Low Carbon Scheduling with Iterative Ant Colony Algorithm

Peng Liang\textsuperscript{1,2,*}, WenSi Chen\textsuperscript{1}, MingQiang Luo\textsuperscript{1} and ShuQin Chen\textsuperscript{1}

\textsuperscript{1}School of Computer Science, GuangDong Polytechnic Normal University, Guangzhou, 510665
\textsuperscript{2}R&D Center, XingFa Aluminum Holdings Limited, Foshan, China, 528061
* Corresponding author, e-mail: cs\_phoenix\_liang@163.com

Abstract

This research considers a low carbon scheduling problem in unrelated parallel machines. To solve this problem, we first establish a low carbon scheduling mathematical model. Then an iterative ant optimization algorithm is presented. Furthermore, parameters of proposed iterative ant optimization algorithm are selected by Taguchi methods on generating test dataset. Finally, comparative experiments indicate the proposed iterative ant optimization algorithm has better performance on minimizing energy consumption as well as total tardiness.

Keywords: ant optimization; energy consumption; low carbon scheduling; Unrelated parallel machine

1. Introduction

In recent years, Great efforts towards energy saving are increased due to sequence of serious environmental impacts and rising energy cost. The most important part of machine energy consumption is machines idling. It is observed that if the inter arrival time until the next job is longer than a breakeven duration, then turning off the machine until the arrival of the next job provides significant energy savings\cite{1}. Especially, a large amount of energy is wasted while keeping idle machine running (i.e. not processing jobs but still running machine) \cite{2-3}. Research on Wichita shows that turning off machines while machines stand idle can save at least 13\% of total energy consumption. Kordonowy investigates that more than 30\% of input energy is consumed by background processes \cite{4}.

Only a few references consider the objective of energy consumption considered energy consumption in control systems to extend the life of batteries \cite{1} \cite{5-6}. Mouzon and Yildirim considered the problem of minimizing total energy consumption and total tardiness on signal machine \cite{7}. The total energy consumption is measured by summation of idle power and machine setup power. However, the key to save energy on single machine problem is to determine if the machine should be turned off during idle time. Yildirim and Mouzon proposed a genetic algorithm to solve minimizing total energy consumption as well as max completion time on signal machine \cite{8}.

Actually, in most of manufacturing systems, efficient machines and inefficient machines are working together. What’s more, the due dates of jobs are as important as the energy consumption costs in scheduling problems. Ant colony optimization (ACO) algorithm has become more popular to solve combinatorial optimization problems \cite{9-11}. Yagmahan and Yenisey proposed a multi-objective ant colony system algorithm to solve a flow shop scheduling problem with respect to both of makespan and total flowtime \cite{12}. Lin et al. consider an ACO algorithm to solve the problem of scheduling unrelated parallel machines to minimize total weighted tardiness \cite{13}. Arnaout et al. addressed the non-preemptive unrelated parallel machine scheduling problem with machine dependent and sequence-dependent setup times via a modified ACO algorithm \cite{14}. In this paper, we
begin the research of minimizing energy consumption and total tardiness on unrelated parallel machine. The problem is formulated by a weighted summation of energy consumption and total tardiness. For solving this problem, we develop an iterative ACO (IACO) algorithm in which a machine reselection operation is applied.

2. Problem Definition

According to the description above, we first define a a mathematical model for unrelated parallel machines with the objective of minimizing energy consumption and total tardiness. Given \( n \) independent jobs and jobs are processed on \( m \) parallel machines. The constraint is each job can be processed by only one machine. Machines are not stop until all jobs are finished. Assuming a job \( j \) arrives at time \( r_j \), has a process time \( p_{ij} \) on machine \( i \), and a due date \( d_j \). The total tardiness can be represented as \( \sum max(C_j - d_j, 0) \), where \( C_j \) is denoted as completion time.

Assuming machine \( i \) consumes power \( P_{idle}^i \) while machine stands idle and consumes power \( E_{setup}^i \) when it turns off and then turns on (i.e. a setup occur). In order to save energy efficiently, a breakeven duration \( T_B^i \) is denoted as Machine \( i \) a setup operation instead of keeping machine at idle, which is represented in Equation (1):

\[
T_B^i = \frac{E_{setup}^i}{P_{idle}^i}
\]  

(1)

3. Mathematical Model

Basic notions:
- \( m \): the number of machines;
- \( n \): the number of jobs;
- \( J_j \): the job \( j \), \( j=1, 2,\ldots, n; \)
- \( M_i \): the machine \( i \), \( i=1, 2,\ldots, m; \)
- \( H_i \): the number of jobs allocated on machine \( M_i \);
- \( w_1 \): weight associated with total tardiness;
- \( w_2 \): weight associated with energy consumption;
- \( c_j \): the completion time of job \( J_j \);
- \( r_j \): the release time of job \( J_j \);
- \( d_j \): the due date of job \( J_j \);
- \( t_i \): the makespan of scheduled jobs on machine \( M_i \);
- \( p_{ij} \): the process time of job \( J_j \) on machine \( M_i \);
- \( P_{j tardiness}^i \): per unit time cost of job \( J_j \) tardiness;
- \( P_{idle}^i \): per unit time energy consumption of machine \( M_i \)

Decision variables:
\[
y_{ik} = \begin{cases} 
  p_{idle}^i (c_j - p_{ij} - c_k), & \text{if } (c_k - p_{ik} - c_j) < T_B^i \\
  E_{setup}^i, & \text{if } J_j \text{ precedes } J_k \\
  0, & \text{otherwise}
\end{cases}
\]  

(2)

The definition of minimizing energy consumption and total tardiness on unrelated parallel machines is formulated as follows.

\[
\min \left( w_1 \sum_{j=1}^n P_{j tardiness}^i \max(0, c_j - d_j) + w_2 \sum_{i=1}^m \sum_{k=1}^{H_i} \sum_{l=1}^{H_j} y_{ik} \right)
\]  

(3)
\[ c_i - p_i \geq r_j, c_i - p_i \geq c_i \text{ or } c_i - p_i \geq c_j \]  

(4)

If \((c_i - p_i - c_i) < T^i_j\) and job \(J_j\) immediately precedes job \(J_i\) on machine \(i\) then \(y_{ia} = P_{ia}(c_i - p_i - c_i)\), else \(y_{ia} = E_{step}\)  

(5)

\(\forall i = 1, 2, ..., m\) \(\forall j = 1, 2, ..., n\)

\(\forall l = 1, 2, ...H_j\) \(\forall k = 1, 2, ...H_j, k \neq l\)

4. Iterative Ant Colony Optimization (IACO)

4.1. Solution Construction

Low carbon scheduling in unrelated parallel machines to minimize energy consumption and total tardiness required two decisions: assignment and job sequence, which will result in a huge solution space. Consequently, the two decisions are often addressed independently to reduce the solution search space. Such as select the first available machine, and then distribute the minimization total tardiness job. Although this strategy could significantly reduce the search space, appealing solutions may be excluded due to the independent decision heuristic. Inspired by ATC heuristic rule proposed by Lin et al. (2013), we use an iterative way to construct solution by three step: first, machine selection, then job selection, and finally machine reselection.

4.2. Machine Selection

First, a machine will be selected according Equation (6) and Equation (7). \(q_m\) is a random number from uniform distribution \([0, 1]\) and \(q_m0=0.9\) is a user-specified number. If \(q_m < q_m0\) an ant is apt to select the smallest makespan machine among all unrelated parallel machines according to Equation (6), otherwise a machine \(i\) is chosen according to the probability distribution \(P_t\) defined in Equation (7).

\[
i^* = \begin{cases} 
\min_{i \in \Pi_m} t_i, & \text{if } q_m < q_m0 \\
i, & \text{otherwise}
\end{cases}
\]  

(6)

\[
P_t = \frac{1}{\sum_{t=1}^{m} 1/t_t} \quad i = 1, 2, ..., m
\]  

(7)

4.3. Job Selection

We consider heuristic information and pheromone trails together in job selection. \(q_j\) is a random number from uniform distribution \([0, 1]\) and \(q_j0=0.9\) is a user-specified number in Equation (8). If \(q_j < q_j0\) an ant is apt to select the smallest tardiness job \(J_j\) processed on machine \(i^*\), otherwise a job \(J\) is chosen according to the probability distribution \(P_{j_t}\) defined in Equation (9). Pheromone trails \(\tau_{j_t}(t)\) indicates the favorability of assigning job \(j\) to a machine \(i^*\) and set to 0 initially. \(\eta_{j_t}(t)\) in Equation (10) is heuristic information which suggests the greedy heuristic of processing the job \(j\) on machine \(i^*\) that takes the least amount of tardiness. Parameters \(\alpha\) and \(\beta\) are the relative importance of pheromone trails and heuristic information respectively. \(\Psi\) represents a set of unscheduled jobs in Equation (9).
\[ j^* = \begin{cases} \max_i \left( \tau_{j, i} (t) \right)^\alpha \cdot \eta_{j, i} (t)^\beta, & \text{if } q_j < q_{j0} \\ J, & \text{otherwise} \end{cases} \] (2)

\[ P_{i, j} = \begin{cases} \left( \tau_{j, i} (t) \right)^\alpha \cdot \eta_{j, i} (t)^\beta, & \text{if } j \in \Psi \\ \sum_{i \in \Psi} \left( \tau_{j, i} (t) \right)^\alpha \cdot \eta_{j, i} (t)^\beta, & \text{otherwise} \end{cases} \] (9)

\[ \eta_{j, i} (t) = \frac{1}{p_{\text{setup}} \times \max \{ t_i + p_{j, i} - d_i, 0 \} + 1} \] (10)

4.4. Machine Reselection

An ant will select machine \(i^**\) according to Equation (11), which aims at minimizing the weighted sum of energy consumption and total tardiness when process job \(j^*\) on machine \(i^**\).

\[ i^** = \arg \min_{i^*} \{ U_{j, i^*} \times \min \{ E_{\text{setup}}, P_{\text{idle}} (r_i - t_i) \} + P_{\text{setup}} (1 - U_{j, i^*}) \times \max \{ t_i + p_{j, i^*} - d_i, 0 \} \} \] (11)

\[ U_{j, i^*} = \begin{cases} 0, & \text{if } r_i \leq t_i \\ 1, & \text{otherwise} \end{cases} \]

After three steps (machine selection, job selection and machine reselection) are executed, a job \(j\) is assigned to a machine \(i^*\). Repeated the operations above until all jobs are distributed, then a solution construction is finished.

4.5. Local Search

Local search strategies can obtain better solutions. However, it is complicated to search in the whole solution space since search computation complexity is high. In order to simplify the local search, two local search strategies (LS1 and LS2) are proposed to improve IACO algorithm. The first local search strategy (LS1) constructs new solutions by swapping jobs on the same machine. The second procedure (LS2) searches for new solution by transferring jobs from the machine with the highest objective value to the machine with the lowest one. The local search algorithm for LS1 and LS2 search strategies is described as followed:
5. Computational Experiments and Results

5.1. Data Generation

The proposed algorithm is implemented in Matlab R2012b running on Windows 7 with Intel core i5 2.30GHz and 4 Gigabytes RAM. The test datasets are divided into three different sizes namely small, medium and large which is shown in Table 2. Processing times $p_j$ are generated randomly from uniform distribution [1, 3]. Release times $r_j$ are generated randomly from uniform distribution [1, 30]. Due dates of jobs $d_j$ are generated by TWK (total work-content) method and calculated by Equation (14), where $c$ represents the relaxation coefficient and is set to 2, 4 and 8. The per unit time of job tardiness cost $E_{screw}'/P_{idle}'$ is calculated by Equation (15). The unit time of machine idle power consumption $P_{idle}$ is generated randomly from uniform distribution [1, 3]. We use $E_{screw}'/P_{idle}'$ ratio to define this instances and set $E_{screw}'/P_{idle}'$ ratios are 2, 4 and 8. Equal relative weightings chosen to total tardiness and energy consumption for total objective value are $w_1=w_2=0.5$ respectively. Parameter combination is suggested as $\alpha=0.01, \beta=0.15, Num_Ants=5$.

$$d_j = r_j + c \times \sum_{i=1}^{\infty} p_{ij} \cdot t_m$$

(14)
\[ P_{\text{setup}}^i = \sum_{j=1}^{m} P_{ij} / m \]  \hspace{1cm} (15)

Table 1. Parameter Setting of the Main Factors in Experimental Design

<table>
<thead>
<tr>
<th>Factor</th>
<th>Count</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ( n/)Machine ( m )</td>
<td>3</td>
<td>20/5 50/5 100/10</td>
</tr>
<tr>
<td>( c )</td>
<td>3</td>
<td>2 4 8</td>
</tr>
<tr>
<td>( E_{\text{setup}}^i P_{\text{idle}}^i )</td>
<td>3</td>
<td>2 4 8</td>
</tr>
</tbody>
</table>

When the data are generated, all the level combinations result in \( 3 \times 3 \times 3 = 27 \) test problem configurations.

In order to evaluate the performance of propose IACO algorithm, we first compare the IACO algorithm with a comparative algorithm named GRASPTETT, which is a multi-objective algorithm to solve the minimization problem of energy consumption and total tardiness on single machine. For more details of GRASPTETT, see Mouzon and Yildirim [7]. We extend the GRASPTETT algorithm on unrelated parallel machines in this paper by using a well-known earliest release time heuristic to assign machines. What’s more, to validate our modified ATC heuristic rule (machine selection, job selection and machine reselection), we also compare IACO with original ACO algorithm (OACO) which only adopts ‘machine first, schedule job second’ solution construction strategy. We incorporate the same parameter setting of OACO and other important parameters \( (E_{\text{setup}} P_{\text{idle}} \) and \( c \) ) in this paper.

5.2. Performance Measure

The relative percentage deviation (RPD) is used to evaluate the performance of multi-objective optimization algorithms. Given an obtained objective value by selected optimization algorithm, The RPD can be defined in Equation (16) as followed:

\[ RPD = \frac{\text{Value}_{\text{sol}} - \text{Min}_{\text{sol}}}{\text{Min}_{\text{sol}}} \times 100\% \]  \hspace{1cm} (16)

where \( \text{Min}_{\text{sol}} \) is the best objective value obtained for each problem configuration.

5.3. Comparative Results

In this section, our proposed IACO algorithm is tested on all 27 problem configurations. Each problem configuration generates 3 instances and each test is repeated with 5 runs for each instance. Parameter settings are the same as discussed in the last section. Performance of solutions to yield using test problem is compared two multi-objective optimization algorithms: GRASPTETT and OACO. The computational results of average RPD for all problem configurations are shown in Table 2 respectively.

As can be seen in Table 2, the IACO algorithm performs better than other two approaches in all problem configurations. The mean RPD values of all three algorithms are consistent when job number and machine number \( n/m \) are increasing. The mean RPD value for all tests of IACO algorithm is 0.96, when the mean RPD value of GRASPTETT algorithm is 4.85 which is 3.89 higher than IACO algorithm. The OACO algorithm shows the weakest performance with 5.95 mean RPD. The factor \( n/m \) has significant influence on GRASPTETT and OACO. Furthermore, machine reselection heuristic rule is crucial.
for solution construction since IACO algorithm outperforms OACO in all instances. When relaxation coefficient $c$ is small ($c=2$), the performances of GRASPTETT and OACO are acceptable, especially in small problem size where $n = 20$ and $m = 5$. For the reason that there are only little scheduling space when the due dates are not well spread and waiting time between release time and due date is small. With the increasing of relaxation coefficient $c$, the differences between IACO and compared approaches become larger, for the reason that the bigger margin between release time and due date, the less probability of job tardiness occurs. By increasing the ratio of $E_{\text{setup}}/p_{\text{idle}}$, which means to increase the length of breakeven duration, all approaches have a little fluctuation. It could be explained by machines trend to remain idle in short waiting time.

Table 2. Comparative Results of Three Multi-Objective Optimization Algorithms

<table>
<thead>
<tr>
<th>$n / m$</th>
<th>$c$</th>
<th>$E_{\text{setup}}/p_{\text{idle}}$</th>
<th>IACO (average RPD)</th>
<th>GRASPTETT (average RPD)</th>
<th>OACO (average RPD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20/5</td>
<td>2</td>
<td>0.46</td>
<td>0.62</td>
<td>2.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.58</td>
<td>1.50</td>
<td>2.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.31</td>
<td>2.35</td>
<td>4.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.27</td>
<td>2.45</td>
<td>3.98</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.41</td>
<td>3.57</td>
<td>5.60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.52</td>
<td>5.78</td>
<td>6.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.36</td>
<td>3.78</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.67</td>
<td>3.30</td>
<td>6.63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.33</td>
<td>6.08</td>
<td>7.47</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>0.66</td>
<td>3.27</td>
<td>4.86</td>
<td></td>
</tr>
<tr>
<td>50/5</td>
<td>2</td>
<td>1.15</td>
<td>2.60</td>
<td>2.88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.97</td>
<td>3.13</td>
<td>3.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.35</td>
<td>4.35</td>
<td>4.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.64</td>
<td>4.59</td>
<td>5.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.64</td>
<td>5.77</td>
<td>6.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.28</td>
<td>6.96</td>
<td>8.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.85</td>
<td>4.84</td>
<td>3.43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.07</td>
<td>5.42</td>
<td>7.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.44</td>
<td>6.20</td>
<td>8.80</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>0.93</td>
<td>4.87</td>
<td>5.64</td>
<td></td>
</tr>
<tr>
<td>100/10</td>
<td>2</td>
<td>2.08</td>
<td>4.75</td>
<td>3.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.36</td>
<td>6.35</td>
<td>6.64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.48</td>
<td>7.23</td>
<td>7.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.45</td>
<td>3.92</td>
<td>5.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.52</td>
<td>7.20</td>
<td>9.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.59</td>
<td>4.77</td>
<td>6.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.71</td>
<td>7.48</td>
<td>9.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.34</td>
<td>9.24</td>
<td>11.08</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>1.29</td>
<td>6.42</td>
<td>7.35</td>
<td></td>
</tr>
<tr>
<td>Mean for all</td>
<td></td>
<td>0.96</td>
<td>4.85</td>
<td>5.95</td>
<td></td>
</tr>
</tbody>
</table>

6. Conclusion

In this paper we have studied the problem of minimizing energy consumption and total tardiness on unrelated parallel machines. Due dates and release times are distinct, and the breakeven duration of each machine is different. A compromise balance has to find between machine energy consumption and total tardiness. We proposed a framework with an ant colony optimization algorithm (ACO) and ATC heuristic rule to solve this problem. Furthermore, it is a new kind of problem for minimization of machine energy consumption and total tardiness on unrelated parallel machines which need to be modeled and solve effectively.

In the computation evaluation, two approaches (GRASPTETT and OACO) for solving minimizing machine energy consumption and total tardiness on single machine are
adapted and compared with proposed IACO algorithm. The IACO algorithm outperforms than other approaches and GRASPTETT shows better than OACO in most of instances.

Acknowledgements

The research is supported by Enterprise Technology Center Foundation of Guangdong (2013389069), Guangdong Natural Science Foundation (2015A030310340).

References