A Genetic Algorithm for Integrated Cell Formation and Layout Decisions

Xiaodan Wu
School of Management
Hebei University of Tech.
Tianjin 300130, P.R.C. (China)

Chao-Hsien Chu
School of Info. Sciences & Tech.
Pennsylvania State University
University Park, PA 16802 USA

Yunfeng Wang
School of Management
Hebei University of Tech.
Tianjin 300130, P.R.C. (China)

Weli Yan
School of Electrical Engineering
Hebei University of Tech.
Tianjin 300130, P.R.C. (China)

Abstract - This paper presents a hierarchical genetic algorithm (GA) to solve the cell formation and layout decisions of cellular manufacturing. The intrinsic features of our proposed GA include using a hierarchical chromosome structure to encode concurrent cell design and layout decisions, developing a new selection scheme to dynamically consider two highly correlated fitness functions, and proposing a group mutation operator to increase the probability of mutation. Our tests show that these modifications are fairly effective in improving solution quality as well as shortening the speed of convergence.

I. INTRODUCTION

Global competitiveness and market demand for rapid response have driven many firms to consider innovative approaches for better design and control of manufacturing systems. Group Technology (GT) is a potential approach that can be used to enhance both flexibility and efficiency under today’s small-to-medium lot production environment. In essence, GT attempts to decompose the manufacturing system into several manageable subsystems also referred to as manufacturing cells [1].

In the design of a cellular manufacturing system (CMS), a number of decisions have to be made, including determining the required types and number of machines, determining the required types and number of material handling tools, grouping machines into cells, laying out machines within cells; and laying out cells with respect to one another [2]. Ideally all of these decisions be addressed simultaneously for an “optimal” solution [3]. However, due to the complexity and NP-complete nature [3][4] of each decision and limitation of the solution approaches most researchers have only addressed these decisions sequentially or independently [4]-[7]. That lead to a local optimization of CMS, bonding the optimal realization of systematical objectives and restraining its development and applications.

Genetic algorithms have been effectively used to solve optimization problems in cell formation [8]-[11] and facility layout [12]-[14]. In this study, a hierarchical genetic algorithm is developed to simultaneously form manufacturing cells and determine machine layout information for cellular manufacturing. In order to manage the complicated and correlated decisions, a hierarchical chromosome structure, a dynamic selection scheme, and a group mutation operator were proposed by considering heuristics from CM decisions.

II. PROBLEM DESCRIPTION

The main objective of our problem is to concurrently optimize cell formation and layout decisions, which can be formulated as follows:

Indices and Parameters:

\[ i \] Machine index; \( i=1, \ldots, m \)
\[ j \] Part index; \( j=1, \ldots, n \)
\[ k \] Cell index; \( k=1, \ldots, c \)
\[ p \] Machine position index; \( p=1, \ldots, mp \)
\[ B_j \] Transfer batch size for part \( j \)
\[ C_i \] The sum of part \( j \) transfer cost
\[ C_i^{A} \] Intracell transfer unit cost of part \( j \)
\[ C_i^{B} \] Intercell transfer unit cost of part \( j \)
\[ C_i^{C} \] Intracell backtracking unit cost of part \( j \)
\[ C_i^{D} \] Transfer cost of part \( j \) between machine \( i \) and \( i' \)
\[ D_j \] Demand quantity of part \( j \)
\[ f_{ij}^{D} \] Number of trips for part \( j \) made by material handling equipment between machine \( i \) and \( i' \)
\[ I_{C_k} \] If cell \( k \) is formed; 0, otherwise
\[ M \] An arbitrary large positive number (>10000)
\[ M_{ip} \] Machine \( i \) is assigned to position \( p \)
\[ NC \] Minimum number of cells to be formed
\[ NM \] Maximum no. of machine types allowed in each cell
\[ R_i \] Operation no. done on part \( j \) using machine \( i \)
\[ X_{ik} \] If machine \( i \) is assigned to cell \( k \); 0, otherwise
\[ Y_{ik} \] If part \( j \) is assigned to cell \( k \); 0, otherwise
\[ Z_{ip} \] If machine \( i \) is assigned to position \( p \); 0, otherwise

Minimize

\[ \sum_{i=1}^{m} \sum_{p=1}^{mp} f_{ij}^{D} C_i^{D} \] 

subject to

\[ \sum_{k=1}^{c} X_{ik} = 1, \quad i = 1, \ldots, m \] (2)
\[ \sum_{k=1}^{c} Y_{jk} = 1, \quad j = 1, \ldots, n \] (3)
\[ \sum_{i=1}^{m} X_{ik} \leq NC \cdot IC_k, \quad k = 1, \ldots, c \] (4)
A. Hierarchical Chromosome Structure

In order to encode cell formation information for parts and machines and layout information for machines, either a lengthy or a hierarchical scheme is needed. In this study, we propose to use a two-layer hierarchical scheme because there exists a highly correlational relationship between different decisions [19]. In the first layer, a direct scheme is used to encode the cell formation results for machine and then part genes. In the second layer, an indirect weighted scheme is used to represent the layout information of machine genes. The genes in the first layer control the genes of second layer in a hierarchical manner.

Assume that there are \( n \) parts and \( m \) machines to be grouped into manufacturing cells. Let \( C \) represent a chromosome in the population and \( l_1 \) and \( l_2 \) be the first and the second layers of the gene vectors respectively. Then, \( l_1 \) consists of \( (m+n) \) genes. Where allele of the first \( m \) genes \( a_i (i=1, 2, \ldots, m) \) represents the cell number to which machine \( i \) was assigned and the allele of the next \( n \) genes \( b_j (j=1, 2, \ldots, n) \) encodes the cell number to which part \( j \) was assigned [10][11]. \( l_2 \) consists of \( m \) genes and the allele of each gene represents the positional weight \( (w_j, i=1, 2, \ldots, m) \) to which the machine was assigned. These weights are then ranked to determine the layout sequence of the machines. The machine with the highest weight will be placed in the first position. The chromosome can be depicted as:

\[
C = \begin{bmatrix}
l_1 \\
l_2
\end{bmatrix} = \begin{bmatrix}
a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n \\
w_1, w_2, \ldots, w_m
\end{bmatrix}
\]

For illustration, let us consider a small data set with 10 parts and 7 machines, to be classified into 2 manufacturing cells. Table I shows a typical chromosome. As shown, the chromosome contains two layers of genes. In layer one, the chromosome contains 17 genes (i.e., \( 7+10 \)). The allele of each gene represents the cell number to which the machine or part belongs. For instance, \( m1 \) was assigned to cell \#2, \( m2 \) was assigned to cell \#1, and so on. As such cell \#1 contains \{m2, m3, m4, m5, p1, p2, p3, p4, p6\} and cell \#2 contains \{m1, m6, m7, p5, p7, p8, p9, p10\}. In layer two, the chromosome contains 7 genes. The allele of genes encodes the positional weight of machines. For instance, the weight for \( m1 \) is 3, the weight for \( m2 \) is 4, and so on. The weight matrix of this example is \{3, 4, 2, 5, 1, 6, 7\}. By ranking the weights, we infer that the machines will be layout in the order of \{m7, m6, m4, m2, m1, m3, m5\}. Combing the cell information obtained from control genes in layer 1, we obtain that the order to layout machines in cell \#1 is \{m4, m2, m3, m5\} and \{m7, m6, m1\} for cell \#2.

<table>
<thead>
<tr>
<th>Locus</th>
<th>Part (j)</th>
<th>Machine (i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_1</td>
<td>1 2 3 4 5 6 7</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>i_2</td>
<td>3 4 2 5 1 6 7</td>
<td>8 9 10</td>
</tr>
</tbody>
</table>

III. PROPOSED GA HEURISTIC

The proposed GA procedure was adapted from the conventional GA, in which it still maintains and follows the common framework of GA – chromosome encoding, population initialization, fitness evaluation, selection, crossover, mutation, replacement etc [17]-[19]. For instance, we randomly generate the initial population, use the roulette wheel approach for selection, etc.; however, the proposed algorithm contains three intrinsic components:

1) A hierarchical chromosome structure is used to encoding the problem with concurrent decisions
2) A new dynamic selection scheme is developed to evolve between two highly correlated fitness functions, and
3) A group mutation operator is developed to increase the probability of mutation for problems with clustering decisions.
B. Dynamic Selection Strategy

A number of measures have been used to evaluate the performance of cell formation (CF) and machine layout. The popular metrics used are number of exceptional elements \( f_1 \) and total cost of movement \( f_2 \) [15]. Where exceptional element was defined as the part in which some of its operations need to be performed outside the part’s designated cell. In order to determine optimal CF and layout decisions simultaneously, the following approaches can be used in selecting fitness function and dealing with chromosome selection [19]:

1) Use a single objective \( f_2 \) as fitness function and roulette wheel approach for selection.
2) Use linearly weighted \( f_1 \) and \( f_2 \) as fitness function and roulette wheel approach for selection.
3) Use a dynamic assignment to switch between fitness function \( f_1 \) and \( f_2 \) and roulette wheel approach for selection.
4) Use Pareto-based fitness assignment for fitness and roulette wheel approach for selection.

There has been no general guideline regarding which approach is the best approach [19]. Our examination of Schema concepts [20] concluded that, for highly correlated fitness functions (such as our case), the dynamic switch assignment could be a better choice.

A schema is a string of symbols taken from the alphabet \{0,1,#\} for a binary string. The symbol “#” indicates that we “don’t care” what attribute occurs at a given position, thus a schema can represent several bit strings. For example the schema #01# represents four strings: 0010, 0011, 1010 and 1011. Schema Theorem is the fundamental theory proposed by Holland [20] to explain how GAs exploit in parallel the many similarities contained in short, high-performance schemata and illustrate the possibility (necessity) of its finding optimal solution by considering the effect of reproduction, crossover, and mutation on a particular schema.

The usefulness of Holland’s schema theorem has been widely debated [21]-[24]. The theorem certainly has some limitations: it gives only a lower bound for the expected value of the number of instances of a schema at the next generation. The presence of the expected operator means that it is not easy to use the theorem to predict the behavior of a genetic algorithm over multiple generations. Also, its lower bound predictions for schema may be difficult to use in practice. In spite of that, the schema theorem refers to “the specific realized values of competing schemata in a population” [22]. “When correctly interpreted, properly developed and used, schema theorem can be a very useful tool to understand evolutionary algorithms, make predictions on their behavior, and help design competent algorithms” [24]. These disputes however centered more on the quantitative calculation of alternative schema. In this study, we emphasize more on using correlated good schemata for improving solutions and using the concept of schemata to increase the speed of convergence for lengthy chromosomes.

Clearly, with a long or hierarchical chromosome and if the search space is huge, finding all good schemas in such a huge search space will be difficult and time consuming. For highly correlated fitness functions, we propose to use the first function as a criteria to quickly find good fitted first layer chromosomes in the population and then switch to use the second fitness function as a criteria to find good fitted second layer chromosomes. Figure 1 shows the computational logic of our proposed dynamic assignment. The main idea behind the logic is that if the fitness remains unchanged for a fixed number of generations, it is time to switch to the other fitness function or stop the evolution process.

1. Define 
   \( \text{gen}^* \): the gen. to switch fitness function
   \( \text{Nogen} \): the max. allowed number of gen. with no change of fitness value
2. Set
   \( \text{gen}^* \leftarrow M \)
   \( \text{Nogen} \leftarrow 0 \)
3. Set new generation.
   \( \text{gen} \leftarrow \text{gen} + 1 \)
4. Create new population
5. Do {
6. If (gen<\text{gen}* )
7. Select two chromosomes based on \( f_1 \)
8. Check if new fitness improved (<\( \varepsilon \) )
9. If not, check if over allowed # of gen
10. \( \text{Nogen} \leftarrow \text{Nogen} + 1 \)
11. If ( \( \text{Nogen} > \text{g}^* \) )
12. Set current generation to \( \text{gen}^* \)
13. \( \text{Nogen} \leftarrow 0 \)
14. If (\( \text{gen}>\text{gen}^*\) )
15. Select two chromosomes based on \( f_2 \)
16. Check if new fitness improved (<\( \varepsilon \) )
17. If not, check if over allowed # of gen
18. \( \text{Nogen} \leftarrow \text{Nogen} + 1 \)
19. If (\( \text{Nogen} > \text{g}^* \) )
20. Stop
21. }

Fig. 1. Logic of dynamic selection

We show the theory behind the dynamic selection logic below:

**Definition 1.** For schema \( H \), if fitness value of the chromosome with \( H \) is always better than one of without \( H \), then schema \( H \) is a good schema for the fitness function \( f(X) \) and defined as \( H^*(f) \), otherwise as \( H^*(f) \).

**Theorem 1.** Let \( f_1 \) be the first fitness function and \( f_2 \) the second fitness function. If \( f_1 \) and \( f_2 \) are positively correlated
(0 < \rho_{f1f2} \leq 1), then the good schema \( H^* \) for \( f_1 \) is also good for \( f_2 \). That is, \( \{ H^*(f_1) \} \subseteq \{ H^*(f_2) \} \)

**Proof:**

By definition 1, it is easy to obtain \( f_1(H^*) \geq f_1(H^{-*}) \) if there exist \( H^*(f_i) \).

Since \( 0 < \rho_{f1f2} \leq 1 \), then \( f_2(H^*) \geq f_2(H^{-*}) \)

Thus, \( H^*(f_1) \in \{ H^*(f_2) \} \)

**Lemma 1.** Let \( f_1 \) be the number of exceptional elements and \( f_2 \) the total cost of movement. Then there exist a positive correlation between \( f_1 \) and \( f_2 \).

**Proof:** The relationship is clear from CF literature [15].

From the above illustration, we conclude that for two highly correlated fitness functions \( f_1 \) and \( f_2 \), a good schema for fitness function \( f_1 \) is also good for fitness function \( f_2 \). At the initial stage of evolution, if we ignore \( f_2 \), only the genes in the first layer are active (i.e., the genes in second layer are inactive; thus the search space decreases by \( m^n \)). At the second stage, if the good schemas for \( f_1 \) have been decided, then the search space may decrease in time of \( C_{m+n}^n \), and solutions will converge at more promising region than usual. This view is consistent with the observation of biological and medical communities that “some genes dominate other genes and there are active and inactive genes” [19]. Our computational experience also supports our observations.

**C. Crossover Operators**

In this study, due to the unique hierarchical chromosome scheme used, a one-point crossover and its variation – partial mapped crossover (PMX) are used [17][18]. At first, a cut point is randomly selected over the length of the chromosome. Then one-point crossover is used for the layer 1 and PMX is applied to layer 2 to prevent from producing illegal chromosomes [17].

Table II shows an example of the PMX crossover. For instance, if only the one-point crossover was used, the reproduced chromosomes \( C_1' \) and \( C_2' \) may contain some redundant alleles (e.g., 4 and 5 for \( C_1' \)) and miss some other needed alleles (e.g., 1 and 6 for \( C_2' \), which are illegal. To prevent these illegal chromosomes from reproducing, we need to apply PMX crossover to layer 2. In layer 2, for chromosome \( C_1' \) we first replace alleles 1, 6, and 7 with 7, 4, and 5, and then determine the mapping relationship (5<->7<->1 and 4<->6), and finally we partially replace the illegal alleles with their corresponding mapping. For instance, in chromosome \( C_1' \), alleles 4 and 5 are illegal. We replace them with alleles 6 and 1 respectively. Similarly, in chromosome \( C_2' \), besides replacing alleles 7, 4, and 5 with 1, 6, and 7, we replace illegal allele 1 with 5 and allele 6 with 4. We thus obtain final legal children chromosomes \( C_1 \) and \( C_2 \).

**D. Mutation Operators**

Mutation is designed to prevent premature convergence and to explore new solution space. However, unlike crossover, the operation alters or mutates one or more genes within an individual chromosome rather than across a pair of chromosomes.

In this study, we use conventional exchange and also propose a new variation called group mutation for exploration. Exchange mutation involves exchanging the allele of genes from two randomly selected locuses each time. Group mutation, on the other hand, exchanges the alleles for the two groups of genes for layer 1’s genes all at once. Table III (a) shows an example of exchange mutation and (b) for group mutations. Where genes 3 and 7 were randomly selected for mutation.

Our motivation for developing group mutation is aroused from the concept of Schema that “It is more difficult to find the schema (H) with long defining length \( \delta(H) \) and larger number of fixed positions [20].” Let \( p_{e} \) denote the probability of each locus being selected. \( P_m \) is the mutation probability. \( P_p \) is the probability of transforming all cell members from one cell number to another using exchange mutation, and \( p_p \) is the probability of transforming all cell members from one cell number to another using group mutation. We give the following theorem to support our proposal.

**Theorem 2.** For classifier problem, the probability of finding good schemas using group mutation is higher than using exchange mutation.
Proof:
If the length of chromosome is \( l \), then \( p_s = 1/l \). Assume that there are \( q \) members in one group and \( r \) members in another cell. The probability of fully exchanging these two groups is:

\[
\begin{align*}
    p_e & = \left( \frac{1}{l} \right)^q \cdot \left( \frac{1}{l} \right)^r \cdot p_m = l^{-(q+r)} p_m \\
    p_g & = \frac{q}{l} \cdot \frac{r}{l} \cdot p_m = \frac{qr}{l^2} p_m = qr \cdot l^{-2} p_m \\
\end{align*}
\]

\[ \therefore \ 1 \leq q < l \text{ and } 1 \leq r < l, \left( \frac{1}{l} \right)^{qr} < q r \left( \frac{1}{l} \right)^2 \]
\[ \therefore \ p_e < p_g \]

In short, group mutation can help to enhance search ability in the entire search space and converge rapidly to a promising region.

### TABLE III
NUMERICAL ILLUSTRATION OF MUTATIONS

<table>
<thead>
<tr>
<th>Locus</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Exchange Mutation:

(b) Group Mutation:

<table>
<thead>
<tr>
<th>Locus</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IV. COMPUTATIONAL ANALYSIS

To test the performance of our proposed heuristic GA, five data sets collected from prior literature are considered. Four of them are relatively small size - \( 5 \times 7, 7 \times 14, 5 \times 7, 15 \times 30 \) [6] and one - \( 40 \times 100 \) [5] is bigger. For each data set, the data was randomly disturbed for 50 times. The worst and best data set were then identified and run for 50 times with different random seeds. We collect the average of performance in terms of total cost of movement. Our purpose of making such arrangements is to detect any possible impact of data sequences and random number generator. The algorithm was coded in Visual C++ and we use a PC with Intel Pentium IV (1300 MHz) processor to test our implementation. The parameter values used are 0.95 for crossover rate, 0.3 for mutation rate, and the population size is 100. These values were determined based on our pilot tests.

For small data sets, we can obtain the best solutions using any combination of operators in a short period of time (less than 5 seconds). However, that is not the case for larger data set. Therefore, we only report our experience for the large data (40x100).

Figure 2 depicts the computational results for cases without dynamic selection, without group mutation, and with both operators. As can be seen, clearly without implementing the dynamic selection, the solution seems converged but the quality was not acceptable. It converged to around 40891, which is far worse than the reference solution 3474 [5]. With dynamic selection but without group mutation, the solutions are significantly improved and converged to 6665. Yet this solution is worse than the reference solution. With both dynamic selection and group mutation, the solution converges to 2812 in around 31 generations (about 24 seconds with our computer). Our computational tests show the relative effectiveness of the proposed group mutation operator and the dynamic selection logic. Clearly, dynamic selection logic plays a very important role toward fast convergence. The group mutation further improved the solution quality.

To explore whether there is any possible impact on solution quality by randomized data and random number seed. We conduct pair-t tests for each case. Table IV summarizes the statistical results. Although we observed few individual fluctuating runs, based upon the statistical analyses, we reject the hypothesis that there is a significant impact by both randomized factors. The average results of both cases are better than the best solution found in [5]. This also confirms the super performance of our proposed heuristic genetic algorithm.

V. CONCLUSIONS

In this paper, we proposed a heuristic genetic algorithm to concurrently solve the cell formation and machine layout decisions. A two-layer hierarchical chromosome structure was developed for problem domains that may need to deal with
concurrent decisions. The hierarchical structure allows us to shorten the length of chromosome; thus, it would help to increase the probability of finding good schemas. It also helps to reduce the number of active genes during operations thus shorten the performance time. We have also proposed a new dynamic selection method to deal with concurrent decisions that involve highly correlated objectives. Our tests show that without implementing the dynamic selection logic the GA would not properly converge to a good solution. The proposed dynamic selection logic can be used to solve problems with multiple objectives, especially when these objectives are correlated. In addition, a new group mutation operator was developed to increase the mutation probability. It can be used to further improve the solution quality, especially for classifier related problems. However, our tests show that using group mutation alone may not be able to obtain good performance. The best results are found when it was combined with dynamic selection operator.

| TABLE IV |
| T TEST OF RANDOM EFFECTS |

<table>
<thead>
<tr>
<th>Source of Variance</th>
<th>Reference</th>
<th>Random Data</th>
<th>Random Seed</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>1383.84</td>
<td>6052.84</td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>1383.84</td>
<td>6052.84</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>0.085746</td>
<td>0.244857</td>
<td></td>
</tr>
<tr>
<td>Conclusion</td>
<td>Not Significant</td>
<td>Not Significant</td>
<td></td>
</tr>
</tbody>
</table>

REFERENCE


