DEVELOPMENT OF MULTI-OBJECTIVE GENETIC ALGORITHMS FOR SCHEDULING

Pei-Chann Chang *, Jih-Chang Hsieh ** and Yen-Wen Wang *

* Department of Industrial Engineering and Management, Yuan-Ze University,
135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C.

** Department of Industrial Management, Vanung University, Taoyuan, Taiwan, R.O.C.

iepchang@saturn.yzu.edu.tw

ABSTRACT

Scheduling in the drilling operation of the printed circuit board industry deeply annoys the production management staff on the shop floor due to its high complexity of permutations or combination of jobs, machines, and resource constraints. In this research, two multi-objective genetic algorithms are proposed to deal with such a complicated real-world case. Real-world instances are applied as well to evaluate the proposed algorithms. The result indicates that both VMOGA and AMOGA are effective.

Keywords: Multi-Objective Genetic Algorithm, Scheduling, Printed Circuit Board

1. BACKGROUND

Production scheduling is the arrangement of jobs to be processed on available machines under some constraints. In this article, we are concerned with the scheduling problems in a real-world printed circuit board (PCB) factory. Two objectives, i.e. makespan and total tardiness time are considered to meet the requirements from the shop floor. Genetic algorithms are applied to deal with the optimization problems. Combining multiple objectives and genetic algorithms, multi-objective genetic algorithms (MOGA) is developed for scheduling in the PCB factory.

the conflict among the contending jobs in the Giffler and Thompson (GT) procedure applied for job shop scheduling problems. Mansouri et al. (2003) dealt with exceptional elements in cellular manufacturing by applying MOGA.

Although these studies have provided valuable developments and applications for MOGA, some improvements in designing MOGA still have not yet been made. This paper proposes two MOGAs. A real-world case of scheduling in a printed circuit board (PCB) factory is introduced to be an application. The results found in this research can be appealing to the PCB industry.

2. PARETO OPTIMAL SOLUTIONS AND PERFORMANCE MEASURES

In this research, two objectives are considered. One is makespan and the other is total tardiness time. The aim of the proposed methods is to search minimal makespan and total tardiness time. The definitions of makespan and total tardiness time are given as follows:

1. makespan (\(C_{\text{max}}\)): the length of time required to finish processing all jobs, i.e. \(C_{\text{max}} = \max \{C_1, C_2, \ldots, C_k\}\). Where \(C_j\) denotes the makespan on machine \(j\), \(\forall j = 1, 2, \ldots, k\).

2. total tardiness (\(\sum T_i\)): the sum of tardiness of each job and the tardiness of each job (\(T_i\)) equals to \(\max \{C_i - d_i, 0\}\). Where \(d_i\) denotes the due-date of job \(i\).

The common method to simultaneously consider multiple objectives is to combine these objectives linearly with fixed weights. Linear combinations actually transform multiple objectives into a single objective. Such combinations cause the loss of diversity in potential solutions. To overcome this shortcoming, Pareto optimal solutions are applied to retain the diversity. The definition of Pareto optimal solutions can be referred to Tamaki et al. (1996).

**Definition: Pareto Optimal Solutions**

Let \(x_0, x_1, x_2 \in F\), and \(F\) is a feasible region. And \(x_0\) is called the Pareto optimal solution in the minimization problem if the following conditions are satisfied.

1. If \(f(x_i)\) is said to be partially greater than \(f(x_2)\), i.e. \(f_i(x_1) \geq f_i(x_2), \forall i = 1, 2, \ldots, n\) and \(f_i(x_1) > f_i(x_2), \exists i = 1, 2, \ldots, n\), Then \(x_1\) is said to be dominated by \(x_2\).

2. If there is no \(x \in F\) s.t. \(x\) dominates \(x_0\), then \(x_0\) is the Pareto optimal solutions.

The geometric interpretation of Pareto optimal solutions for a bi-objective problem is demonstrated in Figure 1.
Two MOGAs are proposed in this paper. Both methods are applied to search Pareto optimal solutions for the scheduling problems. Criteria proposed by Ishibuchi and Murata (1998) are introduced to evaluate the proposed methods and listed as follows:

1. The number of Pareto optimal solutions searched by each approach ($A$).
2. The number of non-dominated solutions of each approach after the approaches are compared to each other ($B$).
3. Percent of non-dominated solutions ($\frac{B}{A} \times \%$).

The first criterion represents the capability of each method that how many Pareto optimal solutions can be searched. The one with more Pareto optimal solutions is the better method. However, adding the consideration of solution quality, we have to count the number of non-dominated solutions. Aggregately speaking, for considering both searching capability and solution quality, the third criterion is a suitable index.

3. DEVELOPMENT OF MULTI-OBJECTIVE GENETIC ALGORITHMS

3.1 MOGA

Murata et al. (1996) proposed a MOGA and applied it to flowshop scheduling. They used weighted sum to combine multiple objectives into single objective. For example:

$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x)$$

where $f_1(x), f_2(x), \ldots, f_n(x)$ are the objective functions and $w_1, w_2, \ldots, w_n$ are the weights of corresponding objectives that satisfy the following conditions.

$$w_i \geq 0 \quad \forall i = 1, 2, \ldots, n$$

$$w_1 + w_2 + \ldots + w_n = 1$$

Once the weights are determined, the searching direction is fixed. To search Pareto optimal solutions as many as possible, the searching directions should be changed again and again to sweep over the whole solution space. Therefore the weights have to be changed again and again. The weights consist of random numbers and they are generated as the following way:

$$w_i = \frac{r_i}{r_1 + r_2 + \cdots + r_n}, \quad \forall i = 1, 2, \ldots, n$$
where $r_1, r_2, \ldots, r_n$ are random numbers within $(0, 1)$.

Solutions searched through changing directions are collected in a set. Then the definition of Pareto optimal solution is applied to determine which solutions in the set are Pareto optimal. The step repeats in every generation in MOGA. The complete MOGA algorithm is introduced in Figure 2 and the details of each step are explained in the following.

### Step 1: Encoding
Integer coding is used in MOGAs in this research. For example a solution consists of 5-bit as 3-1-4-2-5 denotes that job 3 is process first, and then job 1, job 4, job 2, job 5 are processed successively.

### Step 2: Generate Initial Population
Initial solutions are randomly generated and these initial solutions form the first population.

### Step 3: Record Pareto Optimal Solutions
Calculate the objective values of chromosomes in the population and record the Pareto optimal solutions.

### Step 4: Calculate Objective Value
The total objective function is constituted of the linear combination of objective functions. And the weights are randomly assigned. For a solution $x$, the objective function in the study is represented as follows:

$$f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x)$$

where $f_1(x) =$ makespan

$$f_2(x) =$ total tardiness

### Step 5: Evaluate Fitness
The original concept of fitness is the larger the better because solutions with larger fitness tend to propagate to the next generation. In this paper, minimization of objectives is considered. Hence it contradicts the original idea of fitness. A transformation should be made to reverse the minimization to maximization. For a solution $x$, its fitness equals to the maximal objective value in the generation minus itself. The formula is listed in the following.
\[
\text{fitness}(x) = f_{\text{max}}(x) - f(x)
\]

Step 6: Reproduction / Selection
The roulette wheel selection (Goldberg (1989)) is applied in MOGAs in this paper.

Step 7: Crossover
Two-point crossover method (Goldberg (1989)) is applied in MOGAs in this paper.

Step 8: Mutation
Shift mutation (Goldberg (1989)) is applied in MOGAs in this paper.

Step 9: Elite Strategy
The elite strategy retains the top k solutions in order to keep quality solutions in each generation.

Step 10: Replacement
The new population generated by the previous steps updates the old population.

Step 11: Update Pareto Optimal Solutions
Search the Pareto optimal solutions in the new population and update the old Pareto optimal solutions with new ones.

Step 12: Stopping rule
If the number of generations equals to the pre-specified number then stop, otherwise go to step 4.

3.2 VMOGA
The first method proposed is the multi-objective genetic algorithm with variable rates (VMOGA). Huband et al. (2003) proposed similar concept in probabilistic mutation while we have both crossover and mutation rates variable. The steps of VMOGA are the same as MOGA except the crossover and mutation operators. The major difference between MOGA and VMOGA is that the crossover and mutation rates of MOGA are fixed while they are randomly selected within specified intervals in VMOGA. Random rates within specified intervals of crossover and mutation are supposed to enlarge the search space and accelerate the convergence.

3.3 AMOGA
The second method proposed is the adaptive multi-objective genetic algorithm (AMOGA). The steps of AMOGA are the same as MOGA except the way to adjust the rates of crossover and mutation operators. In AMOGA, adaptive genetic operators are incorporated into the algorithm. Each solution has to be compared with the preceding one on two objectives respectively. Let \( r_i = \frac{v_i}{v_{i-1}} \), where \( r_1 \) is the ratio by the first objective and \( r_2 \) is the ratio by the second objective. Note that \( v_x \) is the objective value of a solution \( x \) and \( v_{x-1} \) is the objective value of a solution that precedes \( x \). If \( r_1 \leq 1.001 \) and \( r_2 \leq 1.001 \) then interval rates are applied for crossover and mutation; otherwise increase the rates generation by generation.

4. REAL-WORLD APPLICATION
In this paper, the proposed MOGAs are used to solve a real-world case of scheduling. The case study material is a PCB factory in Taiwan. The production lines of top PCB factories have been highly automated except drilling operations in order to meet the requirements of large production quantity and stable quality. The scheduling of the drilling operations still needs separate treatment from the automated system.

There are 28 machines in the drilling operation station available to process jobs. Due to the combination of the jobs and machines is large, thus it creates a bottleneck in the manufacturing process. Two years ago, the managers used manual manipulation in sequencing
these jobs. However, it usually took more than 4 hours to arrange daily schedules. That wasted too much time. Current scheduling system was introduced about two years. It is a computer-aided system and the time to arrange the daily schedules has dramatically cut down. The times elapsed in sequencing are within minutes. Although the improvement has been done, it is convincing that there are still more advanced skills that can be incorporated because the current scheduling system simply uses dispatching rules plus local search. Therefore MOGA are further developed to be more effective and efficient for scheduling the drilling operation.

5. NUMERICAL EXPERIMENT

In order to verify the effectiveness and efficiency for these methods, real-world cases are used. The processing time and due-date of each job were directly retrieved from the database of the PCB factory.

Notations used in this experiment are explained in the following:

\[ n \] : number of jobs
\[ m \] : number of machines
\[ N_{pop} \] : population size
\[ G_{max} \] : number of evolving generations
\[ P_c \] : crossover rate
\[ P_m \] : mutation rate
\[ N_{elite} \] : number of elites

A statistical analysis, design of experiment, was conducted to determine the optimal levels for each parameter and strategy. The result is reported in Table 1-3, respectively. In every instance, 20 runs were executed for accumulating Pareto optimal solutions and taking averages.

**Table 1. Rates for MOGA**

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>((N_{pop}, G_{max}))</th>
<th>(P_c)</th>
<th>(P_m)</th>
<th>(N_{elite})</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10</td>
<td>(40, 3750)</td>
<td>0.7</td>
<td>0.3</td>
<td>9</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>(30, 5000)</td>
<td>0.7</td>
<td>0.3</td>
<td>5</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>(30, 6000)</td>
<td>0.6</td>
<td>0.4</td>
<td>7</td>
</tr>
</tbody>
</table>

**Table 2. Intervals for VMOGA**

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>((N_{pop}, G_{max}))</th>
<th>(P_c)</th>
<th>(P_m)</th>
<th>(N_{elite})</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10</td>
<td>(40, 3750)</td>
<td>0.65-0.75</td>
<td>0.25-0.35</td>
<td>9</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>(30, 5000)</td>
<td>0.65-0.75</td>
<td>0.25-0.35</td>
<td>5</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>(30, 6000)</td>
<td>0.50-0.70</td>
<td>0.30-0.50</td>
<td>7</td>
</tr>
</tbody>
</table>
Table 3. Strategies for AMOGA

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>((N_{pop}, G_{max}))</th>
<th>Strategies</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10</td>
<td>(40, 3750)</td>
<td>0.50-0.70</td>
<td>0.30-0.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.70+0.00005*G</td>
<td>0.40+0.000026*G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>otherwise</td>
<td>9</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>(30, 5000)</td>
<td>0.65-0.75</td>
<td>0.25-0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.75+0.00004*G</td>
<td>0.35+0.00004*G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>otherwise</td>
<td>5</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>(30, 6000)</td>
<td>0.50-0.70</td>
<td>0.30-0.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.70+0.000042*G</td>
<td>0.5+0.000016*G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>otherwise</td>
<td>7</td>
</tr>
</tbody>
</table>

With these rates, intervals, and strategies, MOGA, VMOGA, and AMOGA are applied to solving real-world scheduling cases. MOGA is regarded as a benchmark method. VMOGA and AMOGA are respectively compared with MOGA for evaluating effectiveness. The numerical results are reported in the following:

5.1 VMOGA v.s. MOGA

Table 4 reports that the comparison of MOGA and VMOGA. In two out of three cases, the solution quality of VMOGA outnumbers MOGA.

Table 4. Comparison of MOGA v.s. VMOGA

<table>
<thead>
<tr>
<th>Methods</th>
<th>n</th>
<th>m</th>
<th>A</th>
<th>B</th>
<th>B/A(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOGA</td>
<td>30</td>
<td>10</td>
<td>45</td>
<td>24</td>
<td>53.33</td>
</tr>
<tr>
<td>VMOGA</td>
<td></td>
<td></td>
<td>35</td>
<td>26</td>
<td>74.29</td>
</tr>
<tr>
<td>MOGA</td>
<td>40</td>
<td>15</td>
<td>85</td>
<td>51</td>
<td>60.00</td>
</tr>
<tr>
<td>VMOGA</td>
<td></td>
<td></td>
<td>95</td>
<td>55</td>
<td>57.89</td>
</tr>
<tr>
<td>MOGA</td>
<td>50</td>
<td>20</td>
<td>116</td>
<td>38</td>
<td>32.76</td>
</tr>
<tr>
<td>VMOGA</td>
<td></td>
<td></td>
<td>94</td>
<td>44</td>
<td>46.81</td>
</tr>
</tbody>
</table>

5.2 AMOGA v.s. MOGA

The numerical result of MOGA and AMOGA is shown in Table 5. On average, the performance in terms of B/A(%) of AMOGA is more effective.
Table 5. Comparison of MOGA v.s. AMOGA

<table>
<thead>
<tr>
<th>Methods</th>
<th>n</th>
<th>m</th>
<th>A</th>
<th>B</th>
<th>B/A(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOGA</td>
<td>30</td>
<td>10</td>
<td>52</td>
<td>38</td>
<td>73.08</td>
</tr>
<tr>
<td>AMOGA</td>
<td>36</td>
<td>28</td>
<td>77.78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOGA</td>
<td>40</td>
<td>15</td>
<td>62</td>
<td>36</td>
<td>58.06</td>
</tr>
<tr>
<td>AMOGA</td>
<td>82</td>
<td>54</td>
<td>68.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOGA</td>
<td>50</td>
<td>20</td>
<td>82</td>
<td>33</td>
<td>26.83</td>
</tr>
<tr>
<td>AMOGA</td>
<td>129</td>
<td>46</td>
<td>35.66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3 Computation time

Time elapsed in computation represents the efficiency. Table 6 reports the computation times of MOGA, VMOGA, and AMOGA in the numerical experiments. The computation times of the three MOGAs are nearly. It means all the MOGAs in this research have almost same efficiency.

Table 6. Average computation times of MOGA, VMOGA, and AMOGA

<table>
<thead>
<tr>
<th>n</th>
<th>M</th>
<th>Average Computation Times (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MOGA</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>149.71</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>184.62</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>308.39</td>
</tr>
</tbody>
</table>

In the above comparisons, VMOGA and AMOGA lead MOGA in solution quality in the test instances. The performance of efficiency of MOGA, VMOGA, and AMOGA has no significant difference. Actually, the computation times are acceptable in the shop floor. It is promising that VMOGA and AMOGA are sufficient with potential in solving the real-world cases.

6. CONCLUSION

In this paper, two methods based on MOGA, i.e. VMOGA and AMOGA, were proposed for scheduling to the drilling operation in printed circuit board industry. Real-world cases were applied to verify the effectiveness of the proposed methods. The results indicate that VMOGA and AMOGA are superior to MOGA in solution quality. All the MOGAs in this research are nearly in computation times. The result reveals that the proposed methods are potential for the practical applications.

REFERENCES


