EVOLUTIONARY STRATEGY FOR MANUFACTURING CELL DESIGN

Abstract

Group Technology (GT) is philosophy, which aims to decompose a manufacturing system into autonomous subsystems (groups). The objective is to aggregating similar parts into families and dissimilar machines into cells such that inter-cell movement of parts is minimized. From the works by E. Falkenauer it appears that a standard scheme and elements of an evolutionary algorithm are not suitable for the problem of grouping the elements. These observations are confirmed by the other researches. As a remedy E. Falkenauer proposed a new encoding scheme and genetic operators adapted to the grouping problem, yielding Grouping Genetic Algorithm (GGA).

In this paper we investigate the use of not specialized evolutionary strategy for manufacturing cell design. We used $(1, \lambda)$-ES, where 30 children are generated from one parent by means of the simple mutations; the crossover is not applied. The best of the descendants becomes the new parent solution. The experiments shown a great usefulness of the evolutionary strategy for the cell design problem. The results confirmed once more the power of the evolutionary algorithms, which consists in ability to generate very good solutions without going into the structure of the problem.

1. INTRODUCTION

Group Technology (GT) is philosophy, which aims to decompose a manufacturing system into autonomous subsystems (groups). The objective is to aggregating similar parts into families and dissimilar machines into cells such that intercell movement of parts is minimized [6].

In this paper, we first discuss the problem of grouping parts into families and machines into cells (group technology problem). Section 3 explains the evolution strategy (ES) approach used to the cell formation problem. In Section 4, we present computational results on published problems, using grouping efficacy as the evaluation function. The conclusions are drawn in Section 5.

2. THE CELL DESIGN PROBLEM
Creation of a Cellular Manufacturing is concerned with the development of efficient techniques for group formation. A widely known method is to group the parts on the basis of the PFA (Process Flow Analysis). On the basis of the analysis of the existing manufacturing process routings, a machine/part incidence matrix $M$ is constructed. A value of 1 for the element $M_{ij}$ means that machine $i$ is used to perform part $j$, a value of 0 - a contrary situation. Therefore the $M_j$ column of the matrix describes the technologic routing for part $j$.

The goal is to transform $M$ into clusters by rearranging rows and columns. The resultant blocks form the cells [7].

Many algorithms have been proposed to solve the manufacturing cell problem, one can find a comprehensive review in [5]. Our approach belongs to artificial intelligence techniques.

3. AN EVOLUTIONARY ALGORITHM FOR GROUPING THE PARTS

From the works by E. Falkenauer [3] it appears that a standard scheme and elements of a genetic algorithm are not suitable for the problem of grouping the products. As a remedy E. Falkenauer proposed a new encoding scheme and genetic operators adapted to the grouping problem, yielding Grouping Genetic Algorithm (GGA). In our opinion there is no need to apply specialized operators and representations for the grouping problems: even simple evolutionary algorithm performs as well as specialized and/or hybridized algorithms.

The version of the evolutionary strategy used by the author takes the algorithm for the bin packing problem [10] as its model; we have not tried to optimize any parameters of the algorithm. It is $(1,\lambda)$ - ES [9] where 30 children are generated from one parent by means of the simple mutations; the crossover of solutions is not employed. The best of the descendants becomes the new parent solution. This manner of selection (the parent do not compete with the descendants) often causes deterioration of the parent solution but it improves the efficiency of the algorithm. The action of the algorithm is terminated after generating only 1 000 generations. It means that only 30 000 evaluations of single solution is made during algorithm running, irrespective of problem size.

It should be emphasized that the algorithm explicitly assigns machines and parts to cells during evaluation of the fitness function so determining the bottleneck machines is unequivocal.

A general structure of the used evolutionary strategy recorded in PASCAL pseudo code is shown below:

BEGIN
    generate and evaluate random starting solution;
    best solution:= start solution;
    REPEAT
        FOR $i=1$ to $\lambda$ DO
            BEGIN

BEGIN
copy parent to create child number \( i \);
mutate child \( i \);
END;
evaluate children;
choose best child based on objective function to be a new parent;
UNTIL (max. generation);
output the best solution;
END.

A widely known ordinal representation modified for the purposes of the grouping the parts was used in the algorithm. The solution is represented by a list of \( n \) parts and \( k \) separators of groups; the value \( j \) (1 \( \leq j \leq n \)) determining the part number can appear in the list just once, just as the value \( i \) (n+1 \( \leq i \leq n+k \)) determining the number of separators.

The decoding and encoding mechanisms are very simple. For example: for 7 products and 3 separators the solution \( S_1 = (1,3,9,8,5,2,7,10,6,4) \) means that the parts are divided into three groups (1,3), (5,2,7) and (6,4) when the solution \( S_2 = (1,10,3,8,5,2,9,7,6,4) \) means that the parts are divided into four groups (1), (3), (5,2) and (7,6,4).

By setting \( k \) to the value required by designer, we can control the number of separating groups in the solutions.

A weighted group efficacy \( \gamma \) developed by Ng [8] was used as a performance index (fitness function - FF).

\[
 FF = \gamma = \frac{q(e-e_0)}{q(e+e_v-e_0)+(1-q)e_0} \tag{1}
\]

where:
- \( e \)-the number of operations in the data matrix
- \( e_v \)-the number of voids in the diagonal blocks
- \( e_0 \)-the number of exceptional elements
- \( q \)-the weight associated with the voids in the diagonal blocks
The default value of \( q \) was assumed as 0.5.

Mutations known from the literature such as exchange, insertion and inversion were applied in the algorithm. The only difference was the fact that the moves within the groups were forbidden. An essential assumption that makes the algorithm powerful is the fact that the parts as well as the separators were subject to the operations.

4. EXPERIMENTAL RESEARCHES

ES algorithm was tested on 17 common data sets found in the literature to study its performance as a clustering tool. The ES parameters used in these experiments are defined in Table 1. The maximum number of permissible clusters \( k_{max} \) was set equal to the best
known number of cells $kL$, as found in literature. Each run of the algorithm was replicated 50 times and parameter $Gen$ - the average number of generations needed to reach the best solution - was calculated.

Table 1 summarizes the experimentation results. In all cases, the evolutionary algorithm determined the solutions with the same number of cells and a grouping efficacy measure equal to any previously reported results. As a benchmark we used the integer-based GA introduced by J. Joines, at al. [4] because it is the best evolutionary heuristic known from literature. This algorithm uses 7 different genetic operators (4 mutations and 3 crossovers) applied to population of 50-80 elements with the maximum number of generations 5 000-20 000 depending on the size of problem. Following Folkenauer lesson the authors use two problem-specific crossover operators to improve the speed and quality of GA performance.

Table 1. Summary of experimental results

<table>
<thead>
<tr>
<th>No.</th>
<th>Reference</th>
<th>Matrix</th>
<th>$e$</th>
<th>$kL$</th>
<th>$\Gamma$</th>
<th>$e\theta$</th>
<th>$e\nu$</th>
<th>GenGA</th>
<th>GenES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chandrasekharan and Rajagopolan</td>
<td>8*20</td>
<td>61</td>
<td>3</td>
<td>0.8525</td>
<td>9</td>
<td>0</td>
<td>119.9</td>
<td>6.7</td>
</tr>
<tr>
<td>2</td>
<td>Simple Chan and Miller</td>
<td>10*15</td>
<td>46</td>
<td>3</td>
<td>0.9200</td>
<td>0</td>
<td>4</td>
<td>97.0</td>
<td>6.4</td>
</tr>
<tr>
<td>3</td>
<td>Complex Chan and Miller</td>
<td>10*15</td>
<td>49</td>
<td>3</td>
<td>0.8000</td>
<td>5</td>
<td>6</td>
<td>97.9</td>
<td>7.6</td>
</tr>
<tr>
<td>4</td>
<td>Srinivasan. et al.</td>
<td>10*20</td>
<td>40</td>
<td>4</td>
<td>0.8163</td>
<td>0</td>
<td>9</td>
<td>262.1</td>
<td>11.6</td>
</tr>
<tr>
<td>5</td>
<td>Seifoddini</td>
<td>11*22</td>
<td>78</td>
<td>3</td>
<td>0.7312</td>
<td>10</td>
<td>15</td>
<td>212.8</td>
<td>14.3</td>
</tr>
<tr>
<td>6</td>
<td>Askin and Subramanian</td>
<td>14*23</td>
<td>58</td>
<td>4</td>
<td>0.6951</td>
<td>2</td>
<td>28</td>
<td>560.1</td>
<td>40.1</td>
</tr>
<tr>
<td>7</td>
<td>Stanfel</td>
<td>14*24</td>
<td>61</td>
<td>5</td>
<td>0.7051</td>
<td>6</td>
<td>17</td>
<td>895.0</td>
<td>11.9</td>
</tr>
<tr>
<td>8</td>
<td>Srinivasan. et al.</td>
<td>16*30</td>
<td>116</td>
<td>4</td>
<td>0.6783</td>
<td>19</td>
<td>27</td>
<td>432.0</td>
<td>45.1</td>
</tr>
<tr>
<td>9</td>
<td>King</td>
<td>16*43</td>
<td>126</td>
<td>4</td>
<td>0.4926</td>
<td>26</td>
<td>77</td>
<td>1134.7</td>
<td>6.7</td>
</tr>
<tr>
<td>10</td>
<td>Simple Joines</td>
<td>20*25</td>
<td>86</td>
<td>5</td>
<td>0.8600</td>
<td>0</td>
<td>14</td>
<td>916.1</td>
<td>26.8</td>
</tr>
<tr>
<td>11</td>
<td>Complex Joines</td>
<td>20*25</td>
<td>97</td>
<td>5</td>
<td>0.7748</td>
<td>11</td>
<td>14</td>
<td>800.0</td>
<td>17.6</td>
</tr>
<tr>
<td>12</td>
<td>Carrie</td>
<td>24*18</td>
<td>88</td>
<td>7</td>
<td>0.5701</td>
<td>27</td>
<td>19</td>
<td>986.3</td>
<td>4.3</td>
</tr>
<tr>
<td>13</td>
<td>Burbidge</td>
<td>20*35</td>
<td>136</td>
<td>4</td>
<td>0.7571</td>
<td>2</td>
<td>41</td>
<td>680.4</td>
<td>22.1</td>
</tr>
<tr>
<td>14</td>
<td>Chandrasekharan and Rajagopolan</td>
<td>24*40</td>
<td>105</td>
<td>7</td>
<td>0.8015</td>
<td>0</td>
<td>26</td>
<td>1637.7</td>
<td>90.4</td>
</tr>
<tr>
<td>15</td>
<td>Chandrasekharan and Rajagopolan</td>
<td>24*40</td>
<td>130</td>
<td>7</td>
<td>0.8510</td>
<td>11</td>
<td>11</td>
<td>2192.0</td>
<td>132.0</td>
</tr>
</tbody>
</table>
The benchmark problems we found as easy for our algorithm: in any case ES did not needed more than 200 generations to find the solution. Results obtained by ES are better than those obtained by GA taking into account parameter $Gen$. while grouping efficacy $\Gamma$ were the same for both algorithms. The superiority of ES over GA significantly increases for more complex instances. For example for 100*40 matrix taken from Chandrasekharan and Rajagopalan [1] ES found the best solution in 161.3 generations. whereas GA found it in 5 682.3 generations (on average). Note that the ES deals with smaller population than GA and the operators are less complicated so we can assume that the algorithm is much faster than GA.

5. FINAL CONCLUSIONS

The researches shown a great usefulness of the evolutionary strategy for the problem of grouping parts into part families. The results of the experiments confirmed once more the power of the evolutionary algorithms, which consists in ability to generate very good solutions without going into the structure of the problem. The simplicity and high elasticity of the evolutionary algorithm is worthy of laying emphasis on: changing only the form of fitness function it is possible to obtain solutions required in a definite manufacturing situation. It is noteworthy that during the calculations many admissible solutions were obtained. They differed in the number of groups, the synthetic index of matching and also in the indexes of the similarity in the groups. In one case the algorithm tested 30 000 solutions what made only an insignificant part of the search space.

We intend to test the algorithm on problems of greater sizes in our further works.

REFERENCES


