A New Algorithm for Solving the Operation Assignment Problem in 3-Machine Robotic Cell Scheduling

1Mohammad Fathian, 2Isa Nakhai Kamalabadi, 3Mehdi Heydari and 4Hiwa Farughi
1,3,4Industrial Engineering, Iran University of Science and Technology, Tehran, Iran.
2Industrial Engineering, Tarbiat Modares University, Tehran, Iran.

Abstract: Robotic manufacturing cells are now used extensively in manufacturing systems in order to ease the operations and replace humans with robots in hard and dangerous working conditions. Since the machines used in these manufacturing systems are CNC machines, their application entails much expenses. Moreover, their being idle during manufacturing operation is not beneficial. Therefore, in this article the problem of optimal assignment of necessary operations to manufacture parts in 3-machine robotic manufacturing cells is considered. So, inspired by 3-partition problem, a mathematical model based on Travelling Salesman Problem is presented for modeling the problem. Moreover, for solving the presented model, along with the exact solution approach two metaheuristic methods named genetic and simulated annealing algorithms are also presented; several problems are solved using these approaches and their results are compared.

Key words: Optimal Operation Assignment, 3-machine Robotic Cell, 3-partition Problem, Metaheuristic Algorithm.

INTRODUCTION

Along with technological development in industries and organizations managers’ decisions and their organizational activities and strategies have become very complicated. One of these strategies is to create an automation system in manufacturing organizations and industries; in this regard, CNC machines and a mechanical and programmable device named robot or manipulator are used to move parts between different machines and stations. By arranging the machinery in a cell layout and using robot for automation process managers proceed to minimize manufacturing time which leads to an increase in the efficiency of production line or, in other words, an increase in manufacturing output in robotic manufacturing cells. On the other hand, decreasing manufacturing expenses in these manufacturing systems is one of the main concerns of production managers. The main expenses in these systems are those related to buying, depreciation, idle time and using manufacturing machinery in these manufacturing cells. Therefore, in this article, optimal assignment of necessary operations to manufacture parts in a 3-machine robotic manufacturing cell is considered. This problem is an evident practical example of partitioning the members of a particular set into several subsets which has long been a focus of attention for different science. One of the commonest problems of this type is 3-partition problem. Optimal operation assignment problem has been considered from different aspects and by different approaches in the past literature. A review of recent development about robotic cell scheduling has been done by Dawande et al., (2005). They also (Dawande, M. et al., 2002) studied maximization the output rate in robotic cell with additive movement times. Crama et al., (1997) considered cyclic scheduling in 3-machine robotic cell scheduling in flow shop environment. Also Hall et al., (1997) and Luan et al., (2006) studied about operation assignment and scheduling in two and three machine robotic cell. Geismar et al., (2005) considered productivity gain by operation assignment and scheduling in robotic cells. Since the machines used in these manufacturing cells are numerical control machines and utilizing them entails high expenses, it is necessary that assigning necessary operation to manufacture parts in these manufacturing cells be in a way that the machines have minimum idle time during manufacturing process. Therefore, it seems reasonable that optimal designation of operation to these machines should be in way that the total processing time of operations on each machine has the minimum difference with those on other machines. If set O={1,2,3,...,P} is the total operations on all three machines, we intend to partition set O into three independent subsets-O1, O2, O3-in a way that O1 will be operations assigned to the first machine, O2 will be operations assigned to the second machine and O3 will be that of the third machine. To optimally assign the set of total operations to all three machines we use the set of total operations’ processing times{t1,t2,t3,...,tP}.

Since the problems of assigning operations are NP-complete problems they cannot be solved by simple mathematical methods, instead, heuristic algorithms are used to solve them. One of these algorithms is Karmarkar and Karps’s (1982) heuristic algorithm which is a differencing algorithm; this algorithm is used for sets with fewer members and it is not a very good method for sets that have many members. Another heuristic algorithm used in this field was Hayes’s (2002) greedy algorithm which was used to partition set S into three subsets S1, S2 and S3. This algorithm will also have optimal answer for sets with limited members; this is an...
impossible approach for sets which have many members. Another heuristic algorithm was presented by Mertens (2006) which is based on differencing algorithm; in this algorithm, the solution is first calculated by a heuristic approach and then, as far as time allows, some better solution are obtained in order. This differencing heuristic approach outperforms the greedy heuristic approach. Some other proposed heuristic algorithms in this field are the studies of Gültekin et al., (2010), Graham (2002), Mertens (1998,2001,1999), Korf et al., (1998) and Borgs et al., (2009). In this article, a new TSP-based algorithm is presented to solve the problem of optimal operation assignment to three machines and its performance is investigated by some examples.

**The Mathematical Model:**

As mentioned before, in this article the scheduling and operation assignment problem is studied in a 3-machine robotic cell with \( l \) distinct operations to manufacture each part in the framework of \( S_1 \) cycle movement policy. Regarding the following assumptions and assuming that each part needs \( l (l = 1,2,3,...,P) \) distinct operations on the machines to be prepared, the scheduling and operation assignment problem is studied in a 3-machine robotic cell. \( S_1 \) is a cycle in which the mechanical arm or robot begins its movement from an initial state of the system, during a cycle performs each activity exactly one time and finally returns to the initial state of the system (Sethi, S.P. et al., 1992). In \( S_1 \) cycle, the robot loads the first machine in the beginning; next, after the processing operation on the part is finished on the first machine the robot moves it to the second machine; then, after the processing operation on the part is finished on the second machine the robot moves it to the third one; finally, after the processing operation on the part is finished on the third machine the robot unloads it and moves it to output buffer. Then turns back to input buffer or initial station. The problem’s assumptions, indices, parameters and decision variables are as follows:

1) **Assumptions:**

In this section, the assumptions related to scheduling problem of a robotic cell with three CNC machines with the condition that the cell is flexible are defined as follows:

1. If a job begins on a machine, it will remain on the same machine until it is finished. In other words, no preemption is allowed.
2. If an operation begins on a machine, no other operation can replace it until it is through unless we define a waiting mode for the problem.
3. The machines cannot be idle during operations processing.
4. No machine fails during the operation; they are all available throughout the schedule.
5. All the machines belong to one type or are identical.
6. The machines are CNC ones.
7. Robotic cell system is flexible, i.e. it can simultaneously process several operations.
8. There is a mechanical arm (robot) in the robotic cell.
9. Production is cyclic.
10. The problem is in a flow shop (Fm) layout.
11. Production policy is the same for all parts.
12. The mechanical arm (robot) can not reload a loaded machine.
13. The mechanical arm (robot) can not unload an idle machine.
14. All the jobs are equally valuable and all of them should complete their process.
15. Machines’ setup time for different jobs can be ignored.

2) **Indices:**

1. \( i \): job related indice (\( i=1,...,n \)).
2. \( k \): machine related indice (\( k=1,...,m \)).

3) **Notations:**

1. \( n \): number of jobs.
2. \( t \): job’s processing time.
3. \( m \): number of machines.
4. \( a \): Total processing times related to the jobs assigned to the first machine.
5. \( b \): Total processing times related to the jobs assigned to the second machine.
6. \( c \): Total processing times related to the jobs assigned to the third machine.
7. \( T \): Total processing time of all jobs.

4) **Decision Variables:**

\[
X_{ki} = \begin{cases} 
1 & \text{if the } i^{th} \text{ operation is processed by the } k^{th} \text{ machine} \\
0 & \text{otherwise} 
\end{cases}
\]
With regard to the mentioned assumptions and variable definitions, parameters and indices the problem’s mathematical model is presented as follows:

\[
\min Z = \left| a - \frac{T}{3} \right| + \left| b - \frac{T}{3} \right| + \left| c - \frac{T}{3} \right| \quad (1)
\]

\[
T = \sum_{i=1}^{n} t_i \quad (2)
\]

\[
a = \sum_{i=1}^{n} x_{1i} \times t_i \quad (3)
\]

\[
b = \sum_{i=1}^{n} x_{2i} \times t_i \quad (4)
\]

\[
c = \sum_{i=1}^{n} x_{3i} \times t_i \quad (5)
\]

\[
x_{1i} + x_{2i} + x_{3i} = 1 \quad \forall i \quad (6)
\]

\[
x_{ki} \in \{0, 1\} \quad \forall i, k \quad (7)
\]

The objective function of the above model (Eq. (1)) illustrates sum of the absolute value of processing time (of the operation on each machine) deviation from one third of the total processing time of all operations (the virtual ideal case is equal processing time for all three machines).

The first constraint (Eq. (2)) calculates sum of processing time of all operations. The second (Eq. (3)), third (Eq. (4)) and fourth constraints (Eq. (5)) calculate the total processing time of operations assigned to the first, second and third machine, respectively. The fifth constraint (Eq. (6)) insures that each operation is assigned only and only to one machine. The sixth constraint (Eq. (7)) illustrates that the problem’s variable is a binary integer variable.

As it is observed, the objective function of the above model is nonlinear; the following variable changes have been used to make it linear:

\[
\left| a - \frac{T}{3} \right| = y_1 + y'_1
\]

\[
\left| b - \frac{T}{3} \right| = y_2 + y'_2
\]

\[
\left| c - \frac{T}{3} \right| = y_3 + y'_3
\]

Considering the above variable changes to make the problem linear, the following constraints are added to the model:

\[
\left| a - \frac{T}{3} \right| = y_1 - y'_1
\]

\[
\left| b - \frac{T}{3} \right| = y_2 - y'_2
\]

\[
\left| c - \frac{T}{3} \right| = y_3 - y'_3
\]

In the above relations, \(y_k\) and \(y'_k\) are non-negative continuous variables.
Problem Solution:

For solving the proposed model in this article, in addition to exact solution methods simulated annealing and genetic algorithm are also used.

3.1. Genetic Algorithm:

This algorithm is the most famous evolutionary algorithm; its overall structure is as follows:

A solution structure is first defined to show the problem’s answers. Using this structure the primary generation of solutions is randomly created by predefined population size. Then the repetitive loop of the algorithm begins. In each run of this loop chromosomes and genes that have the necessary condition for crossover and mutation are selected and crossover and mutation operators are applied on them. As a result of this process new chromosomes (offspring) are generated. The set of primary chromosomes (parents) and new chromosomes (offspring) create the pool. The size of each generation is in genetic algorithm is fixed. Since the number of chromosomes in the pool is larger than generation’s size the next generation is selected from pool chromosomes. Then the stop condition is investigated; if the condition holds, the algorithm will end and the best chromosome of the last generation will be selected as the best answer, otherwise, another repetition of the algorithm will be performed.

The main components of genetic algorithms are as follows:

- Coding.
- Generating primary population.
- Determining fitness.
- Operators of genetic algorithm.
- Selection process.
- Algorithm’s parameter values.

3.1.1. Coding:

The first stage of genetic algorithm is to structure decision variables (coding). The structure of decision variables is often illustrated by a vector or matrix; each item of this vector or matrix is one decision variable. The structure of decision variables in a genetic algorithm is called chromosome and each item of a chromosome is called gene. In this article, the following chromosome structure has been used:

![Fig. 1: Solution structure.](image)

In this structure, $n$ is the number of jobs; $r_1$ to $r_n$ denote the number of a machine to which the related job is designated. Each one of these numbers is selected from one to three (the number of machines is three). After the structure of the chromosome was determined the primary solution is randomly generated. To generate a chromosome for the problem the presented structure has been used; in this structure, each gene of this chromosome is randomly selected from one to three numbers.

3.1.2. Generating the Primary Population:

The collection of chromosomes is called population. Instead of focusing on one solution (chromosome) the genetic algorithm works on a collection of chromosomes. The size of each population or generation of chromosomes is called population size. Population size reveals the number of chromosomes in each generation. If population size is small, only one part of solution space will be studied. On the other hand, if population size is large, the algorithm will be very slow.

3.1.3. Fitness:

The value of objective function according to each chromosome is called fitness of that chromosome.

3.1.4. Genetic Algorithm Operators:

In genetic algorithm, the approaches that are performed on one or more chromosomes and generate one or more new chromosomes are called genetic operators. The most important genetic operators are crossover and mutation.

- Crossover operator: this operator is applied on a pair of chromosomes and generates one or more chromosomes. In this article, one-point-cut crossover operator has been used. To apply a one-point-cut crossover operator the cut point is randomly selected at first. Assume that the cut point is $r$, to apply the
crossover operator $r_r$ genes of each chromosome are swapped so that new chromosomes are generated. In figure 2 one-point-cut crossover operator is illustrated.

![Figure 2: The method of applying the crossover operator on two parent chromosomes.](image)

- **Mutation operator:** this operator is applied on one chromosome and changes one or more genes on it so that a new chromosome is generated. The primary chromosome is called parent and the new one is called offspring. To apply mutation operator in this article, one gene is randomly selected at first, then the gene’s value is changed in a way that the generated chromosome be feasible. Figure 3 illustrates the method of applying the mutation operator on a chromosome.

![Figure 3: The method of applying the mutation operator on one chromosome.](image)

3.1.5. **Selection Process:**
In this article, mix or combined strategy have been used to select new generation chromosomes; in this way, a percentage of the best pool chromosomes are placed in the new generation and then the remaining chromosomes of the next generation are selected randomly from pool chromosomes.

3.1.6. **The Value of Algorithm Parameters:**
The necessary parameters for run of genetic algorithm are population size, crossover probability, mutation probability and the number of generations. The number of repetitions is based on the problem size; if size of the problem is large, the more number of repetitions should be considered. Crossover operator has a great role in genetic algorithm, therefore, its probability is considered to be high. Mutation operator leads to diversity of solutions, but if a high probability is considered for it, the running time of the algorithm will increase; therefore, a small probability is considered for it. In setting the parameters one should be careful that there should be a balance between algorithm’s running time and the quality of solutions.

3.2. **Simulated Annealing Algorithm:**
Simulated annealing algorithm (SA) is inspired by the process of metals annealing. In a real annealing process, metal’s temperature is increased to a point that all the molecules are scattered in a molten form, then temperature will slowly decrease. Temperature decrease in real annealing is like decreasing the value of objective function for minimization problems. As temperature decrease rate in real annealing is effective on the quality of the final material, temperature decrease rate in SA algorithm is also effective on the quality of the final solution. One characteristic of SA algorithm is that it accepts non-improving solutions; this causes the algorithm not to get captured at a local optimum. This algorithm begins with a primary random solution; then in each temperature some of primary solution neighborhoods are studied. If objective function’s value of the neighbor’s solution is better than objective function’s value of the primary solution, the neighbor’s solution will be accepted, otherwise, to escape the local optimum the neighbor’s solution with probability of $exp\left(-\frac{\Delta}{T}\right)$ will be accepted. This process will be repeated until a predefined number of neighborhoods are studied in each temperature. Afterwards the temperature will decrease; the predefined number of neighborhood is also studied in this new temperature. The algorithm will stop when it reaches the stopping criterion. The pseudo-code of simulated annealing algorithm is illustrated in figure 4.
Select an initial temperature \( T_0 \)
select an initial solution, \( S_0 \) and make it the current solution, \( S \) and the current best solution, \( S^* \).
repeat
set repetition counter \( n=1 \)
repeat
generates solution \( S_n \) in the neighbourhood of \( S \)
calculates \( \Delta = f(S_n) - f(S) \)
if (\( \Delta \leq 0 \)) then \( S = S_n \)
else \( S = S_n \) with probability of \( p = \exp \left( -\frac{\Delta}{T} \right) \)
if (\( f(S_n) < f(S^*) \)) then \( S^* = S_n \)
\( n = n+1 \)
until \( n > \) number of repetition allowed at each temperature level (\( L \))
reduce the temperature \( T \)
until stop criterion is true.

Fig. 4: Pseudo-code of SA algorithm for minimization problems (Xambre, A.R., P.M. Vilarinho, 2003).

The basic parameters of SA algorithm are as follows:

3.2.1. Initial Temperature:
Initial temperature is one of the basic parameters of SA algorithm. The initial temperature should be selected in a way that most of non-improving solutions are accepted in the first repetition. In this article, the following heuristic method has been used to determine the initial temperature.

Sub Init Temp
Do
Generate two solution \( X_1, X_2 \) at random
Loop until (\( f(X_1) \neq f(X_2) \))
Set \( T_0 = \frac{-|f(X_1) - f(X_2)|}{\ln (0.9)} \)
End sub

Fig. 5: Pseudo code of the initial temperature generation subroutine (Safaei, N. et al., 2008).

3.2.2. Temperature Decrement Rule:
SA algorithm begins with a relatively high temperature which decreases slowly in each repetition. There are various methods to reduce temperature in each repetition; in this article, geometric scheduling criterion has been used:

\[
T_k = \alpha T_{k-1} \quad 0 < \alpha < 1
\]  

(14)

In the above relation, \( T_k \) is system’s temperature in the \( k^{th} \) repetition and \( \alpha \) is temperature reduction rate. Selecting a large value for \( \alpha \) results in slow temperature decrement and better solution space searching, on the other hand, it increases algorithm’s run time. Selecting a small value for \( \alpha \) results in fast temperature decrement and fast solution space searching. Therefore, \( \alpha \) value should be selected in a way that there will be a balance between algorithm’s run time and the quality of solutions. In this paper, \( \alpha \) value has been considered to be 0.9.

3.2.3. Neighborhood Structure:
Neighbor solutions are a set of feasible solutions which are obtained from the primary solution. Each neighbor solution can be obtained through one movement (a change in the present solution). In this article, the following neighborhood structure has been used; in this structure one of the genes is randomly selected and then its value is changed in a way that the resulted solution will be feasible.

Fig. 6: Neighborhood structure.
3.2.4. Number of Repetitions in Each Temperature (L):
This parameter controls the number of investigated neighborhoods in each temperature. $L$ value should be selected large to the extent that results in an effective neighborhood search. On the other hand, $L$ value should not be so large that would result in ineffective search and increase run time. In this paper, number of repetitions in each temperature ($L$) has been considered to be 10.

3.2.5. Stopping Criterion:
In the presented algorithm, stopping criterion has been considered to be the final temperature. The final temperature should be selected in a way that the probability of accepting non-improving solutions in the final repetitions will be close to zero.

4. Numerical Examples:
In this section, ten numerical examples are solved to evaluate the presented model. The early examples have a small scale, but it increases regularly in a way that the final examples have a large scale. Processing time of each job has been obtained through the following simulation approach:

Processing time of each job (second): uniform distribution $U \sim [10, 200]$.

Small, average and large scale problems have been used to study $SA$ and $GA$ algorithms. Each one of these problems have been solved by $LINGO 8.0$ software at first. $LINGO 8.0$ is able to produce global optimal solutions for problems. But, when dimensions of the problem increase, this software cannot obtain optimal solutions in a reasonable time. In this article, the maximum running time of $LINGO 8.0$ has been considered to be five hours. That is, if $LINGO 8.0$ cannot obtain the global optimal solution in less than five hours, the solution algorithm will stop and the best solution obtained by the software will be considered as its output. Therefore, metaheuristic algorithms should be used for solving high dimension problems. The presented metaheuristic algorithms have been encoded by $MATLAB 7.0$ and have been implemented via a PC with 2.4 GHz of CPU and 4 Gb of RAM. A summary of parameter values necessary to implement the presented genetic algorithms are displayed in table 1.

The results of solving the problems by $LINGO 8.0$ and $GA$ and $SA$ algorithms are illustrated in tables 2 and 3.

Table 1: The values of the applied parameters for genetic algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Population size</th>
<th>Crossover probability</th>
<th>Mutation probability</th>
<th>Number of repetitions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10-100</td>
<td>0.7-0.9</td>
<td>0.01-0.1</td>
<td>Varied</td>
</tr>
</tbody>
</table>

Table 2: The values of the objective functions obtained via $LINGO 8.0$.

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Number of jobs</th>
<th>Best solution value</th>
<th>Objective Bound</th>
<th>run time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>49.3333</td>
<td>49.3333</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8.6667</td>
<td>8.6667</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>1.33333</td>
<td>1.33333</td>
<td>1500</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>1.33333</td>
<td>0</td>
<td>18000</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>1.33333</td>
<td>5</td>
<td>18000</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>1.33333</td>
<td>0</td>
<td>18000</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>1.33333</td>
<td>0</td>
<td>18000</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>1.33333</td>
<td>0</td>
<td>18000</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>1.33333</td>
<td>0</td>
<td>18000</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>1.33333</td>
<td>0</td>
<td>18000</td>
</tr>
</tbody>
</table>

Table 3: The values of the objective functions obtained via metaheuristic algorithms.

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Number of jobs</th>
<th>algorithm</th>
<th>Best solution value</th>
<th>Average solution value</th>
<th>Worst solution value</th>
<th>Average performance time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>GA</td>
<td>49.3333</td>
<td>49.3333</td>
<td>49.3333</td>
<td>0.156</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>49.3333</td>
<td>49.3333</td>
<td>49.3333</td>
<td>0.7737</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>GA</td>
<td>8.6667</td>
<td>8.6667</td>
<td>8.6667</td>
<td>0.18096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>8.6667</td>
<td>8.6667</td>
<td>8.6667</td>
<td>0.85176</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>GA</td>
<td>1.33333</td>
<td>1.33333</td>
<td>1.33333</td>
<td>0.8548</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>3.3333</td>
<td>3.3333</td>
<td>3.3333</td>
<td>1.2248</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>GA</td>
<td>3.3333</td>
<td>1.9999</td>
<td>1.3333</td>
<td>0.86112</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>3.3333</td>
<td>2.5333</td>
<td>1.33333</td>
<td>1.2324</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>GA</td>
<td>3.3333</td>
<td>1.73333</td>
<td>1.33333</td>
<td>0.87048</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>3.3333</td>
<td>1.73333</td>
<td>1.3333</td>
<td>1.24176</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>GA</td>
<td>1.33333</td>
<td>1.33333</td>
<td>1.33333</td>
<td>0.87048</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>4.6667</td>
<td>2.39998</td>
<td>1.33333</td>
<td>1.3135</td>
</tr>
</tbody>
</table>
5. Conclusion:

In this article, a new TSP-based algorithm has been proposed to solve the problem of optimal operations assignment to three machines in robotic manufacturing cells. This problem is an evident instance of 3-partition problem. Exact solution, genetic algorithm and SA algorithm have been used to solve the proposed mathematical model and the obtained results have been compared. The results show that the proposed model is effective in obtaining appropriate solutions as compared to previous heuristic algorithms. Moreover, regarding the model’s complexity, the quality of obtained results from SA and GA methods in shorter run times denotes the appropriate efficiency of these proposed solution algorithms.

REFERENCES


