leq i \leq m, 1 \leq d \leq D \)

Where \( c_1, c_2 \) are two positive constants, the \( w \) plays the role of balancing the global search and local search. It can be a positive constant or even a positive linear or nonlinear function of time. \( \xi_1 \) and \( \xi_2 \) are two random functions in the range 0–1. The second part of Equation (7) is the ‘cognition’ part, which represents the private thinking of the particle itself. The third part is the ‘social’ part, which represents the collaboration among the particles. Equation (7) is used to calculate the particle’s new velocity according to its previous velocity and the distances of its current position from its own best experience (position) and the group’s best experience. Then, the particle flies toward a new position according to Equation (8). The performance of each particle is measured according to a predefined fitness function, which is related to the problem to be solved. The number of particles usually set in the range 20–40, sometimes in the range 100–200 for complex problems. The length of dimensional space is determined by the optimization problem itself. The \( c_1 \) and \( c_2 \) are usually set to be 2, for other problems these two constants are in the range 0–4. The stop condition is the max iteration.

#### 4.2. Iterated Local Search

The ILS is a simple and generally applicable metaheuristic that iteratively applies local search to mutations of the current search point [12,13]. To apply ILS to a given problem, there are four components to consider:

- **Initial solution**: One can start with a random solution or one returned by some greedy construction heuristic, a good initial solution will lead to find the best solution quickly.
- **Local search**: For most problems, a local search algorithm is readily available.
- **Perturbation**: For the perturbation, a random move in a neighborhood of higher order than the one used by the local search algorithm can be surprisingly effective.
- **Acceptance criterion**: Decides to which solution is the best solution applied next.

The overall ILS procedure is pictorially illustrated as follows:

\[
X_0 = \text{InitialSolution} \quad X^* = \text{LocalSearch} (X_0) \quad \text{While (termination condition not met)} \quad X = \text{Perturbation} (X^*, \text{history}) \quad X^* = \text{Local Search} (X) \quad X^* = \text{Acceptance Criterion} (X^*, X^{**}, \text{history}) \quad \text{End} \quad \text{Output } X^*
\]

#### 4.3. Hybrid Algorithm

The hybrid algorithm based on ILS and PSO, denoted by ILSPSO, is first proposed according to the characters of the given problem. The solution space is divided into subsolution (charges) space based on the rule. The PSO is adopting as local search to solve the weight assigning of jobs in charges. Recording the current best solution based on the combination of near-best solution in subsolution space. The random kick method will be adopted, then,
continue the hybrid algorithm until it satisfies the stop condition. The scheduling rule can be classified into 113 kinds [14]. From the saving setup cost point of view, the initial solution will be achieved based on the CAST_LOT number, steel grade, and flow capacity of charges one-by-one. The overall ILSPSO procedure is pictorially illustrated as follows:

Initial solution $X = [x_1, x_2, \ldots, x_n, x_N]$;
While (NLI $\le$ NLI\text{max})
$X = \text{random kick} (X)$;
For all $x_i$
Localsearch $= \text{PSO} (x_i)$ or heuristic ($x_i$);
End
Evaluate($X$);
Renew best solution $X$;
End

5. Numerical Results

The proposed algorithms were programmed by MATLAB (R2011a, Math Works Inc) language and run on a Pentium P IV2.5GHz/2G personal computer.

To illustrate the effectiveness and the performance of the proposed model and algorithm in an actual steel corporation, the job types and sizes are randomly selected with 681 slabs from a large-scale iron–steel plant in China. There are 678 jobs left based on the CAST_LOT. These 678 jobs can be divided into 10 problems (see Table 2). The parameters are as follows:

$WD = 200\text{mm}, \quad v = 1500\text{mm/min}, \quad \rho = 7.85\text{t/m}^3$,

$\sigma = 200\text{mm}, \quad \eta_1 = \frac{50}{T}, \quad \eta_2 = \frac{R}{\sum_{l=1}^{R} n_l^n}, \quad \eta_3 = \frac{10}{\sum_{l=1}^{R} n_l^n}$,

$\eta_4 = \frac{1}{100n}$.

The statistical characteristics for 678 jobs are shown in Table 1.

The parameters of ILSPSO are as follows:
Number of particles is 40, $w$ is a random function in the range 0–1, the max number of iteration for ILSPSO is 50, $w_1 = 0.5$, $w_2 = w_3 = 0.2$, and $w_4 = 0.1$.

The following will give an analysis of parameters (number of local iteration (NLI), number of generation (NG), $c_1$, and $c_2$).

The analysis between ILSPSO and ILS are in Table 2. The ‘best rate’ is tested by 100 runs involving different random seeds. The best rate of ILSPSO is 100%. However, the best rate of ILS is 0%. The ‘bad’ means the worst result within the max number of iteration (50) in ILSPSO.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Instance size</th>
<th>NLI</th>
<th>Bad</th>
<th>Best</th>
<th>Best rate</th>
<th>Variation rate (%)</th>
<th>CPU time/times</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILS [12]</td>
<td>224</td>
<td>50</td>
<td>26.7813</td>
<td>26.7813</td>
<td>0</td>
<td>---</td>
<td>1.6740*</td>
</tr>
<tr>
<td></td>
<td>276</td>
<td>50</td>
<td>35.2459</td>
<td>35.2459</td>
<td>0</td>
<td>---</td>
<td>1.7659*</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>50</td>
<td>45.6702</td>
<td>45.6702</td>
<td>0</td>
<td>---</td>
<td>1.8750*</td>
</tr>
<tr>
<td></td>
<td>378</td>
<td>50</td>
<td>52.3592</td>
<td>52.3592</td>
<td>0</td>
<td>---</td>
<td>2.0121*</td>
</tr>
<tr>
<td></td>
<td>426</td>
<td>50</td>
<td>54.7928</td>
<td>54.7928</td>
<td>0</td>
<td>---</td>
<td>2.2757*</td>
</tr>
<tr>
<td></td>
<td>472</td>
<td>50</td>
<td>55.6902</td>
<td>55.6902</td>
<td>0</td>
<td>---</td>
<td>2.4123*</td>
</tr>
<tr>
<td></td>
<td>524</td>
<td>50</td>
<td>57.2876</td>
<td>57.2876</td>
<td>0</td>
<td>---</td>
<td>2.5752*</td>
</tr>
<tr>
<td></td>
<td>572</td>
<td>50</td>
<td>59.7639</td>
<td>59.7639</td>
<td>0</td>
<td>---</td>
<td>2.6129*</td>
</tr>
<tr>
<td></td>
<td>622</td>
<td>50</td>
<td>61.7865</td>
<td>61.7865</td>
<td>0</td>
<td>---</td>
<td>2.8753*</td>
</tr>
<tr>
<td></td>
<td>678</td>
<td>50</td>
<td>63.9995</td>
<td>63.9995</td>
<td>0</td>
<td>---</td>
<td>2.9690*</td>
</tr>
<tr>
<td>ILSPSO</td>
<td>224</td>
<td>50</td>
<td>26.7813</td>
<td>26.2013</td>
<td>100</td>
<td>2.21</td>
<td>73.232</td>
</tr>
<tr>
<td></td>
<td>276</td>
<td>50</td>
<td>35.2459</td>
<td>34.679</td>
<td>100</td>
<td>1.63</td>
<td>76.726</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>50</td>
<td>45.6702</td>
<td>45.1368</td>
<td>100</td>
<td>1.18</td>
<td>78.375</td>
</tr>
<tr>
<td></td>
<td>378</td>
<td>50</td>
<td>52.3592</td>
<td>51.532</td>
<td>100</td>
<td>0.39</td>
<td>81.973</td>
</tr>
<tr>
<td></td>
<td>426</td>
<td>50</td>
<td>54.7928</td>
<td>54.2087</td>
<td>100</td>
<td>0.39</td>
<td>84.083</td>
</tr>
<tr>
<td></td>
<td>472</td>
<td>50</td>
<td>55.6902</td>
<td>55.1034</td>
<td>100</td>
<td>1.06</td>
<td>87.798</td>
</tr>
<tr>
<td></td>
<td>524</td>
<td>50</td>
<td>57.2876</td>
<td>57.0138</td>
<td>100</td>
<td>0.41</td>
<td>91.627</td>
</tr>
<tr>
<td></td>
<td>572</td>
<td>50</td>
<td>59.7639</td>
<td>59.3027</td>
<td>100</td>
<td>0.82</td>
<td>94.562</td>
</tr>
<tr>
<td></td>
<td>622</td>
<td>50</td>
<td>61.7865</td>
<td>61.4312</td>
<td>100</td>
<td>0.58</td>
<td>96.973</td>
</tr>
<tr>
<td></td>
<td>678</td>
<td>50</td>
<td>63.9995</td>
<td>63.7383</td>
<td>100</td>
<td>0.41</td>
<td>100.687</td>
</tr>
</tbody>
</table>

ILS: iterated local search, ILSPSO: iterated local search and particle swarm optimization.
*After 1500 runs, the solution did not get charged, and CPU time is more than 110 s.

Table 1. Description of numerical data.

<table>
<thead>
<tr>
<th>CC_CODE</th>
<th>No. of charges</th>
<th>No. of CAST_LOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>130160</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>130260</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>140740</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>140450</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. Analysis of algorithms.
As \( c_1 \) and \( c_2 \) are usually set to be \( 2 \) for PSO, Figure 2 shows that a better solution will be found when \( c_1 = c_2 = 2.0 \) and \( \text{NG} = 20 \). Figure 3 (\( c_1 = c_2 = 2.0, \text{NG} = 20 \)) shows that with \( \text{NLI} \) increasing, the setup costs are converging to 63.7401. The analysis of different \( c_1 \) and \( c_2 \) are tested in Table 3 (\( \text{NG} = 20 \)), the results show that when \( c_1 = c_2 = 3.4 \), it can obtain a better solution (63.7386) by far.

Figure 4 shows the analysis of charging \( \text{NG} \) of ILPSO (\( c_1 = c_2 = 3.4 \)), the results show that a better solution will be found when \( \text{NG} = 25 \). Figure 5 (\( c_1 = c_2 = 3.4, \text{NG} = 25 \)) shows that with \( \text{NLI} \) increasing, the setup costs are converging to 63.7383. It shows that the convergence of the proposed algorithm is better. The analysis of different \( c_1 \) and \( c_2 \) are tested in Table 4 (\( \text{NG} = 25 \)), the results show that the setting of \( c_1 = c_2 = 3.4 \) is better.

Combining with above analysis and Figures 6 and 7, a conclusion, which is the best solution, will be found when \( c_1 = c_2 = 3.4 \) (\( \text{NG} = 25 \)). Namely, when

---

**Table 3. Results of ILPSO (NG = 20).**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( c_1, c_2 )</th>
<th>NLI</th>
<th>Bad</th>
<th>Best</th>
<th>Variation rate (%)</th>
<th>CPU time/times</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILPSO</td>
<td>0.4</td>
<td>50</td>
<td>63.9995</td>
<td>63.9978</td>
<td>0.003</td>
<td>82.92</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>50</td>
<td>63.9995</td>
<td>63.9967</td>
<td>0.004</td>
<td>83.58</td>
</tr>
<tr>
<td></td>
<td>2.4</td>
<td>50</td>
<td>63.9995</td>
<td>63.7840</td>
<td>0.338</td>
<td>84.13</td>
</tr>
<tr>
<td></td>
<td>3.4</td>
<td>50</td>
<td>63.9995</td>
<td>63.7386</td>
<td>0.409</td>
<td>81.66</td>
</tr>
<tr>
<td></td>
<td>0.4, 0.6</td>
<td>50</td>
<td>63.9995</td>
<td>63.9928</td>
<td>0.011</td>
<td>82.64</td>
</tr>
<tr>
<td></td>
<td>1.4, 1.6</td>
<td>50</td>
<td>63.9995</td>
<td>63.9995</td>
<td>—</td>
<td>83.98</td>
</tr>
<tr>
<td></td>
<td>2.4, 2.6</td>
<td>50</td>
<td>63.9995</td>
<td>63.8949</td>
<td>0.164</td>
<td>84.11</td>
</tr>
<tr>
<td></td>
<td>3.4, 3.6</td>
<td>50</td>
<td>63.9995</td>
<td>63.7839</td>
<td>0.338</td>
<td>83.33</td>
</tr>
</tbody>
</table>

ILPSO: iterated local search and particle swarm optimization.

---

**Figure 2.** Iterated local search and particle swarm optimization (ILPSO) with changing \( \text{NG} (c_1 = c_2 = 2.0) \).

**Figure 3.** Iterated local search and particle swarm optimization (ILPSO) with changing \( \text{NLI} (c_1 = c_2 = 2.0, \text{NG} = 20) \).

**Figure 4.** Analysis of iterated local search and particle swarm optimization (ILPSO) with changing \( \text{NG} (c_1 = c_2 = 3.4) \).

**Figure 5.** Iterated local search and particle swarm optimization (ILPSO) with changing \( \text{NLI} (c_1 = c_2 = 3.4, \text{NG} = 25) \).
Table 4. Result of ILSPSO (NG = 25).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$c_1$, $c_2$</th>
<th>NLI Bad</th>
<th>Best</th>
<th>Variation rate (%)</th>
<th>CPU time/times</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILSPSO</td>
<td>0.4, 50</td>
<td>63.9995</td>
<td>63.9978</td>
<td>0.003</td>
<td>103.77</td>
</tr>
<tr>
<td></td>
<td>1.4, 50</td>
<td>63.9995</td>
<td>63.9902</td>
<td>0.015</td>
<td>100.81</td>
</tr>
<tr>
<td></td>
<td>2.4, 50</td>
<td>63.9995</td>
<td>63.7821</td>
<td>0.341</td>
<td>101.84</td>
</tr>
<tr>
<td></td>
<td>3.4, 50</td>
<td>63.9995</td>
<td>63.7383</td>
<td>0.410</td>
<td>100.99</td>
</tr>
<tr>
<td></td>
<td>0.4, 0.6, 50</td>
<td>63.9995</td>
<td>63.9925</td>
<td>0.011</td>
<td>101.08</td>
</tr>
<tr>
<td></td>
<td>1.4, 1.6, 50</td>
<td>63.9995</td>
<td>63.9995</td>
<td>—</td>
<td>101.08</td>
</tr>
<tr>
<td></td>
<td>2.4, 2.6, 50</td>
<td>63.9995</td>
<td>63.8704</td>
<td>0.202</td>
<td>100.73</td>
</tr>
<tr>
<td></td>
<td>3.4, 3.6, 50</td>
<td>63.9995</td>
<td>63.7832</td>
<td>0.339</td>
<td>100.88</td>
</tr>
</tbody>
</table>

ILSPSO: iterated local search and particle swarm optimization.

Figure 6. Iterated local search and particle swarm optimization (ILSPSO) with changing $c_1$ and $c_2$ (NG = 25).
6. Conclusions

This article focuses on the BMSCCFJ with single strand to minimize the setup costs. The problem is divided into three sections. First, the CAST_LOT sort problem is proposed based on the rules. Then, the mathematical model with minimizing setup cost is proposed, and the hybrid algorithm based on ILS and PSO is designed. Here, the PSO is used for local search to solve the continuous optimization problem. The maximum number of iteration and kick method are introduced to improve the efficiency of hybrid algorithm. Based on the actual data, the proposed algorithm and ILS are compared, and the changing of learning factors, number of generation, and the maximum number of iteration are also tested, respectively. Finally, the analysis results show that the proposed algorithm can solve the problem efficiently.

$c_1 = c_2 = 3.4$ (NG = 25), the ILPSO based on the above parameters can save more setup costs for BMSCCFJM.

Figure 7. Iterated local search and particle swarm optimization (ILPSO) with changing $c_1$ and $c_2$ (NG = 25).
Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No.71071028, No. 71021061, No.70931001 and No.61070162, Specialized Research Fund for the Doctoral Program of Higher Education under Grant No.20070145017 and No. 20100042110025, the Fundamental Research Funds for the Central Universities under Grant No.N090504006, No. N100604021 and No. N090504003.

References


Dong Hong Yu

Hong Yu Dong received the PhD degree in systems engineering from Northeastern University, Shenyang, China, in 2009. He joined the Department of Automation, National Iron & Steel Making Plant Integration Research Center, Chongqing, China, in 2009. He has authored and coauthored many international journal and conference papers and 1 patent. His main research interests are production planning and scheduling, optimization, expert system, simulation based on optimization, and their applications.

Min Huang

Min Huang is a professor in the Department of Systems Engineering, the College of Information Science and Engineering, Northeastern University in Shenyang, China. She is currently a member of INFORMS. She has published many papers on international journals and conference. She has investigated the National Natural Science Foundation of China, the National High-Tech Research and Development Plan (863 planning), the Scientific Research Foundation for the Returned Overseas Chinese Scholars of State Education Ministry, and Liaoning Province Natural Science Foundation of China, etc as principal investigator. She was awarded of the Science and Technology Achievement of State, the Science and Technology Achievement of Education Ministry and the Natural Science Achievement of Liaoning province, etc. And she was honored by the Program for New Century Excellent Talents in University of Education Ministry, etc. Her researches are mainly devoted to production planning and inventory control, risk management for VE, optimization for 4th party logistics, as well as soft computing.