CMS scheduling problem considering material handling and routing flexibility

Mojdeh Pajoutan · Amirmohsen Golmohammadi · Mehdi Seifbarghy

Received: 5 March 2012 / Accepted: 6 February 2014 / Published online: 1 March 2014
© Springer-Verlag London 2014

Abstract Cell manufacturing as an application of group technology increases the flexibility and efficiency of the production. Cell scheduling problem, one of the subjects in cell manufacturing, has not been widely studied by researchers compared with other problems in cell manufacturing. In spite of great importance of material handling in cell scheduling, it has not been paid enough attention by researches. In this paper, a new mathematical model for cell scheduling problem considering material handling time and routing flexibility is proposed. The proposed model belongs to the mixed-integer nonlinear programs (MINLP). A linearization procedure is proposed to convert the MINLP to an integer program (IP) in order to develop more powerful optimization tools. Furthermore, a simulated annealing-based heuristic is developed to solve the large-size problems.

Keywords Cell manufacturing · Routing flexibility · Material handling · Simulated annealing

1 Introduction

In the recent decades, production systems have changed over due to the increase of competition in markets. In the past, production volume and finished costs of goods were the two major determinant competitive factors; thus, the tendency to use flowshop production systems was pervasive among the companies. Nowadays, other factors such as variety in products and swift response to the market’s demand are of higher importance. Consequently, production systems with higher level of flexibility have been developed and applied by both practitioners and academics. Job shop can be considered as a clear-cut example of such systems.

The aforementioned changes along with the development of new businesses have created a novel environment for manufacturing and competing in markets. In such an environment, reducing production costs and make-span, increasing flexibility, and precipitating reactions to the market’s needs are turning into the critical competitive advantages of companies. In the 1970s, through highly competitive atmosphere among the companies in the US, a few modern management concepts such as just in time (JIT) and group technology (GT) came into existence. Cell manufacturing system (CMS) is considered as an application of GT, in which the whole machines are divided into a number of distinctive groups according to the similarities in processing the assigned parts. The groups are called cells. The set of parts processed in each cell is called a part family. Cell formation, cell layout, production planning, and cell scheduling are the major issues in CMS (Solimanpur et al. [25]). Unlike the cell formation problems, cell scheduling has not been widely studied by researchers. Moreover, in spite of the great importance of material handling issue, it has not been considered in the most of cell scheduling problems.

As Nomden and Van der Zee [19], routing flexibility provides the possibility to choose from among a number of machines to execute an operation. We consider routing flexibility in terms of alternative machines available for a product family. On the other hand, there will be alternative production routes per product family. This issue can increase the flexibility of a manufacturing system; however, it can increase the complexity of the problem.
In the current research, a new mathematical model for a cell scheduling problem is proposed. The main contributions of the proposed model are explained as follows:

- There are some papers that have considered routing flexibility in cellular manufacturing problem such as Kioon [14]. To the best of our knowledge, this issue has not been directly integrated with CMS scheduling problem while it may have great influence on the sequencing of the jobs.
- Material handling time is also one of the subjects, which is undermined in the literature of CMS scheduling. In the proposed model, the time is determined based on the characteristics of the part and position of the machines.
- A number of the researches in this sphere assume that all the operations of a part family on a particular machine have to be done consecutively and without any interruption [16, 25]. This assumption leads to minimizing the total setup times and may be interpreted as a proper policy for minimizing the makespan. This assumption reduces the solution space and as a result can decrease the computational time. However, it is clear that in many cases, consecutively, performing operations of a part family is not optimal and may impede the solving algorithm from finding the global optimum. In the given model, a machine can process the parts without any restriction on the sequence considering the setup times. This approach extends the applicability of the proposed model to the job shop environment scheduling.

The complexity of scheduling problems and the nonlinearity of the models hinder the applicability of the regular optimization tools. Therefore, we alleviate this problem by converting the nonlinear model into a linear one in order to implement more efficient optimization tools. Finally, a simulated annealing (SA) based heuristic is proposed in order to tackle the large-size instances of the problem.

This paper is organized as follows: a literature review of CMS scheduling is presented in Section 2. In Section 3, the problem description and the mathematical model are given. The linearization process is demonstrated in Section 4, and we introduce the proposed SA-based heuristic in Section 5. Section 6 gives the computational results. In Section 7, conclusions and further research ideas are given.

2 Literature review

Generally, a significant portion of the literature in CMS is devoted to the flowline-based cellular manufacturing. In this sphere, it is assumed that all the parts follow a similar sequence of processes; however, there may be some missing operations on some machines. Since the proposed model in this paper can be considered as a submodel in cell scheduling, we directly go through the corresponding literature.

There are a few researches focusing on the minimization of tardiness of jobs in cell scheduling problem. Parthasarathy and Rajendran [20] study the problem of scheduling in flowshop and flowline-based manufacturing cell (or simply, a cell) with the objective of minimizing mean tardiness of jobs. A heuristic algorithm, based on the SA technique, is developed. The proposed SA algorithm with two schemes is evaluated against the existing heuristics that seek to minimize mean tardiness of jobs. The results of the computational evaluation reveal that the SA algorithm with two schemes performs better than the existing rival heuristics. Rajendran and Ziegler [22] study the addressed problem with the objective of minimizing the sum of weighted flowtime and weighted tardiness of jobs. Initially, heuristic preference relations are developed considering lower bounds on the completion times, operation due dates, and weights for holding and tardiness of jobs; then, a heuristic algorithm is proposed making use of the heuristic preference relations. Two more heuristic algorithms are developed, implementing an improvement scheme on the solution given by the first heuristic algorithm.

There are a number of successive researches on the scheduling of part families and jobs within each family in a flowline or flowshop (all jobs available at time zero, different job availability times known a priori) manufacturing cell with sequence-dependent family setups times. Schaller et al. [23] study the addressed problem. The objective is to minimize the makespan while processing parts (jobs) in each family together. França et al. [6] study the problem in a flowshop manufacturing cell. Two evolutionary algorithms including a genetic and a memetic algorithm with local search are proposed and empirically evaluated as to their effectiveness in finding optimal permutation schedules. Hendizadeh et al. [9] present various tabu search (TS)-based meta-heuristics for the problem to minimize makespan. Concepts of elitism and the acceptance of worse moves inspired from simulated annealing are considered in the proposed meta-heuristics to improve intensification and diversification. The effectiveness and efficiency of the proposed heuristics are compared against the best rival meta-heuristic and heuristic algorithms. Lin et al. [16] declare that almost all published studies in this regard focus on using permutation schedules to deal with flowline manufacturing cell scheduling problem with sequence-dependent family setup times. To explore the potential effectiveness of treating this argument using nonpermutation schedules, three prominent types of meta-heuristics including SA, genetic algorithm (GA), and TS are proposed and empirically evaluated. The experimental results demonstrate that the improvement made by nonpermutation schedules over permutation schedules for the due date-based performance criteria were significantly better than that for the completion-time-based criteria. Bouabda et al. [2] address the permutation flowline manufacturing cell with sequence-dependent family setup time problem with the objective to minimize the
makespan criterion. A cooperative approach including a GA and a branch and bound (B&B) procedure is developed. The latter is probabilistically integrated within the GA.

There are a number of other dispersed researches related to the cell scheduling problem which are reported here. Sridhar and Rajendran [27] develop a multiobjective model minimizing the makespan, the total flow time, and the machine idle time. They utilize a GA-based heuristic to solve the proposed problem. Solimanpur et al. [25] declare that the scheduling problem in a cellular manufacturing environment is treated as group scheduling problem, implying that all parts in a part family to be processed in the same cell. However, there could be some exceptional parts, which need to visit machines in the other cells. This fact limits the applicability of group scheduling approaches. The addressed research study the scheduling of manufacturing cells in which parts may need to visit different cells. A two-stage heuristic is proposed to solve this problem. These stages are termed as intracell scheduling and intercell scheduling. Through intracell scheduling, the sequence of parts within manufacturing cells is determined. In intercell scheduling, however, the sequence of cells is obtained. Sidhartha and Canel [4] focus on the problem of scheduling batches of parts in a flexible manufacturing system (FMS). Due to the use of serial access material-handling systems in many FMSs, the problem is modeled for a multiecell FMS with flowshop characteristics. A B&B solution method is developed in order to solve the problem. Das and Canel [4] propose a new multiecell scheduling model. They also developed a B&B method to solve the problem. Although they developed a model assuming multiple cells, the processes of parts on machines were not distinguished, and only the process times of parts were given on each cell. Considering this assumption, they dominated the characteristics of flowshop on their proposed model. Venkataramanaiyah [30] develop a SA-based heuristic to minimize makespan, flow time, and machine idle time in cellular manufacturing environment.

Kesen and Gungor [13] design a GA-based heuristic approach for job scheduling in virtual manufacturing cells (VMCs). In a VMC, machines are dedicated to a part as in a regular cell, but machines are not physically relocated in a contiguous area. Cell configurations are therefore temporary, and assignments are made to optimize the scheduling objective under changing demand conditions. This research considers the case where there are multiple jobs with different processing routes. There are multiple machine types with several identical machines in each type. Weighted makespan and total traveling distance are considered as the scheduling objectives. Gholidpur-Kanani et al. [8] develop a new model which minimizes the number of intercellular movements as well as intracellular movements, makespan tardiness, and sequence-dependent set-up time, however, not considering routing flexibility and material handling time. In addition, they posed the characteristics of the flowline-based cellular manufacturing to their model assuming that the process of each part family cannot be interrupted by other parts. This assumption can increase the machines’ idle time and consequently the makespan significantly. Chalapathi Pasupuleti [3] declares that once the cellular manufacturing system is designed, scheduling of jobs is essential for the day-to-day production in the machine cells. A methodology for prioritizing the parts, as well as preparing the total schedules in cellular manufacturing system, is proposed in this research. It takes into account the processing sequences of the jobs, processing and setup times, and due dates. The method presents the sequence of parts to process on each machine and the total schedules for all the operations of the parts. Detailed reviews of flowline-based cellular manufacturing scheduling problem can be found in Allahverdi et al. [1].

Although there are a number of researches in cell scheduling, but there is not any research considering material handling time and routing flexibility. Jensen et al. [10] believes that for a parallel shop, full routing flexibility can be unfavorable because this results in extensive set-up efforts; instead, the tradeoffs between routing flexibility and setup efficiency must be made carefully. Sheikhzadeh et al. [24] declare that routing flexibility is useful in the cases where either the number of machines or length of setup times is high. Tsubone and Horikawa [29] were of the opinion that increasing routing flexibility always makes progress in the performance, despite the presence of family set-ups. Garavelli [7] believes that this holds especially for high shop loads. Diallo et al. [5] expresses that routing flexibility allows the machines usage factor and the throughput time to be improved. The aforementioned authors seem to unite in that little investment in routing flexibility results in significant performance improvements.

Since we have developed a few meta-heuristic-based heuristics in this research, it is good to point that the application of meta-heuristic algorithms to the design and manufacturing problems is growing in the recent years. Yildiz [31], Yildiz [32], and Yildiz [33] are just three sample researched in this regard. Since the current problem in this research deals with routing and material handling, vehicle design and scheduling can be added to the problem. The addressed heuristics are also applicable to the vehicle design problems using the research by Yildiz and Solanki [34]. According to the published researches, scheduling is an NP-hard problem. As a result, researchers have tried to develop heuristic and meta-heuristic algorithms for cell scheduling problems (such as [11, 17, 25–28]). The reader can refer to Ponnambalam [21] for an exhaustive review.

### 3 Problem description and formulation

#### 3.1 Problem description

A cellular manufacturing problem including $n$ parts ($i=1,\ldots,n$) and $m$ machines ($j=1,\ldots,m$) is considered. A number of predetermined processes should be done on each part. Notation $p$ represents the processes ($p=1,2,\ldots,P_i$) and $P_i$
represents the number of processes to be done on part \( i \). Operations required for manufacturing parts are determined in the production planning. Based on the similarities between production processes, parts are divided into \( K \) part families (\( k = 1, \ldots, K \)). Each part family requires a common setup time on each machine. In addition to a specific setup time for each part family, each part may need a separate setup time which is assumed to be considered in the corresponding process time. The objective function is to minimize the total processing times of all parts.

3.2 Assumptions

- A common setup time is required for part families on each machine.
- Setup time for each individual part, when needed, is considered as a part of the corresponding process time.
- Some processes for a particular part can be done on more than one machine; this is the concept of the routing flexibility.
- It is necessary to process all parts of a part family on a particular machine without any interruption from other part families.

3.3 Parameters and notations

- \( A_{ij} \): part-machine incident matrix (PMIM), \( \forall i = 1, 2, \ldots, n \forall j = 1, 2, \ldots, m \)
- \( D_{ij} \): distance matrix between machines, \( \forall j, j' = 1, \ldots, m \)
- \( H_i \): the required travel time per distance unit of part \( \forall i = 1, 2, \ldots, n \)
- \( T_{ij} \): processing time of part \( i \) on machine \( j \), \( \forall i = 1, 2, \ldots, n \forall j = 1, 2, \ldots, m \)
- \( PC_k \): Part–Cell matrix = \[ \begin{cases} 1 & \text{if part } i \text{ belongs to the part family } k \\ 0 & \text{otherwise} \end{cases} \] \( \forall i = 1, 2, \ldots, n \forall k = 1, 2, \ldots, K \)
- \( S_{kj} \): set-up time of family \( k \) on machine \( j \), \( \forall j = 1, 2, \ldots, m \forall k = 1, 2, \ldots, K \)

3.4 Decision variables

- \( x_{ijp} \) = \begin{cases} 1 & \text{if the process } p \text{ of part } i \text{ is done by machine } j \\ 0 & \text{otherwise} \end{cases} \)
- \( Z_{dij} \) = \begin{cases} 1 & \text{if the part } i \text{ is processed by machine } j \text{ just before part } i \\ 0 & \text{otherwise} \end{cases} \)
- \( C_{ip} \) = completion time of processes \( p \) of part \( i \)
- \( R_{ij} \) = \begin{cases} 1 & \text{if part } i \text{ requires machine } j \\ 0 & \text{otherwise} \end{cases} \)
- \( G \) = The variable that indicates makespan

3.5 Mathematical model

\[
\text{Min } G
\]

\[
\sum_{j=1}^{m} x_{ijp} = 1 \quad \forall i, p : i = 1, 2, \ldots, n \quad p = 1, 2, \ldots, P_i
\]  \( \text{(2)} \)

\[
\sum_{p=1}^{P_i} x_{ijp} \leq 1 \quad \forall i, j : i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m
\]  \( \text{(3)} \)

\[
C_{ij} \leq \left( \sum_{j=1}^{n} T_{ij} \times x_{ijp} \right) + \left( \sum_{k=1}^{K} \sum_{j=1}^{m} x_{ijp} \times S_{kj} \times (1-PC_k) \times S_k \right) + \left( \sum_{k=1}^{K} \sum_{j=1}^{m} x_{ijp} \times PC_k \times S_k \times \left( \frac{1}{\sum_{j=1}^{m} Z_{dij}} \right) \right) \quad \forall i, j : i = 1, 2, \ldots, n
\]  \( \text{(4)} \)

\[
C_{ip} \geq \left( \sum_{j=1}^{n} T_{ij} \times x_{ijp} \right) + \left( \sum_{k=1}^{K} \sum_{j=1}^{m} x_{ijp} \times S_{kj} \times (1-PC_k) \times S_k \right) + \left( \sum_{k=1}^{K} \sum_{j=1}^{m} x_{ijp} \times PC_k \times S_k \times \left( \frac{1}{\sum_{j=1}^{m} Z_{dij}} \right) \right) \quad \forall i, p : i = 1, 2, \ldots, n \quad p = 1, 2, \ldots, P_i
\]  \( \text{(5)} \)

\[
Z_{dij} \leq \sum_{p=1}^{P_i} x_{ijp} \times x_{ijp'} \quad \forall i, j, j' : i, i' = 1, 2, \ldots, n
\]  \( \text{(7)} \)

\[
\sum_{j=1}^{m} P \times x_{ijp} = A_{ij} \times R_{ij} \quad \forall i, j : i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m
\]  \( \text{(8)} \)

\[
G = \text{Max}\{ C_{ip} \} \quad \forall i, p : i = 1, 2, \ldots, n \quad p = 1, 2, \ldots, P
\]  \( \text{(9)} \)
The objective function in (1) represents the makespan given by constraint (9) in which \( G \) represents the maximum of the completion times of all processes of all parts. Constraint (2) ensures that the process \( p \) of part \( i \) is done on exactly one machine. As constraint (3), at most one process, of part \( i \) can be done on machine \( j \).

Constraints (4), (5), and (6) provide lower bounds for makespans. The first process of part \( i \) is the trigger of its total makespan. From the second process onwards, the makespan of the former process of part \( i \), traveling time between machines, and the waiting time for the idle machine affect the total makespan. Consequently, constraint (4), which is just written for the first process of part \( i \), only considers the process and setup times. The first term represents the sum of process time of part \( i \); the second term is the setup time added to the first term if parts \( i \) and \( i' \), the previous part on machine \( j \), do not belong to the same part family. The third term is added as a setup time if part \( i \) is the first part of its family which is processed on machine \( j \).

Constraint (5), which is written for all processes of the parts except for process 1, takes into account the traveling time between machines by term four and makespan of the former process of part \( i \) by term 5 as well as the explained three terms in constraint (4) for the first process of each part. If part \( i \) reaches machine \( j \) while it is in use by another part, it should wait. This issue is considered in the last term of constraint (6). In fact, the makespans of part \( i \) must be greater than the maximum of the values given by constraints (5) and (6) from the second process onwards. If part \( i \) reaches machine \( j \) while it is in use, the value in constraint (6) is greater than that of constraints (5), while, when machine \( j \) is free and is immediately used by part \( i \), the value in constraint (5) is greater than that of constraint (6).

Constraint (7) indicates that part \( i \) can be the former part of part \( i \) on machine \( j \), if and only if each of part \( i \) and part \( i \) has a process on machine \( j \). Constraint (8) satisfies the routing flexibility assumption and restricts the machines that can be used for each process based on the PMIM matrix. Constraint (10) illustrates the simple fact that there is no former part for the first part on machine \( j \). The fact that a particular machine cannot operate on more than one part simultaneously is guaranteed by constraints (11) and (12). Constraint (11) guarantees at most one part can be processed after part \( i \) on the same machine, while constraint (12) ensures that not more than one part can be operated just before part \( i \).

### 3.6 Numerical example

A clear-cut numerical example is presented in this section in order to clarify the performance of the proposed model.

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} Z_{ij} = \left( \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \right)^{-1} \quad \forall j : j = 1, 2, \ldots, m \tag{10}
\]

\[
\sum_{i=1}^{n} Z_{ij} \leq 1 \quad \forall i, j : i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m \tag{11}
\]

\[
\sum_{i=1}^{n} Z_{ij} \leq 1 \quad \forall i, j : i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m \tag{12}
\]

\[
x_{ijp}, Z_{ij} \in \{0, 1\} \quad \forall i, j, p : i = 1, \ldots, n \quad j = 1, 2, \ldots, m \quad p = 1, 2, \ldots, P
\]

\[
A_{ij} = \begin{bmatrix}
1 & 2 & 2 & 0 \\
3 & 1 & 2 & 3 \\
0 & 2 & 1 & 0
\end{bmatrix}
\]

Obviously, the matrix illustrates that part 1 needs two processes; the first process can be done only on machine 1, while the second one can be done on both machines 2 and 3. Other parts can be interpreted the same way.

- An example of the matrix PC\(_{ik}\) is illustrated in the following:

\[
PC_{ik} = \begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1
\end{bmatrix}
\]

In this example, three parts are divided into two part families. Parts 1 and 2 form one family while part 3 forms the other family. Apparently, it is a zero-one matrix with only one in each line.
Imagine a CMS problem with four parts, four machines, and with two cells. Input parameters are given as follows:

\[
PC_{ik} = \begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1 \\
0 & 1 \\
\end{bmatrix}, \quad D_{jj} = \begin{bmatrix}
1 & 0 & 5 & 5 \\
5 & 0 & 5 & 5 \\
5 & 5 & 0 & 5 \\
5 & 5 & 5 & 0 \\
\end{bmatrix},
\]

\[
A_{ij} = \begin{bmatrix}
1 & 2 & 0 & 1 \\
1 & 2 & 0 & 0 \\
0 & 0 & 1 & 2 \\
0 & 0 & 1 & 2 \\
\end{bmatrix}, \quad T_{ij} = \begin{bmatrix}
5 & 5 & 0 & 5 \\
5 & 5 & 0 & 0 \\
0 & 0 & 5 & 5 \\
0 & 0 & 5 & 5 \\
\end{bmatrix},
\]

\[
S_{jk} = \begin{bmatrix}
2 & 2 \\
2 & 2 \\
2 & 2 \\
2 & 2 \\
\end{bmatrix}, \quad H_i = \begin{bmatrix}
1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

We utilize Lingo solver to solve this example using B&B method. The results are given in Table 1.

The first part family consists of parts 1 and 2, while parts 3 and 4 form the second part family. Figure 1 depicts the final solution schematically. It indicates operational sequences with time consideration. The left side number in the container represents the number of the part, and the right side number represents the number of process done on the part. In addition, movements are drawn using the spiral arrows for the parts. As it is clear from the figure, the first process of parts 2 and 4 on machines 1 and 3 are the starter of the whole process. It takes two units for set-up time and five more units for the process time, so the first processes of these two parts are completed at the seventh minute. Part 2 is then moved to the second machine; considering the handling, setup, and processing time, the second operation of this part is completed at 19th minute. Totally, 24 min is required to complete all the processes of all the parts.

3.7 Optimization of nonlinear problem

As it was mentioned in Section 3.1, the presented nonlinear model is solved using Lingo solver on a PC with 2 GHz processor and 2 GB RAM for different sizes of problems. Table 2 gives the size of the problems and the corresponding CPU runtime.

In Table 2, NP, NM, and NC represent the number of parts, number of machine, and the number of part families, respectively. Since the solver uses B&B method, \( F_{\text{best}} \) represents the best feasible objective function value while \( F_{\text{bound}} \) represents the lower bound of the objective function value. The final solution is optimum if \( F_{\text{bound}} \) equals to \( F_{\text{best}} \). We stop the solver when the computation time reaches 14,400 s. The solver is capable to find the optimal solution for the first two problems. As the size of the problem increases, the solver presents a feasible solution and a lower bound for the problem in a reasonable processing time (the third problem). In the fourth problem, the solver is not even capable of finding a feasible solution within 4 h, although this problem is considered a small-size problem.

4 Linearization and analysis

4.1 Linearization of the model

The unsatisfactory performance of the aforementioned solver regarding bigger sizes of the problem motivated us to modify
the structure of the problem in such a way as to implement more efficient optimization tools. Some terms in the constraints are nonlinear which cause intricacy in the model. In this section, the linearization of the mathematical model using some new variables is presented.

**Proposition 1:** the term \( \sum_k \sum_j j \neq j \sum_j x_{ijp} \times PC_{ik} \times (1-PC_{jk}) \times S_{jk} \) can be linearized by the following terms:

\[
\sum_k \sum_j j \neq j \sum_j x_{ijp} \times PC_{ik} \times (1-PC_{jk}) \times S_{jk} \tag{14}
\]

\[
st : V_{ijp} \geq x_{ijp} + z_{d,j} - 1 \tag{15}
\]

\[
V_{ijp} \geq 0 \tag{16}
\]

**Proof** Consider the following statements:

1. \( x_{ijp} = 1 \) and \( z_{d,j} = 1 \)
2. \( x_{ijp} = 0 \) and \( z_{d,j} = 1 \)
3. \( x_{ijp} = 1 \) and \( z_{d,j} = 0 \)
4. \( x_{ijp} = 0 \) and \( z_{d,j} = 0 \)

**Table 2** The objective function and CPU time results for the nonlinear model

| Pro. no. | Problem size | Results | \( f^{\text{bound}} \) | \( f^{\text{best}} \) | Run time (s) |
|----------|--------------|---------|----------------|----------------|-------------|
| 1        | 4 \times 4 \times 2 | 51 | 51 | 7 |
| 2        | 6 \times 4 \times 2 | 96 | 96 | 1,134 |
| 3        | 7 \times 6 \times 2 | 121.44 | 179 | 14,400 |
| 4        | 8 \times 8 \times 2 | – | – | 14,400 |

In the first statement, \( V_{ijp} \geq 1 \). In as much as this is a minimizing problem and \( PC_{ik} \times (1-PC_{jk}) \times S_{jk} \geq 0 \), \( V_{ijp} \) will have the least possible value which is 1; therefore nothing will change in the optimal solution.

In the other cases, \( V_{ijp} \geq 0 \) and with the same logic \( V_{ijp} \) would be equal to zero and the optimal solution will not change.

The same explanation can be given for exchanging the following proposition for linear terms:

**Proposition 2:** \( \sum_k \sum_j j \neq j H_i \times D_j \times x_{ij \text{ } p-1}, x_{ijp} \) can be replaced with

\[
\sum_k \sum_j j \neq j H_i \times D_j \times Y_{ij, \text{ } p-1} \tag{17}
\]

\[
st : Y_{ij, \text{ } p-1} \geq x_{ij \text{ } p-1} + x_{ijp} - 1 \tag{18}
\]

\[
Y_{ij, \text{ } p-1} \geq 0 \tag{19}
\]

**Proposition 3:** term \( \sum_k \sum_j j \neq j x_{ijp} \times PC_{ik} \times S_{jk} \times (1-\sum_i \neq i z_{ii,j}) \) can also be written as follows:

\[
\sum_k \sum_j j \neq j x_{ijp} \times PC_{ik} \times S_{jk} \times (1-\sum_i \neq i z_{ii,j}) \tag{20}
\]

The second nonlinear term can be linearized as follows:

\[
\sum_k \sum_j j \neq j B_{ijp} \times PC_{ik} \times S_{jk} \tag{21}
\]

\[
st : B_{ijp} \leq \frac{x_{ijp}}{2} + \frac{z_{d,j}}{2} \tag{22}
\]

\( B_{ijp} \in \{0, 1\} \) \tag{23}

**Proof** The following cases are possible:

1. \( x_{ijp} = 1 \) and \( z_{d,j} = 1 \)
2. \( x_{ijp} = 1 \) and \( z_{d,j} = 0 \)
3. \( x_{ijp} = 0 \) and \( z_{d,j} = 1 \)
4. \( x_{ijp} = 0 \) and \( z_{d,j} = 0 \)

In the first statement, \( B_{ijp} \leq 1 \). Since the second nonlinear term has a negative sign and the problem is of minimizing, thus \( B_{ijp} \) will have the maximum possible quantity which is 1. For the other cases, either \( B_{ijp} \leq \frac{1}{2} \) or \( B_{ijp} \leq 0 \), \( B_{ijp} \) can only have zero value because it is a binary variable.
Proposition 4: \( \sum_{j} \sum_{i \neq j} p' x_{ijp} \times z_{ij} \times C_{ip} \times x_{ijp'} \) can be linearized by the following terms:

\[
\sum j \sum i \neq i \sum p' U_{i'jpp'}
\]  

(24)

\[ st : U_{i'jpp'} \geq C_{i'p} - M \left( 1 - V_{i'jp} \right) - M \left( 1 - x_{ijp'} \right) \]  

(25)

\[ V_{i'jp} \geq x_{ijp} + z_{ij} - 1 \]  

(26)

\[ V_{i'jp} \geq 0 \]  

(27)

\[ U_{i'jpp'} \geq 0 \]  

(28)

\[ V_{i'jp} \geq 0 \]  

(29)

\[ st : V_{i'jp} \geq x_{ijp} + z_{ij} - 1 \]  

(30)

\[ V_{i'jp} \geq 0 \]  

(31)

Proof As we have hypothesized in proposition 1:

\[ x_{ijp} \times z_{ij} = V_{i'jp} \]  

(32)

\[ st : V_{i'jp} \geq x_{ijp} + z_{ij} - 1 \]  

(33)

\[ V_{i'jp} \geq 0 \]  

(34)

Owing to this, the nonlinear term in this section will turn into the following terms:

\[
\sum j \sum i \neq i \sum p' \left( V_{i'jp} \times C_{i'p} \times x_{ijp'} \right)
\]  

(35)

Consider the following six statements below:

1. \( x_{ijp} = 1 \) and \( z_{ij} = 1 \) and \( x_{ijp'} = 1 \)
2. \( x_{ijp} = 1 \) and \( z_{ij} = 1 \) and \( x_{ijp'} = 0 \)
3. \( x_{ijp} = 1 \) and \( z_{ij} = 0 \) and \( x_{ijp'} = 0 \)
4. \( x_{ijp} = 0 \) and \( z_{ij} = 0 \) and \( x_{ijp'} = 0 \)
5. \( x_{ijp} = 0 \) and \( z_{ij} = 1 \) and \( x_{ijp'} = 1 \)
6. \( x_{ijp} = 1 \) and \( z_{ij} = 0 \) and \( x_{ijp'} = 1 \)

As the first statement, \( V_{i'jp} = 1 \) and \( x_{ijp'} = 1 \). As a result, \( U_{i'jpp'} \geq C_{i'p} \). Due to the minimizing nature of the model, \( U_{i'jpp'} \) will have the least value, which is \( C_{i'p} \).

In the other cases, \( U_{i'jpp'} \geq -M \). \( M \) stands for a very big positive number. Since the model is of minimizing and \( U_{i'jpp'} \geq 0 \), \( U_{i'jpp'} \) will get zero value.

Proposition 5: \( z_{ij} \geq \sum p' = 1 \sum p' = 1 \times x_{ijp} \times x_{ijp'} \) can be linearized by the following terms:

\[
z_{ij} = \frac{\sum_{p=1} x_{ijp}}{2} + \frac{\sum_{p'} x_{ijp'}}{2}
\]  

(36)

\[ st : U_{i'jpp'} \geq 0 \]  

(37)

Proof According to (37), \( z_{ij} \) can only be equal to 1 when both \( x_{ijp} \) and \( x_{ijp'} \) are equal to 1. On the other hand, according to the rest of the constraints, \( z_{ij} \) would never be equal to 0 when both \( x_{ijp} \) and \( x_{ijp'} \) are equal to 1.

Proposition 6: \( G = \text{Max} \{ C_{ip} \} \) can be linearized by replacing it with the following set of constraints:

\[ G \geq C_{ip} \quad \forall i, p \]  

(38)

Proof According to (38), \( G \) is greater than all the value of \( C_{ip} \), for every \( i \) and \( p \). Since the problem has a minimization form, \( G \) is equal to the maximal value of completion times.

4.2 Optimization of the linear model

Linearization of the model brings up the possibility of implementing more specialized tools for integer linear programming. In this section, we solve a set of problems with
Cplex and the results are demonstrated in Table 3. The same PC as expressed in Section 3.2 is used to solve the problems. As given by Table 3, we have been able to find the optimum solution for a problem with 25 machines and 35 parts which are divided into 6 part families (problem no 10). Regarding the optimization results of the nonlinear model, this is a huge improvement in term of CPU run time. For the 11th problem, the Cplex was unable to find the optimum solution within 14,400 s.

5 Solution algorithms

Clearly, the considered problem is NP-hard. In Section 4, we tried to modify the problem in such a way as to find the optimal solution for bigger sizes of the problem. However, due to the NP-hard nature of the problem, solving the big sizes of the problem is still impossible within a reasonable time. Therefore, heuristic algorithms are applied for this situation. Venkataramanaiah [30] indicated the promising performance of SA in CMS scheduling problems. Lin et al. [16] indicated the superiority of SA over GA and TS in this context. These researches motivated us to select SA algorithm from among all the existing options in order to solve the under study CMS scheduling problem.

5.1 Simulated annealing

SA is an imitation of the annealing process, in which a material is heated and then allowed to cool very slowly until it reaches its most regular possible crystalline state with corresponding minimum energy [12]. Metropolis et al. [18] was the first research, which suggested the structure of this algorithm. In optimization, SA was used by Kirkpatrick et al. [15] for the first time [12]. The key feature of SA is that it provides a condition by which we can escape local optima by allowing hill climbing moves (i.e., moves which worsen the objective function value) in the hope of finding a global optimum [12]. Kirkpatrick et al. [15] indicated that the Metropolis algorithm could be applied to discrete optimization problems by defining feasible solutions as states, and using an objective function to represent the change in energy between states.

5.2 Implementation of SA

In this section, the utilization of SA in order to solve the current problem is explained. Figure 2 depicts the different stages of the designed SA-based heuristic.

Different steps are explained as follows:

5.2.1 Initial solution generation

At the first step, the structure of the solution should be defined. As it is depicted in Fig. 3, an $n \times m$ matrix is proposed to represent each solution of the problem.

$$
\begin{pmatrix}
{l_{11}} & l_{12} & \cdots & l_{1m} \\
l_{21} & l_{22} & \cdots & l_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
l_{n1} & l_{n2} & \cdots & l_{nm}
\end{pmatrix}
$$
In this structure, \( l_{ij} \) represents the number of the part whose process is supposed to be done on machine \( j \) with order \( s \). For instance, if \( l_{23} \) is equal to 4, it means that the second part whose process is to be done on machine 3 is part 4. Two methods are proposed in order to generate the solution as follows:

- Random generation: It is a simple method in which the initial solution is generated randomly.
- Using shortest processing time (SPT) method: Previous researches reveal that using a proper initial solution can improve the performance of a meta-heuristic algorithm; consequently, SPT method was applied to generate a convenient initial solution for the SA-based heuristic. According to SPT, the total sum of processing times of each part family is calculated initially; then, part families are ranked in ascending order based on their processing times. Part families’ ranking determines the sequence of processing. The part family with the shortest process time will be of higher priority. The sequence of the parts in each part family is also determined in the similar way, in which the sum of processing time of each part is computed and they are ranked in an ascending order.

5.2.2 Modification of the solutions

Swapping operator is applied in order to find the optimum solution. As this operator, the sequences of two

\[
\begin{pmatrix}
5 & 4 & 1 & 2 \\
3 & 5 & 3 & 6 \\
1 & 2 & 4 & 4 \\
2 & 1 & 5 & 3 \\
6 & 6 & 2 & 1 \\
4 & 3 & 6 & 5
\end{pmatrix}
\]

Fig. 4 Initial sequence of six parts on four machines parts, which are selected randomly, are being exchanged on each machine. For instance, suppose that the sequence of six parts on four machines is given as in Fig. 4.

Exchanging the sequence of parts 5 and 6 on the first machine implies the modified solution as given by Fig. 5.

5.2.3 Replacement of the solutions

Two conditions are set to replace the new modified solution with the initial one: (a) if the modified solution improves the objective function value, it will be replaced with the initial solution; (b) a fraction of nonimproving solutions are accepted in the hope of escaping local optimum while searching global optimum. The probability of accepting nonimproving solutions is calculated as the Cauchy function instead of that of the Boltzmann, which can reduce the probability of trapping in local optimum [16]. The probability of replacing the initial solution \( P \) is given by Eq. (37) noting that \( \Delta F \) is equal to the gap between the objective function’s values of the modified and the initial solutions and \( T \) is the temperature of the existing iteration.

\[
P = \frac{T}{T^2 + \Delta F^2}
\]  

(37)

5.2.4 Initial temperature and stopping rule

Some statistical tests have been developed in order to select a proper value for the initial temperature and cooling rate expressed in Section 7.1. The temperature decreases using the cooling rate so that the new temperatures are found through multiplying it by the current temperature. The algorithm stops if the current temperature reaches to a predetermined value.

| Parameter | Value |
|-----------|-------|
| \( S_{jk} \) | \( U(10, 100) \) |
| \( T_{ij} \) | \( U(10, 50) \) |

Table 4 The SA Parameters’ values

| Parameters     | Level1 | Level2 | Level3 |
|----------------|--------|--------|--------|
| Cooling rate   | 0.97   | 0.98   | 0.985  |
| Initial temperature | 100   | 150   | 200    |

Table 5 The parameters values of the designed problems
6 Computational results

6.1 Tuning SA parameters

The cooling rate and the initial temperature are the two parameters of the presented heuristic to be tuned. Experimentally, we have considered three levels for these parameters. The values selected for each parameter are given as in Table 4.

Table 4: The parameters to be tuned

| Algorithm Parameters | Levels of experiment | Considered criterion | P value |
|----------------------|----------------------|----------------------|---------|
| Cooling rate         | 3 × 1                | Quality of solution  | 0.000   |
|                      | 3 × 2                | Quality of solution  | 0.000   |
|                      | 1 × 3                | CPU runtime          | 0.000   |
|                      | 2 × 3                | CPU runtime          | 0.000   |
| Initial temperature  | 1 × 3                | Quality of solution  | 0.810   |
|                      | 1 × 2                | Quality of solution  | 0.291   |
|                      | 1 × 3                | CPU runtime          | 0.001   |
|                      | 1 × 2                | CPU runtime          | 0.002   |

Ten numerical problems are designed as given by Table 5. The given parameters are stochastic with given probability distribution functions. Each problem is run for 100 times. Factorial tests with \( a = 0.05 \) are implemented. The quality of the solution and the CPU run time are the two criteria, which are considered for evaluating the performance of the proposed heuristics. In each condition, the null hypothesis is the equality of the two comparing levels for the mentioned parameters.

Table 5: Computational results of the SA-based heuristics and CPLEX (using B&B)

| Pro. no. | Problem size | CPLEX | SA | Hybrid SA with SPT |
|----------|--------------|-------|----|-------------------|
|          | NP × NM × NC | Opt. Sol. | Run time (s) | Zmean | Zbest | SD | Run time (s) | Zmean | Zbest | SD | Run time (s) |
| Small    |              |        |     |                   |       |     |    |                   |       |     |    |                   |
| 1        | 4 × 4 × 2    | 51     | 0.14 | 51                | 51    | 0   | 0.87 | 51                | 51    | 0   | 1.07 |
| 2        | 6 × 4 × 2    | 96     | 0.23 | 96                | 96    | 0   | 1.25 | 96                | 96    | 0   | 1.40 |
| 3        | 7 × 6 × 2    | 151    | 0.36 | 151.64            | 151   | 3.51| 2.19 | 151.64            | 151   | 3.51| 2.19 |
| 4        | 8 × 8 × 2    | 126    | 0.44 | 126.52            | 126   | 3.04| 2.75 | 126.52            | 126   | 3.04| 2.75 |
| 5        | 12 × 10 × 3  | 190    | 19.59| 209.65            | 190   | 20.6 | 5.02 | 209.65            | 190   | 20.6 | 5.02 |
| Average  | 122.80       | 4.15   | 126.96| 122.80            | 122.80| 5.43| 2.42 | 126.96            | 122.80| 5.43| 2.42 |
| Medium   |              |        |     |                   |       |     |    |                   |       |     |    |                   |
| 6        | 12 × 12 × 3  | 196    | 17.27| 197.90            | 196   | 6.16| 6.86 | 197.90            | 196   | 6.16| 6.86 |
| 7        | 15 × 12 × 4  | 148    | 232.04| 181.01            | 148   | 20.09| 6.87 | 181.01            | 148   | 20.09| 6.87 |
| 8        | 20 × 15 × 5  | 129    | 1,002.81| 175.02   | 170   | 15.55| 9.15 | 175.02   | 170   | 15.55| 9.15 |
| 9        | 30 × 20 × 6  | 198    | 1,011.75| 253.90   | 231   | 27.69| 19.49 | 253.90   | 231   | 27.69| 19.49 |
| 10       | 35 × 25 × 6  | 203    | 967.07| 203.90   | 203   | 5.07 | 21.75 | 203.90   | 203   | 5.07 | 21.75 |
| Average  | 174.8        | 646.19 | 202.35| 189.6   | 14.91 | 12.82| 201.37| 189.6   | 14.91| 12.82| 201.37|
| Large    |              |        |     |                   |       |     |    |                   |       |     |    |                   |
| 11       | 35 × 30 × 7  | –      | 14,400| 241.54          | 215   | 22.77| 29.40 | 241.54          | 215   | 22.77| 29.40 |
| 12       | 40 × 30 × 7  | –      | –     | 245.35          | 166   | 51.65| 35.30 | 245.35          | 166   | 51.65| 35.30 |
| 13       | 35 × 35 × 8  | –      | –     | 203.07          | 144   | 32.60| 31.49 | 203.07          | 144   | 32.60| 31.49 |
| 14       | 40 × 35 × 8  | –      | –     | 273.75          | 206   | 27.84| 35.80 | 273.75          | 206   | 27.84| 35.80 |
| 15       | 43 × 40 × 10 | –      | –     | 227.16          | 161   | 33.32| 43.67 | 227.16          | 161   | 33.32| 43.67 |
| 16       | 45 × 45 × 5  | –      | –     | 253.04          | 218   | 26.10| 46.57 | 253.04          | 218   | 26.10| 46.57 |
| 17       | 60 × 35 × 7  | –      | –     | 405.32          | 239   | 42.92| 53.97 | 405.32          | 239   | 42.92| 53.97 |
| 18       | 65 × 40 × 8  | –      | –     | 300.22          | 276   | 43.19| 61.14 | 300.22          | 276   | 43.19| 61.14 |
| 19       | 70 × 45 × 8  | –      | –     | 397.39          | 228   | 121.04| 75.31 | 397.39          | 228   | 121.04| 75.31 |
| 20       | 80 × 40 × 8  | –      | –     | 407.26          | 383   | 35.74| 75.69 | 407.26          | 383   | 35.74| 75.69 |
| Average  | 295.41       | 223.60 | 43.72 | 48.83 | 307.24 | 211.4 | 43.40 | 50.74 |
| Comprehensive average | 208.24 | 178.67 | 21.35 | 21.36 | 211.61 | 171.3 | 20.74 | 22.03 |

Fig. 6 Zmean comparison of SA and hybrid SA with SPT
criteria. If the $P$ value $<$ $a$, the null hypothesis is not accepted. Results are illustrated by Table 6. Based on the quality of the solution and the CPU run time, level three (0.985) and level one (100) are more appropriate for the cooling rate and the initial temperature, respectively.

6.2 Measuring the robustness

Twenty problems with varying sizes have been designed in order to measure the robustness of the proposed heuristics. According to the number of parts (NP), number of machines (NM), and number of cells (NC), the numerical problems are divided into three categories: small, medium, and large sizes. Two types of SAs are implemented, which differ on their initial solution generation procedure. The initial solution is generated randomly for the first type while for the second type, the SPT method is used. The comparison among the solution obtained from CPLEX and the two types of SAs is given in Table 5. Each problem was run for 100 times. $Z_{\text{mean}}$, which represents the average of the objective function values, are reported in Table 7 along with the average of the run times. $Z_{\text{best}}$ is the best value found for the objective function, and SD represents the standard deviation of the obtained solutions (from 100 runs).

CPLEX is able to find the optimum solution for small- and medium-size problems. In small-size problems, the SA-based heuristics are capable of finding the optimum solution. SA heuristics also show a promising performance for the medium-size problems. There is an almost 3 % gap between the optimum solution and the best solution of the hybrid SA, while this gap for the non-hybrid SA is almost 8 %. Furthermore, there is a huge difference between the run time of CPLEX and SA heuristics.

In term of the quality of solution, hybridizing SA with the SPT algorithm can improve the quality of solution. The best solutions of the hybrid SA with SPT are more than 4 % better than the normal SA on average. The SA with SPT also preserves its superiority in term of standard deviation of the solutions. In terms of the CPU runtime, SA has an advantage over SA with SPT. Figures 6, 7, 8, and 9 depict comparisons between SA and hybrid SA with SPT performances.

7 Conclusion and future research

Cell scheduling problem with routing flexibility and handling times was considered in this paper. A mathematical model was developed to minimize the makespan with sequence-dependent family setup times. The optimum solutions are found even for bigger sizes of the problem converting the initial nonlinear model to a linear form; furthermore, the results in term of CPU runtime are improved. Unfortunately, linearization cannot solve the problem of high CPU runtime for very large problems; therefore, based on the previous researches, we developed a SA-based heuristic in order to solve the problem. The performance of the initial heuristic can be improved hybridizing it with SPT algorithm. The results for a number of small-, medium-, and large-size problems indicate that the heuristics have a promising performance.

Improvement of the structure of the model from both optimization and applicability aspects can be a good opportunity for the future researches. We believe there is a possibility to define the variables and constraints in a different way that can smooth the optimization process. Different assumptions
can be added to the model that are compatible with the real-world phenomena. More efficient algorithms can be developed to find better solution in a more reasonable time.

References

1. Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187(3):985–1032
2. Bouabdela R, Jarboui B, Eddaly M, Rebai A (2011) A branch and bound enhanced genetic algorithm for scheduling a flowline manufacturing cell with sequence dependent family setup times. Comput Oper Res 38:387–393
3. Chalapathi Pasupuleti V (2012) Scheduling in cellular manufacturing systems. Iberoam J Ind Eng 4(7):231–243
4. Das SR, Canel C (2005) An algorithm for scheduling batches of parts in a multi-cell flexible manufacturing system. Int J Prod Econ 97:247–262
5. Diao M, Pierreval H, Quilliot A (2001) Manufacturing cells design with flexible routing capability in presence of unreliable machines. Int J Prod Econ 74:175–182
6. França PM, Gupta JND, Mendes AS, Moscato P, Veltink KJ (2005) Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. Comput Ind Eng 48(3):491–506
7. Garavelli AC (2001) Performance analysis of a batch production system with limited flexibility. Int J Prod Econ 69:39–48
8. Gholipour-Kanani Y, Tavakkoli-Moghaddam R, Khorrami A (2011) Solving a multi criteria group scheduling problem for a cellular manufacturing system by scatter search. J Chin Ins Ind Eng 28:192–205
9. Hendizadeh SH, Faramarzi H, Mansouri SA, Gupta JND, ElMekkawy TY (2008) Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. Int J Prod Econ 111(2):593–605
10. Jensen JB, Malhotra MK, Philipoom PR (1996) Machine dedication and process flexibility in a group technology environment. J Oper Manag 14:19–39
11. Jeon G, Leep HR (2006) Forming part families by using genetic algorithm and designing machine cells under demand changes. Comput Oper Res 33:263–283
12. Johnson AW, Henderson D, Jacobson SH (2003) The theory and practice of simulated annealing. In: Glover FW, Kochenberger GA (eds) Handbook of metaheuristics, USA. Kluwer Academic Publishers, Massachusetts, pp 287–316
13. Kesen SE, Das SK, Gungor Z (2010) A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (VMCs). Comp Oper Res 37:1148–1156
14. Kioon S (2009) Integrated cellular manufacturing system design with production planning and dynamic reconfiguration. Eur J Oper Res 192:414–428
15. Kirkpatrick S, Gelatt JCD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
16. Lin SW, Ying KC, Lee ZJ (2009) Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times. Comp Oper Res 36:1110–1121
17. Logendran R, Nuclonasboon N (1991) Minimizing the makespan of a group scheduling problem: a new heuristic. Int J Prod Econ 22:217–230
18. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
19. Nomden G, Van der Zee DJ (2008) Virtual cellular manufacturing: configuring routing flexibility. Int J Prod Econ 112:439–451
20. Parthasarthathy S, Rajendran C (1998) Scheduling to minimize mean tardiness and weighted mean tardiness in flowshop and flowline-based manufacturing cell. Comput Ind Eng 34(2):531–546
21. Ponnambalam SG, Aravindan P, Reddy KRR (1999) Analysis of group-scheduling heuristics in a manufacturing cell. Int J Adv Manuf Technol 15:914–932
22. Rajendran C, Ziegler H (1999) Heuristics for scheduling in flowshops and flowline-based manufacturing cells to minimize the sum of weighted flowtime and weighted tardiness of jobs. Comput Ind Eng 37(4):671–690
23. Schaller JE, Gupta JND, Vakharia AJ (2000) Scheduling a flowline manufacturing cell with sequence dependent family setup times. Eur J Oper Res 125:324–339
24. Sheikhzadeh M, Benjaafar S, Gupta D (1998) Machine sharing in manufacturing systems: total flexibility versus chaining. Int J Flex Manuf Syst 10:351–378
25. Solimanpur M, Vrat P, Shankar R (2004) A heuristic to minimize makespan of cell scheduling problem. Int J Prod Econ 88:231–241
26. Sridhar J, Rajendran C (1994) A genetic algorithm for family and job scheduling in a flowline-based manufacturing cell. Comp Ind Eng 27:469–472
27. Sridhar J, Rajendran C (1996) Scheduling in flow shop and cellular manufacturing system with multiple objectives—a genetic algorithmic approach. Prod Plan Control 7:374–382
28. Suer GA, Vazquez R, Cortes M (2005) A hybrid approach of genetic algorithms and local optimizers in cell loading. Comp Ind Eng 48:625–641
29. Tsubone H, Horikawa M (1999) A comparison between machine flexibility and routing flexibility. Int J Flex Manuf Syst 11:83–101
30. Venkataramaniah S (2007) Scheduling in cellular manufacturing systems: a heuristic approach. Int J Prod Res 46:429–449
31. Yildiz AR (2009) A novel hybrid immune algorithm for global optimization in design and manufacturing. Robot Comput Integr Manuf 25:261–270
32. Yildiz AR (2009) A novel particle swarm optimization approach for product design and manufacturing. Int J Adv Manuf Technol 40:617–628
33. Yildiz AR (2009) An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry. J Mater Process Technol 209:2773–2780
34. Yildiz’ AR, Solanki KN (2012) Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach. Int J Adv Manuf Technol 59:367–376