Numerical Control Machining Grid-based Processing Schedule Optimization Using Forward Checking Algorithm

Huiwen Zhan¹,a, Yabo Luo¹,b, Xia Li¹,c*

¹School of Mechanical and Electronic Engineering, Wuhan University of Technology, Luoshi Road 205, Wuhan City, 430070 P.R. China;

a2244103076@qq.com, bluoyabo1973@163.com, c2510113987@qq.com

Keywords: NC machining grid; processing optimization; constraints satisfied problem; processing grid logistics

Abstract: In recent years, the development of grid technology provides the possibility to sharing machining resources over Internet. However, the current research on grid scheduling optimization are based on the certain machining routes regardless of the complex associated constraints, while which is one of the main features in NC (NC: Numerical Control) machining grid, including location constraints, parallel process constraints, and so on. To solve the NC machining grid optimization problem with complex associated constraints, from the viewpoint of "Constraint Satisfaction Problem (CSP)", this research established a CSP-based optimization model, and proposed the solving methods based on the forward checking algorithm based on derivation of a series of mathematical formulas. Comparative experiments demonstrated the feasibility and efficiency of the proposed model and methods.

1. Introduction

The service mode of "multi-task to multi-resource" is able to effectively improve the machining efficiency by both integrating machining capacity and machining tasks, which makes total objective value optimal [1-2]. In recent years, the development of grid technology [3-4] provides the possibility to realize "multi-task to multi-resources" mode. However, NC machining grid exists complex associated constraints, which makes scheduling optimization become a difficult problem [5-6]. This paper studies modeling and processing schedule optimization algorithm from the viewpoint of "Constraint Satisfaction Problem (CSP)", and established a CSP-based optimization model solved by the forward checking algorithm, which is proved feasible and efficient by comparative experiments.

2. Symbols Definitions and Variables Descriptions

- \( n \) —— number of work-pieces;
- \( J \) —— set of work-pieces, \( J = \{J_1, J_2, ..., J_n\} \);
- Attributes of work-pieces: \( \{Wno, Wjobsum, Wstarttime, Wdelivtime\} \);
- \( O_i \) —— number of processes, the processes number of i-th work-piece, \( i = 1, 2, ..., n \);
- \( J_i \) —— processes set of work-piece \( i \), \( J_i = \{J_{i1}, J_{i2}, ..., J_{iO_i}\} \), \( j = 1, 2, ..., O_i \);
- \( J_{ij} \) —— j-th process of work-piece \( i \);
- Attributes of work-pieces: \( \{Jno, JWno, Jtype, Jpreci, Jptime, Jbejobs\} \);
- \( m \) —— number of resource nodes;
- \( R_k \) —— k-th resource node;
- \( RN \) —— set of resource nodes, \( \{R_1, R_2, ..., R_m\} \);
- Attributes of resource nodes: \( \{RNno, RNtype, RNpreci, RNlocation, RNstatus\} \);
- \( R_{ij} \) —— set of all resource nodes matched to j-th process function constrain of work-piece \( i \), \( R_{ij} \in RN \);
- \( r_{ij} \) —— machining resource node selected by j-th process of work-piece \( i \), \( r_{ij} \in R_{ij} \);
- \( P_{ij} \) ——machining time of j-th process of work-piece \( i \);
$S_{ij}$ —— available start machining time set of $j$-th process of work-piece $i$, $S_{ij} = (es_{ij}, ls_{ij})$, $es_{ij}$ is the earliest start machining time of the process, $ls_{ij}$ is the latest machining time of the process;

$s_{ij}$ —— start machining time of $j$-th process of work-piece $i$, $s_{ij} \in S_{ij}$;

$f_{ij}$ —— finishing time of $j$-th process of work-piece $i$;

$G_i$ —— the set of serial process couple $[J_{il}, J_{ik}]$ of work-piece $i$, wherein the process $J_{il}$ must be machined before process $J_{ik}$;

$Q_i$ —— the set of disorder process couple $[J_{il}, J_{ik}]$, wherein process $J_{il}$ and process $J_{ik}$ can be machined in any order;

$M_{ijkl}$ —— the needed logistics time that work-piece $i$ is moved to the next resource node $l$ to be machined for next process after the $j$-th process having been finished in the resource node $k$;

$MC$ —— Logistics cost per unit time;

$TC$ —— per unit time cost due to delivery in advance;

$rd_{ii}$ —— releasing periods of the work-piece $i$, namely the earliest time that first machining process of work-piece $i$ can be machined;

$dd_{ii}$ —— delivery time of work-piece $i$, namely the latest finishing time of the last machined process of work-piece $i$.

### 3. NC Machining Grid Optimization Method Based on Forward Checking Algorithm (FCA)

Using the constraint propagation algorithm and searching algorithm interactively, FCA includes compatibility technology and Back-jumping Algorithm (BA). The BA used in FCA is same as that of recalling algorithm based on contradiction oriented; compatibility technology constraint disposal includes constraints transmission and constraints checking.

Constraints transmission includes sequence constraint transmission and resource compatibility constraint transmission:

1. Sequence constraint transmission consistency disposal method: referring to process start machining time spreads along the processes routes, namely process start time assignment result in the last searching status spread to the upstream and downstream processes within work-pieces along the processes routes, start machining time set $S_{ik}$ of each process is got reduction, and sequence constraint consistency is enhanced.

2. Resource compatibility constraint transmission consistency disposal method: aiming at difficult resource node machining capacity constraint, assigning values according to process start time in the last searching status, cut down the other scheduling process start time set competing for the same resource node with constraint transmission method, thereby reducing the occurrence probability of capacity constraint conflict in the subsequent searching process.

Meanwhile, check each unscheduled process variable domain after the constraint propagation. If the domain value is $\Phi$, illustrating scheduling processes occur conflicts, conduct back-jumping disposal, otherwise saving intermediate scheduling results;

Due to machining order among processes exists successive constrain relation in scheduling problems, which makes each process variable has certain transitivity on time constrains, according to the machining route successive relation, start time domain will transfer to next automatically. So, just check and cut down the next serial process variable of current process in compatibility checking, not necessary to check all uninitialized process variables after the current process variable.

### 4. Comparative Experiment Study

For comparing and analyzing the merits and demerits of GSA, this study chooses a medium-sized scheduling problem as the test instance, using traditional optimization algorithm FIFO (First In First Out) to solve the same problem and compare and analyze the two scheduling schemes, the selected problem is described as follows: Supposing now there are five machining types of NCM resource nodes for a total of 10, and the number of tasks needed to complete is 10. Supposing logistics cost per unit time is 3.6, and delivery cost in advance per unit time is 3.2, the total costs are required lowest.
The following will compare the effectiveness of two algorithms from two aspects of solving ability and algorithm efficiency including iterations and solving time. Solving ability numerical comparison of two algorithms is shown as Table 1.

Table 1. Solving ability numerical comparison.

<table>
<thead>
<tr>
<th>Feasible solution number (fsn)</th>
<th>GSA Dissimilar Feasible solution number</th>
<th>Optimal target value</th>
<th>FIFO Dissimilar Feasible solution number</th>
<th>Optimal target value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>1172.00</td>
<td>10</td>
<td>1796.40</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1105.60</td>
<td>20</td>
<td>1736.00</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>1019.60</td>
<td>29</td>
<td>1721.60</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>1002.80</td>
<td>39</td>
<td>1712.40</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>1086.00</td>
<td>48</td>
<td>1709.60</td>
</tr>
<tr>
<td>100</td>
<td>99</td>
<td>1030.40</td>
<td>95</td>
<td>1686.00</td>
</tr>
<tr>
<td>200</td>
<td>195</td>
<td>926.00</td>
<td>180</td>
<td>1654.80</td>
</tr>
<tr>
<td>500</td>
<td>480</td>
<td>901.60</td>
<td>420</td>
<td>1676.40</td>
</tr>
<tr>
<td>1000</td>
<td>876</td>
<td>934.80</td>
<td>678</td>
<td>1631.00</td>
</tr>
</tbody>
</table>

As can be seen from Table 1, when feasible solution number (fsn≤50) is smaller, dissimilar feasible solutions number both they can find are almost the same as feasible solution number. But with feasible solution number (fsn>50) increasing, the difference of dissimilar feasible solutions number both they can find is more obvious. When fsn=500, GSA can find 480 different solutions, while FIFO algorithm can only find 420 different solutions. It can be seen in the term of finding dissimilar feasible solution proportion, for the ability to find solutions GSA is better than FIFO algorithm.

In the term of optimal target value, GSA’s advantages are more obvious. The optimal target value 1654.80 obtained by FIFO algorithm in its all solutions is higher 42 percent (the target function required minimum) than 1172.00 obtained by GSA solving 10 times randomly. After 500 times, the optimal target value 901.6 obtained by GSA is much better than the scheduling results of FIFO algorithm, by nearly 84%. And in the solving process, GSA can always jump out of local optimal, with global searching ability. Solving efficiency comparison is shown as Table 2.

Table 2. Solving efficiency comparison.

<table>
<thead>
<tr>
<th>Target value range (tvr)</th>
<th>GSA iterations</th>
<th>Solving time</th>
<th>Target value</th>
<th>FIFO iterations</th>
<th>Solving time</th>
<th>Target value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1750</td>
<td>1</td>
<td>344</td>
<td>1654.80</td>
<td>13</td>
<td>1732.80</td>
<td></td>
</tr>
<tr>
<td>&lt;1725</td>
<td>1</td>
<td>344</td>
<td>1654.80</td>
<td>138</td>
<td>1722.00</td>
<td></td>
</tr>
<tr>
<td>&lt;1700</td>
<td>1</td>
<td>344</td>
<td>1654.80</td>
<td>353</td>
<td>1699.20</td>
<td></td>
</tr>
<tr>
<td>&lt;1675</td>
<td>1</td>
<td>344</td>
<td>1654.80</td>
<td>471</td>
<td>17781</td>
<td>1672.80</td>
</tr>
<tr>
<td>&lt;1650</td>
<td>16</td>
<td>593</td>
<td>1476.80</td>
<td>552</td>
<td>22734</td>
<td>1640.00</td>
</tr>
<tr>
<td>&lt;1625</td>
<td>16</td>
<td>593</td>
<td>1476.80</td>
<td>∞</td>
<td>∞</td>
<td>1631.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>∞</td>
<td>∞</td>
<td>1631.00</td>
</tr>
<tr>
<td>&lt;1100</td>
<td>58</td>
<td>3012</td>
<td>1019.60</td>
<td>∞</td>
<td>∞</td>
<td>1631.00</td>
</tr>
<tr>
<td>&lt;1000</td>
<td>382</td>
<td>19453</td>
<td>926.00</td>
<td>∞</td>
<td>∞</td>
<td>1631.00</td>
</tr>
<tr>
<td>&lt;925</td>
<td>956</td>
<td>108644</td>
<td>901.60</td>
<td>∞</td>
<td>∞</td>
<td>1631.00</td>
</tr>
<tr>
<td>&lt;900</td>
<td>∞</td>
<td>∞</td>
<td>901.60</td>
<td>∞</td>
<td>∞</td>
<td>1631.00</td>
</tr>
</tbody>
</table>
As can be seen from Table 2, When GSA iterates once and spends 344 system time, its target values can reach 1654.80. But also reaching a similar target value, FIFO algorithm need iterate 552 times and spend 22734 system time. And when FIFO algorithm can only reach optimal target value 1631.00 spending time $\infty$, GSA can easily obtain the scheduling scheme that optimal target value is 1476.80, while its optimal target value can reaches 901.60 that is much smaller than FIFO algorithm. Obviously, in term of solving efficiency, GSA is much higher than FIFO algorithm.

In terms of problem solving scale, GSA is far superior to FIFO algorithm, from the small-scale scheduling problems only a few jobs, to large-scale problems with hundreds of jobs, GSA all can complete scheduling demands, but FIFO algorithm cannot.

Of course, GSA also exist some problems, as can be seen from the scheduling result comparison table, since GSA adopts random selection strategy, at the time of improving global search, but led to the poor algorithm solving stability, large fluctuation, and the slow algorithm convergence.

5. Conclusions

NC machining grid technology provides a new possibility to share NC machining resource over Internet. To solve the complex associated constraint problem involved in the NC machining grid, after derivation of a series of mathematical formulas, this research established a CSP-based optimization model, and proposed the solving methods based on the forward checking algorithm. Comparative experiments demonstrated that the proposed model and methods have a good ability to find the global optimal solution because of the strong ability to jump out of local optimal area, and also a higher solving efficiency, therefore, which is feasible and efficient.

Acknowledgments

This research is supported by the National Natural Science Funds of China (NSFC. 51375357) and the teaching research project of WHUT (NO.2013028).

References