Optimization of Inter Cellular Movement of Parts in Cellular Manufacturing System Using Genetic Algorithm

Siva Prasad Darla, C.D. Naiju, Polu Vidya Sagar and B. Venkat Likhit
SMBS, VIT University, Vellore, Tamil Nadu-632014, India

Abstract: In the modern manufacturing environment, Cellular Manufacturing Systems (CMS) have gained greater importance in job shop or batch-type production to gain economic advantage similar to those of mass production. Successful implementation of CMS highly depends on the determination of part families; machine cells and minimizing inter cellular movement. This study considers machine component grouping problems namely inter-cellular movement and cell load variation by developing a mathematical model and optimizing the solution using Genetic Algorithm to arrive at a cell formation to minimize the inter-cellular movement and cell load variation. The results are presented with a numerical example.

Keywords: Cell formation, cellular manufacturing, genetic algorithm, optimization

INTRODUCTION

Modern manufacturing environment is characterized by competitive market, larger product variety, fast changing customer needs and shorter product life cycles. This leads to efficient resource utilization and attempts to preempt a competitive response. This brings into the picture the technique of Cellular Manufacturing Systems (CMS) which has received considerable attention among researchers in providing a cell formation method such that the resources are utilized to the maximum extent with minimization of cost, maximizing the output and to over all control of the production process.

Cellular manufacturing systems acts as a compromise between flow line production and job shop production. These are the result of direct application of Group Technology (GT) philosophy. In GT, parts that have similar processing requirements such as machines, tools, routes and geometrical shapes are classified into part families (Javadian et al., 2010).

Machine cells contain groups of functionally dissimilar machine types such that each machine cell processes one or more part families. Reorganization of functional shop processes leads to several benefits such as reduction in production costs, setup and throughput times, work-in process inventories, material handling as well as improvements in machine utilization.

A bi-criteria mathematical model is proposed for a machine cell formation problem and solution approach using Genetic algorithm (Venugopal and Narendran, 1992). A simultaneous solution of the machine grouping and cell layout problem is modeled using simulated annealing technique (Attahiru et al., 1992). A machine cell component problem is modeled and solution approach with a genetic algorithm (Onwubolu and Mutingi, 2001). A competitive study of simulated annealing; genetic algorithm and Tabu search is made for designing CMS (Ahemed et al., 2004).

A methodology for forming manufacturing cells is developed (Genwook, 2006) using a new similarity coefficient and solution using Genetic algorithm. A mathematical model is considered to identify machine groups and part families using an efficient simulated annealing (Abdelmole and Taboun, 1999). A comparative study of Tabu search procedures is made (Jeffery, 2005) for the cell formation problem with four other procedures that generate heuristic solutions. A mathematical model is developed (Poornachandra and Vira, 2005) and used Kusiak’s method, in design of CMS with assembly considerations, for finding out similarity coefficient. Recently, an optimization algorithm for cell load variation was developed (Hossein and Tang, 2013). A hybrid algorithm for design of incremental cellular manufacturing systems was developed using Genetic algorithm and neural network (Rezaeian et al., 2011).

Inter cellular movement value corresponds to the total number of inter-cellular moves of all the parts in a particular machine-cell cluster or machine cell arrangement. Inter-cell movement is one of the important variables for cellular manufacturing design. The value of the inter-cell movement changes with machine-arrangement. Optimization of inter-cell movements leads to minimizing the transportation cost between cells, which results in reducing manufacturing cost. Cell load variation is also another important criterion in design of CMS. Minimization of cell load is
ideal where balancing work load with in cell is the
design criteria or where there is restriction on the
available working time period.

Genetic Algorithm (GA) is heuristic technique
used to find best solutions to optimization and search
problems. These are a part of evolutionary algorithms
that use techniques like inheritance, mutation, selection
and cross over. This algorithm is an iterative process
which ends once the termination condition is satisfied.
Here in this study, a cell formation problem is solved
using bi-criteria mathematical model subjected to
constraints and the results are optimized using genetic
algorithm technique to obtain a cell configuration such
that inter-cellular movements and cello load variations
are minimized. The Genetic algorithm codes for given
problem statement is written and simulated in
MATLAB 7 and results are explained with a case study.
The system requirements to simulate algorithm are
windows operating system, minimum of 256MB RAM
and 2000MB free space on hard disk.

PROBLEM FORMULATION

In this study, a machine component grouping
problem with two main objectives is considered .A
fifteen parts ten machines problem is considered to find
out the best possible cellular grouping of machines and
parts to minimize cellular movement. The objectives
considered are minimization of inter cell movements
and minimization of cell load variations.

Indices and parameters:

- \( r \) = Part of the product 1, 2, 3…n
- \( n \) = Total number of parts
- \( p \) = Machine cells 1, 2,...k
- \( k \) = Total number of cells
- \( m \) = Total number of machines
- \( i \) = Machine index 1, 2,...m
- \( t_{ir} \) = Processing time (hour/piece) of component \( r \) on
  machine \( i \)
- \( T_i \) = Available time on machine \( i \) in a given period of
time
- \( N_r \) = Production requirement of component \( r \) in a
given period of time
- \( W = [w_{ir}] \) is an \( m \times n \) machine-component incidence
  matrix where
  - \( w_{ir} \) = Workload on machine \( i \) induced by part \( r \)
- \( r = (t_r \times N_r)/ T_i \)
- \( X = [x_{ip}] \) is an \( m \times k \) cell membership matrix where
  - \( x_{ip} = 1 \) if \( i \)-th machine is in cell \( p = 0 \) otherwise
- \( M = [m_{ip}] \) is a \( k \times n \) matrix of average cell load where
  - \( m_{ip} = \Sigma_{i=1}^{m} x_{ip} w_{ir} / (\Sigma_{i=1}^{m} x_{ip}) \)
- \( E = [e_{ir}] \) is an \( n \times m \) matrix where
  - \( e_{ir} = 1 \) if \( t_{ir} > 0 = 0 \) otherwise
  - \( y_{rp} = 1 \) if \( \Sigma_{i=1}^{m} (e_{ir} x_{ip}) > 0 = 0 \) otherwise

Objective function:

- Minimization of Intercellular Movement:
  \n  \[ F_1 = \sum_{r=1}^{n} N_r \left[ \sum_{p=1}^{k} (y_{rp} - 1) \right] \]

- Minimization of cell load variation:
  \n  \[ F_2 = \sum_{i=1}^{m} \sum_{p=1}^{k} x_{ip} \sum_{r=1}^{n} (w_{ir} - m_{ip})^2 \]

Constraints:

- Unique assignment of machines to cells
  \n  \[ \sum_{p=1}^{k} x_{ip} = 1 \text{ for } i=1,2,3,\ldots m \]

- Number of machines in a particular cell(cell size)
  \n  \[ \sum_{i=1}^{m} x_{ip} > 2 \text{ for } p=1,2,3,\ldots k \]

IMPLEMENTED ALGORITHM

This section gives the sequence of steps followed
in generating genetic algorithm and finally with the
given input data, its performance is calculated.

Initialization and evaluation: Randomly generate
initial population \( P_1 \) by input values of population size,
number of cells, number of generations, probability of
crossover and probability of mutation. Constraints
should be looked after while generating population.

Reproduction/selection: Strings are selected
stochastically from parent population \( P_1 \) to form two
pools, \( OP_1 \) and \( OP_2 \) that is being recombined. Evaluate
the string by calculating the objective function values
\( F_1 \) and \( F_2 \) for parent population \( P_1 \).

Crossover/recombination: Two candidates for
crossover are selected randomly one from pool \( OP_1 \) and
other from pool \( OP_2 \) and multiple crossover operators
is applied to selected strings with a probability of
crossover given in the input values. Also apply
mutation to sub-sequent strings with a probability of
mutation. Now, test for constraints for generated new
offspring population after crossover and mutation and
if violated, apply mutation until constraints are
satisfied. Evaluate objective function values for
offspring population and sort them in the increasing
order of their function values.

Replacement strategy: Compare the sorted strings
functional values in offspring’s with parent population
\( P_1 \) functional values and chose the best of out for next
generation. For other off springs, a random selection is
made with a particular probability.
found and part families are created and assigned to the 
out with lower objective functional values in the final 
The optimal solution would be one string that stands 
generation is less, then current population becomes old 
value of the current generation is less than the value of 
machined on particular machine, Table 3 shows total 
time required for part on machine 
below.

Table 1: Machine-part time matrix

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
<th>P13</th>
<th>P14</th>
<th>P15</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
<td>0.5</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>M2</td>
<td>0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.3</td>
<td>0.6</td>
<td>0.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>M3</td>
<td>0.6</td>
<td>0.0</td>
<td>0.7</td>
<td>0.3</td>
<td>0.0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.9</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>M4</td>
<td>0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>M5</td>
<td>0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>M6</td>
<td>0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>M7</td>
<td>0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>M8</td>
<td>0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>M9</td>
<td>0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>M10</td>
<td>0.6</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.9</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 2: Machine part index matrix

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
<th>P13</th>
<th>P14</th>
<th>P15</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Total time available on the machine

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
<th>M10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>8</td>
<td>8.6</td>
<td>8.5</td>
<td>9</td>
<td>7.9</td>
<td>7.7</td>
<td>8.3</td>
<td>8.5</td>
<td>8.7</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4: Lot size of part types being manufactured

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
<th>P13</th>
<th>P14</th>
<th>P15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nr</td>
<td>155</td>
<td>150</td>
<td>148</td>
<td>160</td>
<td>144</td>
<td>158</td>
<td>152</td>
<td>155</td>
<td>164</td>
<td>148</td>
<td>140</td>
<td>144</td>
<td>145</td>
<td>162</td>
<td>170</td>
</tr>
</tbody>
</table>

New generation: New generation is generated if the 
value of the current generation is less than the value of 
number of generation values given in the input. If 
generation is less, then current population becomes old 
and the process is repeated. Otherwise stop. 
The optimal solution would be one string that stands 
out with lower objective functional values in the final 
population.

RESULTS AND DISCUSSION

Genetic algorithm is applied to the input data given 
below and is run for 100 iterations. The final machine 
cell formation with minimum inter-cell movement was 
found and part families are created and assigned to the 
cells.

Input: Input data for the problem is shown in Table 1 
to 4. Table 1 shows time required for part on machine 
to do the machining, Table 2 shows machine part index 
matrix which implies whether or not a part gets 
machined on particular machine, Table 3 shows total 
machine time available, Table 4 shows the lot size of 
part types that needs to be manufactured. This input 
data is taken for simulating the algorithm and output is 
shown below.

Output: The following results are obtained after 100 
iterations. The string is the representation of cells. The 
string (2 2 2 1 3 1 2 3 1 2) denotes that 1st, 2nd, 3rd, 4th, 
5th, 6th, 7th machine should be in 2nd cell, 4th, 6th, 9th should be in 
1st cell and 4th, 5th, 8th machine should be in 3rd cell.

Cell 1 = (M4, M6, M9) 
Cell 2 = (M1, M2, M3, M7, M10) 
Cell 3 = (M5, M8)

Intercellular Movement values, Cell load values 
obtained and corresponding machine cell string are as 
follows:

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[1820 600 2420 2420 1665 3040 600 600 3485]</td>
<td>[3263.7 2612.8 1347.4, 1347.4 3207.2 3561.9 3114.7 2746.4 3373.6]</td>
</tr>
</tbody>
</table>

Table 5 shows the final selected 10 populations 
after 100 iterations. The results obtained shows that
intercellular movement and cell load minimum values are 600 and 1347.4 respectively for the given ten machines -fifteen parts arrangement. From the Fig. 1, it is shown that the solution of genetic algorithm converges to a low values as the number of iteration increases and hence the inter cell movement is decreased in the system.

**CONCLUSION**

Implementing cellular manufacturing is effective in grouping the clustering machines and parts according to similar operation characteristics and assigning the part families basis of inter-cell movement and cell load. Exceptional elements can also be minimized which in turn leads to minimization of transportation cost between cells and also manufacturing cost. Minimization of cell load is ideal where balancing work load with in cell design or when there is restriction on the available working time period. This method allows specifying the number of cells. With such a facility, the manufacturing system designers will be able to arrive at an optimum machine allocation to the cell. Genetic algorithm optimization technique used in this study gave better results and but there also exists several other optimization techniques which can be applied to cellular manufacturing systems. Thus implementation of cellular manufacturing systems in any organization with specific objectives gives competitive edge in the modern technology over other organizations.

**REFERENCES**


