A PERFORMANCE COMPARISON AND EVALUATION OF METAHEURISTICS FOR A BATCH SCHEDULING PROBLEM IN A MULTI-HYBRID CELL MANUFACTURING SYSTEM WITH SKILLED WORKFORCE ASSIGNMENT

OMER FARUK YILMAZ*
Department of Industrial Engineering
Yalova University
Yalova, Turkey

MEHMET BULENT DURMUSOGLU
Department of Industrial Engineering
Istanbul Technical University
Istanbul, Turkey

(Communicated by Ines Marques)

Abstract. This paper focuses on the batch scheduling problem in multi-hybrid cell manufacturing systems (MHCMS) in a dual-resource constrained (DRC) setting, considering skilled workforce assignment (SWA). This problem consists of finding the sequence of batches on each cell, the starting time of each batch, and assigning employees to the operations of batches in accordance with the desired objective. Because handling both the scheduling and assignment decisions simultaneously presents a challenging optimization problem, it is difficult to solve the formulated model, even for small-sized problem instances. Three metaheuristics are proposed to solve the batch scheduling problem, namely the genetic algorithm (GA), simulated annealing (SA) algorithm, and artificial bee colony (ABC) algorithm. A serial scheduling scheme (SSS) is introduced and employed in all metaheuristics to obtain a feasible schedule for each individual. The main aim of this study is to identify an effective metaheuristic for determining the scheduling and assignment decisions that minimize the average cell response time. Detailed computational experiments were conducted, based on real production data, to evaluate the performance of the metaheuristics. The experimental results show that the performance of the proposed ABC algorithm is superior to other metaheuristics for different levels of experimental factors determined for the number of batches and the employee flexibility.

1. Introduction. Cellular manufacturing systems (CMS) are effective productions systems for producing part families, which are composed of parts with similar processing requirements [58, 63, 62]. Through the utilization of CMS, the setup time, work-in-process inventories, material handling cost, and lead time are reduced, while the visibility and quality are enhanced [59, 60]. There are several activities to be
carried out in CMS before starting the production process. Scheduling is the final operational activity, and plays a key role in the smooth operation of CMS. Hence, finding good solutions to the scheduling problems is very important for the successful implementation of CMS.

Scheduling problems in cellular manufacturing system (CMS) are widely investigated in the academic literature to increase the operational efficiency in production. These problems can mainly be classified into two groups: those that consider a single cell and those that consider multiple cells [36].

The process for cell scheduling, known as group scheduling, can be divided into family and job scheduling stages. The family sequence and the job sequences at each family are determined simultaneously in order to compute the completion time for each job [50]. However, the effect of employee skills on the cell cycle times, employee timetabling, and the application of one-piece flow within cells are not considered in cell scheduling. For this reason, a comprehensive batch scheduling problem that integrates employee timetabling and cell scheduling by considering employee skills together with the application of one-piece flow within cells would be a valuable addition to the existing related literature.

In the current study, the batch scheduling problem in a multi-hybrid cell manufacturing system (MHCMS) is addressed, considering skilled workforce assignment (SWA), which refers to employee flexibility. The MHCMS is a type of CMS that consists of a number of parallel independent hybrid cells. Unlike automated and assembly cells, both automatic and manual operations are performed in these hybrid cells; hence the reason for calling the cells hybrid [28]. The importance of SWA for batch scheduling problems becomes especially clear for hybrid cells [62]. As long as the employee flexibility is concerned for these cells, it requires taking into account the different skill levels for the employees to perform the manual operations; thus, the SWA has a crucial effect on the MHCMS performance whenever the manual operations have to be performed in the cells [62, 28]. The reasons for this problem being referred to as the batch scheduling problem are explained in Section 2.

Production systems in a dual-resource constrained (DRC) setting are ones in which the capacity constraints on the output depend on the availability of both the machines and the employees [27]. Employees required for manual operations are considered to be valuable resources when there are fewer employees than the number of workstations at which manual operations are performed. This kind of cellular manufacturing system (CMS) is considered to be a DRC system, which is a well-established trend in the literature [16]. In this type of system, human operators are often constrained resources, and the movements of employees are both intercellular and intracellular [61, 49]. One of the most notable examples of CMS in a DRC setting in which employee resources directly affect the processing times of the batches are hybrid manufacturing cells, where both automatic and manual operations are performed [62, 28]. Furthermore, employee involvement is unrestricted in the hybrid cells, and the assignment of employees has a significant effect on the cell cycle times. Because of the manual operations within the hybrid cells, the cell cycle times vary significantly, and the batch processing times depend on the employees assigned to perform the operations on these batches [62]. Considering employees as the critical resources for both their availability and effect on the manual operation times makes it more important to research this problem. In real manufacturing environments, employees are not unlimited resources, and they cannot be treated equally regarding their skill levels.
In this study, the cells are arranged in a multi-product flow shop layout that applies one-piece flow, and the MHCMS has a DRC setting in such a manner that a batch can only be processed if employee resources and machine resources are available. This problem is composed of the scheduling and assignment decisions, where the sequence of batches, starting times of batches, and assignment of employees to cells for the batches should be determined simultaneously [62].

There are few studies in the academic literature relating to the batch scheduling problem. Das and Canel [19] developed a branch and bound technique to find a solution to the batch scheduling problem in a multi-cell flexible manufacturing system (MCFMS). Celano et al. [8] dealt with the batch scheduling problem for the multiple distinctive manufacturing cells using simulation method. Balaji and Porsevli [6] also developed an optimization model for the batch scheduling problem considering sequence dependent setup times. Yilmaz et al. [62] proposed a novel goal programming optimization model, in which the objectives were the total number of employees in the system and the average flow times, for the batch scheduling problem in the MHCMS. They did not consider limited employee resource as a constraint with SWA affecting the manual operation times on cells. In their study, the primary purpose was the examination of the interaction between the total number of employees and the flow times considering employee resources are unlimited. However, it does not provide any chance to evaluate the effects of different employee skill scenarios on the cell response times with limited employee resources. Therefore, we present a new optimization model to address these critical issues. The selected metaheuristiscs are implemented to solve the problem and compared to each other regarding the solution quality. The test problems were generated based on real production data, presented in the study of Yilmaz et al. [62], and used to investigate both the performance of metaheuristics and the effects of several factors on the average cell response time.

Most studies on batch scheduling have focused on static scheduling problems. In these types of problems, employee resources are accepted to be non-renewable, and they cannot be assigned to more than one operation. In addition, the number of employees in each cell remains unchanged during the scheduling horizon. On the other hand, the employee resources are assumed to be renewable for dynamic scheduling problems, so they can be assigned to several non-overlapping operations. Therefore, the number of employees in each cell varies dynamically throughout the scheduling horizon [21].

The current study focuses on the dynamic batch scheduling problem, considering employee resources to be limited. Employee (worker) allocation and assignment in CMS have been prevalently examined in the existing literature by Suer and Dagli [53], Cesani and Steudel [9], Suer and Tummaluri [54], Fan et al. [24], Suer and Alhawari [52], and Niakan et al. [42].

In recent years, employee flexibility in the CMS has been considered in some studies.

Jensen [29] used the simulation method to examine the performance advantages of the labor flexibility for the departments, the hybrid cells, and the strict cells. Askin and Huang [4] proposed a multi-objective model for the employee assignment and training process. Norman et al. [43] analyzed the problem of assigning employees to the cells while considering technical and human skills. Fowler et al. [25] proposed an optimization model by considering general cognitive ability (GCA) of employees. Davis et al. [20] analyzed employee flexibility with cross-training using simulation.
method. Azadeh et al. [5] examined the employee allocation problem with the learning effects in a CMS by integrating fuzzy data envelopment analysis (FDEA) and simulation method. Eglilmez et al. [22] dealt with the problem of stochastic skill-based workforce assignment in a CMS. Liu et al. [37] analyzed employee allocation in a CMS by considering the learning and the forgetting effects of employees. Costa et al. [17] investigated the effect of the assignment of the heterogeneous workers on both the workforce cost and the makespan in a work cell with a flow shop layout. They stated that allocation of skilled workers to perform setup tasks affects the cell responses. Costa et al. [15] investigated the effect of limited human resources having different skill levels on the performance of a parallel machine production system. Three scenarios, namely high-skill (HS), average-skill (AS), and low-skill (LS), were evaluated for comparison purposes.

Considering the large body of extensive literature, there have been studies that investigate the batch scheduling problem for CMS. However, to the authors’ best knowledge this study is the first attempt to examine the batch scheduling problem for MHCMS in a DRC setting addressing SWA. A detailed explanation of the problem is given in Section 2.

One of our main interests in this paper is to compare the performance of some of the most commonly employed metaheuristics in the literature, and to determine the best among them for the considered problem.

The contributions of this study are as follows:
- The batch scheduling problem considering one-piece flow application within cells for the MHCMS is investigated, which has not been studied in the academic literature to date. In addition, SWA with a DRC setting for MHCMS is considered.
- A new optimization model is presented for the problem.
- A lower bound is determined to use for an experimental comparison of the metaheuristics.
- Several structural properties are described, for a better explanation of the problem.
- The metaheuristics are compared regarding the solution quality, and the best among them is determined as the result of detailed analyses.

The remainder of the study is organized as follows. Section 2 introduces the problem definition and optimization model. Section 3 presents structural properties and the lower bound for the problem. Section 4 describes the selected metaheuristics. Section 5 reports the computational comparison of the metaheuristics. Finally, the conclusion is given in Section 6.

2. Problem definition. In this study, the batch scheduling problem for MHCMS in a DRC setting is examined, considering SWA. The decision problems to be met include determining the sequence and starting times of batches, and assigning employees to cells in which operations are performed while considering employee skill levels.

There are $K$ independent labor-intensive hybrid cells in the CMS. Each hybrid cell is composed of $M$ machines designed as multi-product flow shops allocated to process $N$ batches. The structure of the cells and the allocation of the batches to the cells are known in advance. The batches are available to be processed at time zero. Since only one product family is allocated to each cell, the setup times are ignored for the batches. As long as a batch is being processed in a cell it cannot be interrupted and another batch is not allowed to enter the same cell. The parts/jobs in each batch are processed in an overlapping fashion thanks to the unidirectional
and one-piece flow of parts in each hybrid cell. The term of unidirectional flow implies that there is no backtracking for parts of a batch through the machines in a cell. If an operation is a missing operation (MO), then parts are not processed in the corresponding machine and it is not included in the main operation that is comprised of the remaining operations. Each employee has one skill level to perform operations on cells, and the assignment of employees to the operations affects the batch processing times.

There are two main principles that can be implemented to provide the flow of parts through operations in a hybrid manufacturing cell. One of these is called one-piece flow, while the other is called batch-flow. Figure 1 illustrates the one-piece flow and batch-flow principles for a hybrid manufacturing cell, respectively.

The advantage of one-piece flow is that the parts moving through operations in the cell do not have to wait for the rest of the batch to be completed on any machine [48]. The cycle time is used to compute the batch processing time when one-piece flow is applied. On the other hand, the parts that move based on the batch-flow must wait for the other parts of the batch to be completed on each machine. The cycle time cannot be used in the calculation of the batch processing time when batch-flow is applied. The batch processing time is computed by multiplying the sum of operation times (OT1 + OT2 + OT3) by the batch size (BS) in the case that batch-flow is applied. The walking times are assumed to be equal to zero for each of the cells.

Therefore, implementing one-piece flow within a cell may significantly reduce the batch processing time compared with implementing batch-flow within a cell. It is proved in the structural properties that the batch processing time calculated when batch-flow is applied (PT1) and that calculated when one-piece flow is applied (PT2) are not the same (see Lemma 3.4 in Section 3.1).

Once a batch is considered as a part, it is not possible to apply one-piece flow, and batch flow must be applied. For the investigated problem, a batch cannot
be treated as a part, because the one-piece flow principle is implemented in the cells. If the problem is called the part/job scheduling problem, which means each batch is treated as a part/job, then it can lead to a significant increase in the batch processing time. This is one of the reasons for referring to the problem as the batch scheduling problem. The other reason is that the batches are scheduled on the cells, not on the machines. In other words, the sequence of batches is determined for the cells, not for the machines. If the batches are scheduled on the machines, this may reduce the investigated problem to a cell scheduling problem, which is a different type of problem. As explained in Section 1, the main aim of this study is to present a comprehensive problem containing SWA, employee timetabling, and the one-piece flow principle for the batch scheduling.

The cell cycle time is equal to the difference between the completion times of two adjacent parts processed in the same cell, in which one-piece flow is applied. The cell cycle time is computed by comparing the times provided by employees who perform operations on the cell (equation 8). Each employee is responsible for carrying out some or all manual operations in the cell, and the cell cycle time cannot be smaller than the maximum operation time in the cell (equation 19). Furthermore, the cell cycle time cannot be larger than the sum of the manual operation times and the total walking time [28, 7, 40].

Employees have different skill levels. Costa et al. [15] determined three levels of expertise for an employee: senior, normal, and junior. These levels of expertise are also employed in this study. A senior employee has more work experience than a normal employee, and a normal employee similarly has more work experience than a junior employee. The durations of the manual operations are affected by the skill levels of employees, i.e., a manual operation is performed in less time by a senior employee than by a normal or junior employee. Similarly, the cell cycle times are affected by the numbers of employees assigned to the cells for the batches. Therefore, the batch processing times depend on both the number of employees and the skill levels of the employees assigned to the operations of the batches. The coefficients used to determine the manual operation times are given in Table 7 in Section 5.1.

In our problem, the hybrid manufacturing cells have multi-product flow shop layout that applies one-piece flow in each cell. Immediately before a batch enters a cell, the batch is split into parts/jobs, and one-piece flow is carried out. Therefore, each part of the batch is affected by the manual operation times which are directly related to the SWA. Immediately after the parts leave the cell, they are batched again and transferred. The main difficulty of the investigated problem is that the assignment of employees to the operations of batches, the sequence of batches, and the starting times of batches must be concurrently designated which are related to each other.

In general, there are three types of resources: renewable, non-renewable, and doubly constrained [21]. A resource is called renewable if it can be assigned to several non-overlapping operations, and it is called non-renewable if it cannot be assigned to more than one operation throughout the scheduling horizon. On the other hand, if a resource has both renewable and non-renewable characteristics that vary with the time intervals of the scheduling horizon, then it is called doubly constrained. In a significant number of studies, employees are assumed to be either renewable or non-renewable resources, and the problems are classified with respect to the resource characteristics. Regarding the resource type, there are two kinds of
problems: dynamic scheduling and static scheduling problems. The dynamic scheduling problems deal with renewable resource assignment, whereas static scheduling problems consider non-renewable resource assignment. In this study, we investigate renewable resource assignment, i.e., employee timetabling, and formulate the batch scheduling problem in a dynamic environment to further understand the effect of employee flexibility on the average cell response time objective. The employee timetabling adds significant complexity to the investigated problem, because of the additional set of employee availability constraints. The complexity of the problem for both renewable and non-renewable resource allocation is explained in Section 2.2.

Since the employees are considered to be renewable resources for the investigated problem, a serial scheduling scheme (SSS) is constructed to obtain feasible schedules. The SSS is described in Section 4.2.

2.1. **Optimization model.** An integer programming (IP) optimization model for the batch scheduling problem is presented in this section. The basic version of the problem is parallel machines flexible resources scheduling (PMFRS) problem consisting of determining the sequence of jobs and finding the starting time of jobs with an additional amount of resources [18]. The optimization model is reformulated and modified based on the studies of Edis et al. [21] and Yilmaz et al. [62] by considering the SWA.

The indices, parameters, variables, decision variables and optimization model are as follows:

**Indices**
- $i, j$: Index of batches ($i, j = 1, \ldots, N$
- $f, g$: Index of parts/jobs ($f, g = 1, \ldots, F$
- $z$: Index of employees ($z = 1, \ldots, Z$
- $m$: Index of machines ($m = 1, \ldots, M$
- $k, l$: Index of cells ($k, l = 1, \ldots, K$
- $t$: Index of time periods ($t = 1, \ldots, T$
- $S$: Set of batches

**Parameters**
- $N_k$: Total number of batches assigned to cell $k$
- $N$: Total number of batches \( \sum_{k=1}^{K} N_k = N \)
- $F$: Total number of parts/jobs \( \sum_{i=1}^{N} q_i = F \)
- $q_i$: Number of parts in batch $i$
- $e_{zim}$: Manual operation time of parts in batch $i$ on machine $m$ for employee $z$
- $r_{im}$: Automatic operation time of parts in batch $i$ on machine $m$
- $o_k$: Total walking time of employees for batch $i$
- $a_k$: Average walking time of employees for batch $i$
- $u_{im}$: If machine $m$ is used for any operation of batch $i$, 1; if not, 0
- $T$: Number of time periods in the scheduling horizon (Let $T$ be an upper bound on the cell response time)

**Variables**
- $C_k$: Response time of cell $k$/Completion time of last batch on cell $k$
- $c_i$: Completion time of batch $i$
- $c_i$: Processing time of batch $i$
- $c_y_i$: Cell cycle time for batch $i$
- $w_{1zi}$: If worker $z$ is assigned to main operation of batch $i$, 1; if not, 0
- $n_{ij}$: If starting time of batch $i$ is smaller than batch $j$, 1; if not, 0
Decision variables

- \( n_{1ij} \): If completion time of batch \( i \) is larger than batch \( j \), 1; if not, 0
- \( cn_i \): Minimum cell cycle time for batch \( i \)
- \( cx_i \): Maximum cell cycle time for batch \( i \)
- \( s_i \): Starting time of batch \( i \)
- \( v_{zim} \): Operation time of parts in batch \( i \) on machine \( m \) for employee \( z \)
- \( d_i \): Longest operation time of parts in batch \( i \)
- \( FT_i \): Lead time of the first part in batch \( i \)

\( W_i \): Maximum number of employees that can be assigned to the main operation of batch \( i \)

\( W1_i \): Minimum number of employees that can be assigned to the main operation of batch \( i \)

**Decision variables**

- \( x_{it} \): If batch \( i \) completes its processing at time \( t \), 1; if not, 0
- \( b_{ij} \): If batch \( i \) is processed after batch \( j \), 1; if not, 0
- \( w_{zim} \): If worker \( z \) is assigned to machine \( m \) for operation of batch \( i \), 1; if not, 0

The mathematical formulation of the IP optimization model is as follows:

**Objective function**

\[
\min \text{ objective } = \frac{\sum_{k=1}^{K} C_k}{K} \quad (1)
\]

**Constraints**

\[
C_k \geq c_i \quad i \in S_k/k = 1,...,K \quad (2)
\]

\[
c_i = \sum_{t=1}^{T} t \times x_{it} \quad i = 1,...,N \quad (3)
\]

\[
\sum_{t=1}^{T} x_{it} = 1 \quad i = 1,...,N \quad (4)
\]

\[
c_i \geq (c_j + p_j) - T \times (1 - b_{ij}) \quad i \in S_k, j \in S_k/k = 1,...,K \quad (5)
\]

\[
b_{ij} + b_{ji} = 1 \quad i \in S_k, j \in S_k/k = 1,...,K \quad (6)
\]

\[
p_i = cy_i \times (q_i - 1) + \sum_{z=1}^{Z} \sum_{m=1}^{M} w_{zim} \times (e_{zim} + r_{im}) + o_i \quad i = 1,...,N \quad (7)
\]

\[
cy_i \geq w_{zim} \times c_{zim} \times r_{im} \quad i = 1,...,N; z = 1,...,Z; m = 1,...,M \quad (8)
\]

\[
cy_i \geq \sum_{m=1}^{M} w_{zim} \times (a_i + e_{z,i,m}) \quad i = 1,...,N; z = 1,...,Z \quad (9)
\]

\[
\sum_{z=1}^{Z} w_{zim} = u_{im} \quad i = 1,...,N; m = 1,...,M \quad (10)
\]

\[
w_{1zi} \geq w_{zim} \quad i = 1,...,N; z = 1,...,Z; m = 1,...,M \quad (11)
\]

\[
w_{1zi} \geq W_i \quad i = 1,...,N; z = 1,...,Z \quad (12)
\]

\[
3 - (n_{ij} + n_{1ij}) \geq w_{1zi} + w_{1zj} \quad i \in S_k, j \in S_k/k, l = 1,...,K; k \neq l \quad (13)
\]

\[
T \times n_{ij} \geq (c_j - p_j) \quad i \in S_k/k, l = 1,...,K; k \neq l \quad (14)
\]

\[
T \times n_{1ij} \geq c_j - (c_j - p_j) \quad i \in S_k/k, l = 1,...,K; k \neq l \quad (15)
\]

\[
x_{it}, w_{zim}, w_{1zi}, n_{ij}, n_{1ij}, b_{ij}, x_{it} \in 0,1; cy_i, p_i, c_i \geq 0 \quad (16)
\]

The objective function 1 minimizes the average cell response time which is the average completion time of the last batches on the cells [55]. Equation 2 states that the cell response time \( C_k \) is larger than or equal to the completion time of all
batches in this cell. Equations 3 and 4 ensure that each batch must be completed at a unique time. Equation 5 guarantees that at most one batch can be processed in a cell at the same time. Equation 6 ensure that, within each cell \( k \), either batch \( i \in S_k \) precedes batch \( j \in S_k \), or batch \( j \) precedes batch \( i \). Equation 7 is used to compute the batch processing times, which depend on the employees assigned to the operations of batches. This equation is modified and adapted to one-piece flow structure of parts in cells. Equations 8 and 9 are utilized to calculate the cell cycle time for each batch. The study of Miltenburg [40] is used to obtain these equations.

In a hybrid cell, the cycle time for a batch is made up of the manual operation times and the time to move between machines (walking time). The cell cycle time for the batch must be larger than the minimum cycle time and the cycle time computing for each employee that assigned to the operations in the cells. Equation 10 is used to assign employees to the machines to perform operations. Equation 11 shows that an employee assigned to a machine for an operation of a batch is also assigned to the main operation of this batch. While each machine in a cell corresponds to an operation, the cell corresponds to the main operation if the cell is evaluated as a machine. The variables and the equations related to the main operations are designed to prevent employees from being assigned to more than one cell at the same time. Equation 12 states that the total number of employees assigned to the main operation of a batch must be smaller than the maximum number of employees that can be assigned to the main operation of this batch. Employees are considered renewable resources and they can be assigned several operations over the scheduling horizon. However, employees cannot be assigned overlapping operations performed on different cells. Therefore, Equations 13, 14 and 15 are developed to take employee availability constraint into consideration. Equation 13 states that each employee can be assigned at most one main operation at the same time. This constraint avoids the assignment of the same employee to two main operations overlapped in time. Equations 14 and 15 are used to determine the main operations overlapped in time. Equation 16 enforces the binary and sign restrictions on the variables.

The parameters \( cx_i \) and \( cn_i \) are represented in Equations 17 and 18, respectively [62].

\[
\begin{align*}
    cx_i &= \max\left(\sum_{m=1}^{M} \max_{z}(e_{zim} + a_i); d_i \right) \quad i = 1, ..., N \\
    cn_i &= d_i \quad i = 1, ..., N
\end{align*}
\] (17) (18)

The longest operation time and the operation times for parts within a batch on each machine are computed by Equations 19 and 20, respectively.

\[
\begin{align*}
    d_i &= \max_{\forall z}(v_{zim}) \quad i = 1, ..., N \\
    v_{zim} &= e_{zim} + r_{im} \quad i = 1, ..., N; m = 1, ..., M; z = 1, ..., Z
\end{align*}
\] (19) (20)

Equation 21 is used to calculate the maximum number of employees that can be assigned to the main operation of batch \( i \).

\[
W_i = \left[\frac{cx_i}{cn_i}\right] + 1 \quad i = 1, ..., N
\] (21)

The following equation is used to compute the upper bound \( T \).

\[
T = \max_{vk}(\sum_{i=1}^{N} cx_{ik} \times (q_i - 1) \times FT_{ik})
\] (22)
The following equation is used to compute the starting time $s_i$ of batch $i$. In this equation, $p_i$ represents the batch processing time.

$$s_i = c_i - p_i \quad i = 1, ..., N$$  \hspace{1cm} (23)

The second part of equation 7 is used for computing the first lead time $FT_i$ for batch $i$, which is given in the following equation 24.

$$FT_i = \sum_{z=1}^{Z} \sum_{m=1}^{M} w_{zim} \times (e_{zim} + r_{im}) + o_i \quad i = 1, ..., N$$  \hspace{1cm} (24)

2.2. Complexity of the problem. For the investigated batch scheduling problem, if we make the following assumptions then the problem is reduced to the static PMFRS problem [21]:

- The employee resources are non-renewable and identical.
- The cells are identical.
- Each batch is treated as a part.

For the static PMFRS problem, Daniels et al. [18] developed an algorithm that solves the problem in polynomial time.

If we replace the first assumption with the following, without changing the other assumptions, then the problem is reduced to the dynamic PMFRS problem.

- The employee resources are renewable and identical.

The dynamic PMFRS problem has been proved to be NP-hard in the academic literature [21, 18].

In our problem, the objective is to determine the batch scheduling and the employee assignment decisions that minimize the average cell response time. Thus, the problem with non-identical cells and renewable and non-identical resources is also NP-hard.

Owing to the NP-hard nature of the problem, the computational time of the optimization model increases exponentially with the problem size [21]. Therefore, using metaheuristics to solve the problem is plausible.

3. Structural properties and the lower bound for the problem.

3.1. Structural properties.

Lemma 3.1. There exist a schedule $\pi = (b_1, b_2, ..., b_N)$ for the problem, in which the solutions change when: (1) positions (order) of any two batches are processed in the same cell are exchanged, (2) positions of any two parts in different batches processed in the same cell are exchanged. The solutions remain unchanged when: (3) positions of any two parts in the same batch are exchanged.

Proof. (1) Without loss of generality, let us consider batches $b_i$ and $b_j$ are processed on the same cell where $b_i, b_j \in S_k \ (k=1, ..., K; \ K=1)$. When $b_i$ is processed before $b_j$, there are $c^1_i = s^1_i + p^1_i$, $p^1_i = cy_i \times (q_i - 1) + FT_i$, $c^1_j = s^1_j + p^1_j$, and $p^1_j = cy_j \times (q_j - 1) + FT_j$. After the positions of batches $b_i$ and $b_j$ are exchanged, the updated processing times are $p^2_i = cy_i \times (q_i - 1) + FT_i$ and $p^2_j = cy_j \times (q_j - 1) + FT_j$.

After the processing times $p_i$ and $p_j$ of batches are not affected. However, the starting times $s_i$ and $s_j$ and the completion times $c_i$ and $c_j$ of batches are dependent on both the sequence of batches and the availability of employees. The updated completion times are $c^2_i = s^2_i + p^2_i$ and $c^2_j = s^2_j + p^2_j$. It can be concluded that the difference between the cell response time $C^1_k$ and the updated cell response time $C^2_k$ depend on the difference between the completion times and the updated completion.
times of batches \(b_i\) and \(b_j\). Hence, it is seen that \(c^1_i - c^2_i \neq 0\) \(c^1_j - c^2_j \neq 0\). When multiple cells are considered \((k = K; K > 1)\), we see that the starting times \(s_i\) and \(s_j\) and the completion times \(c_i\) and \(c_j\) of batches \(b_i\) and \(b_j\) change and they affect the starting and completion times of other batches. It is seen that the results are the same for both single cell and multiple cells’ cases.

(2) Without loss of generality, let us consider batches \(b_i\) and \(b_j\) are processed on the same cell where \(b_i, b_j \in S_k\) \(J_f \in b_i, J_g \in b_j\) \((f, g = 1, \ldots, F; k = 1, \ldots, K; K = 1)\). When \(b_i\) and \(b_j\) are processed on the same cell, there are \(c^1_i = s^1_i + p^1_i\), \(p^1_i = cy_i + (q_i - 1) + FT_i\), \(c^1_j = s^1_j + p^1_j\), and \(p^1_j = cy_j + (q_j - 1) + FT_j\). After the positions of jobs \(J_f\) and \(J_g\) are exchanged, the updated batch processing time is \(p^2_f = cy_i \times (q_i - 1) + FT_i\) where \(cy_i\) and \(cy_j\) are dependent on parts belonging to different batches (Equations 8 and 9).

When the positions of jobs are exchanged, the processing times are affected since the manual operation times are not the same. It can be concluded that the difference between the cell response time \(C^1_k\) and the updated cell response time \(C^2_k\) depends on the difference between the completion times of the updated completion times of batches \(b_i\) and \(b_j\). Hence, it is seen that \(c^1_i - c^2_i \neq 0\) and \(c^1_j - c^2_j \neq 0\). When multiple cells are considered \((k = K; K > 1)\), we see that the starting times \(s_i\) and \(s_j\) and the completion times \(c_i\) and \(c_j\) of batches \(b_i\) and \(b_j\) change and they affect the starting times and completion times of other batches. It is seen that the results are the same for both single cell and multiple cells’ cases.

(3) Without loss of generality, let us consider batch \(b_i\) is processed on a cell where \(b_i \in S_k, J_f \in b_i, J_g \in b_i\) \((k = 1, \ldots, K; K = 1)\). When \(J_f\) is processed before \(J_g\) in batch \(b_i\), there are \(c^1_i = s^1_i + p^1_i\) and \(p^1_i = cy_i \times (q_i - 1) + FT_i\). After the positions of jobs \(J_f\) and \(J_g\) are exchanged in batch \(b_i\), the updated batch processing time is \(p^2_f = cy_i \times (q_i - 1) + FT_i\) where \(cy_i\) is not dependent on the sequence of jobs in the same batch (Equations 8 and 9). The updated completion time \(c^2_i = s^2_i + p^2_i\) where \(s_i\) is not affected by the sequence of jobs in batches. Since the difference between the cell response time \(C^1_k\) and the updated cell response time \(C^2_k\) depends on the difference between the completion time \(c^1_i\) and the updated completion time \(c^2_i\) of batch \(b_i\), it can be concluded that \(C^1_k - C^2_k = 0\). When multiple cells are considered \((k = K; K > 1)\), we see that the processing time \(p_i\) of batch \(b_i\) does not change and it does not affect the starting and completion times of the other batches. It is seen that the results are the same for both single cell and multiple cells’ cases.

**Lemma 3.2.** The minimum cell response time of cell \(k\) is \(C_k = \sum_{i \in S_k} p_i\).

**Proof.** Without loss of generality, we assume that there exist a schedule \(\pi = (b_1, b_2, \ldots, b_k, \ldots, b_N)\). It is easy to get that \(C_k = \sum_{i \in S_k} p_i = \sum_{i \in S_k} (cy_i \times (q_i - 1) + FT_i)\). The resulting results can be concluded that the minimum response time of cell \(k\) is \(C_k = \sum_{i \in S_k} p_i = \sum_{i \in S_k} (cy_i \times (q_i - 1) + FT_i)\) when the cell cycle time \(cy_i\) is equal to the minimum cell cycle time \(cn_i\) for batch \(b_i\). In addition, the minimum response time of cell \(k\) is \(C_k = \sum_{i \in S_k} p_i = \sum_{i \in S_k} (cy_i \times (q_i - 1) + FT_i)\) when the cell cycle time \(cy_i\) is equal to the maximum cell cycle time \(cx_i\) for batch \(b_i\).

**Lemma 3.3.** The sum of minimum cell response time of cell \(k\) is \(\sum_{k=1}^{K} C_k = \sum_{i=1}^{N} p_i\).

**Proof.** Based on Lemma 3.2, \(\sum_{k=1}^{K} C_k = \sum_{k=1}^{K} \sum_{i \in S_k} p_i = \sum_{k=1}^{K} \sum_{i \in S_k} cy_i \times (q_i - 1) + FT_i\). The resulting results can be concluded that the sum of minimum response time of all cells is \(\sum_{k=1}^{K} C_k = \sum_{k=1}^{K} \sum_{i \in S_k} p_i = \sum_{k=1}^{K} \sum_{i \in S_k} cy_i \times (q_i - 1) + FT_i\) when the cell cycle time \(cy_i\) is equal to minimum cell cycle time \(cn_i\) for batch \(b_i\). The sum of minimum response time of all cells is \(\sum_{k=1}^{K} C_k = \sum_{k=1}^{K} \sum_{i \in S_k} p_i = \sum_{k=1}^{K} \sum_{i \in S_k} cx_i \times \).
(q_i - 1) + FT_i when the cell cycle time c_{y_i} is equal to minimum cell cycle time c_{x_i} for batch b_i.

**Lemma 3.4.** The batch processing time \( p_i \) is changed when batch \( b_i \) is treated as a part/job.

**Proof.** Without loss of generality, let us consider batch \( b_i \) is processed on a cell where \( b_i \in S_k \) (k = 1, ... , K; K = 1). When batch \( b_i \) is treated as a part/job, one-piece flow cannot be applied within the cell and the batch processing time is

\[
p_i^1 = \left( \sum_{z=1}^{Z} \sum_{m=1}^{M} w_{zim} \times (e_{zim} + r_{im}) + o_i \right) \times q_i \text{ where } \sum_{z=1}^{Z} \sum_{m=1}^{M} w_{zim} \times (e_{zim} + r_{im}) + o_i = FT_i.
\]

We also know that when one-piece flow is implemented during the processing of the batch \( b_i \) in the cell, the batch processing time is \( p_i^2 = cy_i \times (q_i - 1) + FT_i \). If the batch size \( q_i \) is equal to 1, \( p_i^1 = \sum_{z=1}^{Z} \sum_{m=1}^{M} w_{zim} \times (e_{zim} + r_{im}) + o_i = FT_i \) and \( p_i^2 = FT_i \); otherwise \( p_i^1 = FT_i \times q_i \) and \( p_i^2 = cy_i \times (q_i - 1) + FT_i \). Thus, we can conclude that \( p_i^1 = p_i^2 \) when \( q_i = 1 \), and \( p_i^1 \neq p_i^2 \) when \( q_i \) is larger than 1.

**3.2. Lower bound.**

**Theorem 3.5.** The lower bound for the problem is \( LB = \frac{\sum_{k=1}^{K} \sum_{i \in S_k} p_i}{K} \) where \( p_i = cn_i \times (q_i - 1) + FT_i \) for batch \( b_i \) (\( b_i \in S_k \); \( k = 1, ..., K \)).

**Proof.** Assume that there is a solution value \( AC \), which is smaller than \( LB \), i.e.,

\[
AC < \frac{\sum_{k=1}^{K} \sum_{i \in S_k} p_i}{K}.
\]

Thus, \( \sum_{k=1}^{K} \frac{C_k}{K} < \frac{\sum_{k=1}^{K} \sum_{i \in S_k} p_i}{K} \) and \( AC \times K = \sum_{k=1}^{K} C_k < \sum_{k=1}^{K} \sum_{i \in S_k} p_i \). From Lemma 3.3, we know that the sum of minimum cell response times is

\[
\sum_{k=1}^{K} \sum_{i \in S_k} p_i = \sum_{k=1}^{K} \sum_{i \in S_k} c_{y_i} \times (q_i - 1) + FT_i
\]

when the cell cycle time \( c_{y_i} \) is equal to the minimum cell cycle time \( c_{x_i} \) for batch \( b_i \). Therefore, \( AC \times K = \sum_{k=1}^{K} C_k < \sum_{k=1}^{K} \sum_{i \in S_k} p_i \) and \( AC < \frac{\sum_{k=1}^{K} \sum_{i \in S_k} p_i}{K} \) obviously contradict with Lemma 3.2 and Lemma 3.3. Thus, we obtain \( LB = \frac{\sum_{k=1}^{K} \sum_{i \in S_k} p_i}{K} \) where \( p_i = cn_i \times (q_i - 1) + FT_i \) for batch \( b_i \).

**4. Metaheuristics.** In this section, we define three metaheuristics, which will later be used to solve the batch scheduling problem addressing SWA.

**4.1. Representation scheme and an illustrative example.** To implement the algorithms for the problem, a representation scheme has to be developed to encode the solutions. The representation schemes may differ according to the problem structure, i.e., these are problem specific [44].

In this paper, the batch list (BL) representation scheme, which is a modified version of the activity list (AL) representation scheme [34], is developed. The representation of an individual/solution shows the priority among the batches to be scheduled. A solution is comprised of four parts, corresponding to the position numbers, the batch list, the assignment of employees to the batches, and the assignment of employees to the machines. The last two parts of the solution are formed depending on the second part.

As an illustration of the problem, an example is provided in Table 1, where a set of four batches are processed on two cells operated by three employees. Batch 1 and batch 2 are processed on cell 1, and batch 3 and batch 4 are processed on cell 2. The maximum and minimum cell cycle times for batch \( i \), denoted by \( c_{x_i} \) and \( cn_i \) respectively, are computed using the equations 17, 18, 19 and 20. The skill levels of all employees are assumed to be normal for this example (see Table 7 in Section 5.1).
Equation 21 is used to calculate the maximum number of employees that can be assigned to batch \( i \). The number of employees is assigned to each batch has to be determined using the values in Table 2.

For example, the number of employees assigned to batch 2 cannot be larger than three or smaller than one. If four employees are assigned to cell 1 for batch 2, then one employee has no effect on the cell cycle time, and this will lead to wasted time for the employee.

Table 2. Maximum and minimum number of employees

Table 3 shows the batch list representation scheme for this problem. While the first row/part is related to the priorities among batches, which are given in the second row, the third and fourth rows are related to the employee availability constraints (equations 13, 14 and 15). While the third row shows the assignment of employees to the cells, the fourth row indicates the assignment of employees to the machines for the batches. The numbers of employees assigned to the cells are determined by considering the values given in Table 2. The last two parts are created to prevent the assignment of employees to overlapping operations. In order to generate a feasible solution from the BL scheme, the serial scheduling scheme (SSS) is applied in a stage-wise fashion. The SSS is introduced in Section 4.2.

Table 3. The batch list representation scheme (Encoding scheme)

To compute the batch processing times, the cell cycle times \( c_{yi} \) and the first lead times \( FT_i \) have to be calculated. While the cell cycle times are computed using the equations 8 and 9, the first lead times are calculated using the equation 24. Table 4 shows the cell cycle times and the first lead times of the batches.

4.2. Decoding procedure and objective function. Every solution (individual) of the population has to be assessed by a fitness value, which is equal to the average cell response time of the schedule. In this study, the serial scheduling scheme (SSS)
is adopted, and applied to each individual to create a resource feasible schedule. SSS consists of \( i = 1, \ldots, N \) phases, in each of which one batch is taken with the position number and scheduled until a feasible schedule is generated. The sequence is designated through the batch list of the chromosome, as given in the first row of Table 3, and the earliest feasible starting time for each batch is determined such that employee availability constraints are satisfied. The fitness value is computed following the SSS implementation. As an illustration, the individual detailed in Table 3 can be represented as shown in Figure 2, and the corresponding average cell response time is equal to 1213 seconds. The batch processing times are calculated considering the equation 7, which requires the cell cycle times \( c_{yi} \) and the first lead times \( FT_i \) for the batches \( (i \in S_k; k = 1, \ldots, K) \). A detailed explanation of SSS can be found in the study of Kolisch [39].

While Figure 2 shows the schedule for the batches, Figure 3 represents the MHCMS for this illustrative example. There are four batches assigned to two cells to be processed. The sequence of batches and the employee assignments are given in Table 3. Owing to the overlapping operations, the employee assignments to the cells and machines directly affect the average flow time objective.

4.3. **Genetic algorithm.** The genetic algorithm (GA), devised by Holland [26], is a population-based evolutionary algorithm. Since being proposed, this algorithm

|   | \( c_{yi} \) | employee1 | employee2 | employee3 | \( c_{yi} \) | \( FT_i \) |
|---|---|---|---|---|---|---|
| 1 (Cell1) | 35 | - | 66 | - | 66 | 86 |
| 2 (Cell1) | 30 | 43 | 43 | - | 43 | 91 |
| 3 (Cell2) | 30 | 50 | - | 25 | 50 | 120 |
| 4 (Cell2) | 20 | 35 | 20 | - | 35 | 85 |

**Table 4.** The cell cycle times and the first lead times

![Figure 2. Average cell response time (Solution decoding)](image-url)
has been applied to solve a wide range of optimization problems, such as computational intelligence, networks, speech recognition, filtering, and production planning and control [57]-[14].

The process is fundamentally based on evolution in nature, and uses selection, crossover, and mutation operators. Because the effectiveness of GA for solving discrete optimization problems has been demonstrated in many studies [2]-[1], it is included in this study for comparison purposes.

The generic pseudo-code of the GA is given as follows:

Initialization
(1) Set the input parameters, including crossover probability \(pc_{\text{crossover}}\), mutation probability \(pm_{\text{mutation}}\), the population size \(PS\) and the termination condition.
(2) Initialize population with randomly generated individuals.

Loop
(3) Evaluate each individual by the fitness function and choose two individuals.
(4) Improve two individuals with the aid of local search.
(5) If the fitness values of the new individuals are better than existing individuals, update the population; otherwise, do not update.
(6) Select individuals to form a new population with using binary tournament selection.
(7) Perform one-point crossover with probability \(pc_{\text{crossover}}\).
(8) Perform mutation with probability \(pm_{\text{mutation}}\).
(9) If the termination condition is reached, calculate fitness functions and go to Step 9; otherwise, go to Step 3.

End Loop
(10) Output the best solution found so far.
The solution representation scheme and fitness function presented in Sections 4.1 and 4.2, respectively, are used for the GA algorithm.

4.3.1. Genetic operators. There are three classes of genetic operators: selection, crossover, and mutation. To create good individuals in the population, the choice of genetic operators is critical [35].

Selection: GA uses a selection scheme to determine the parents which are utilized to generate the offspring from them. Tournament selection is a selection scheme in which \( S \) competitors are evaluated according to their fitness value and determined the best individual to be a parent [39]. Binary tournament selection (\( S=2 \)) approach is utilized to select the parent individuals for crossover.

Crossover: One-point crossover has been used for all parts of the chromosome. In this process, each pair of individuals is chosen as the parents, a father, and a mother, with a probability \( p_{cross} \) and two new individuals, a daughter and a son, are produced as offspring. It first determines a single crossover point \( t \) randomly and then creates an offspring as follows. The first \( t \) positions in all parts of the chromosome are taken from the father and the remaining positions in all parts of the chromosome are derived from the mother. Note that the employee assignment and the machine assignment parts in both the daughter and the son are obtained...
with the employee and machine assignments of the corresponding batches derived from the father and the mother. The crossover operator is illustrated in Figure 4.

Mutation: Swapping mutation operator is utilized for the diversification purpose for each newly obtained individual. It first selects two positions with a probability of $p_{\text{mutation}}$, and swaps the batches in the selected positions. Note that this affects the employee and the machine assignments: that is, swapping is applied to both the batch-employee assignment and the employee-machine assignment parts. Next, the mutation operator modifies the batch-employee and the employee-machine assignment parts with a probability of $p_{\text{mutation}}$.

4.4. Artificial bee colony. The artificial bee colony (ABC) algorithm, developed by Karaboga [30], is a swarm intelligence-based algorithm. There are three groups of operators used in iterative processes in the algorithm: employed bee, onlooker bee, and scout bee. A bee in charge of exploiting the food sources is named an employed bee. A bee waiting in the hive to choose a food source found by employed bees is called an onlooker bee. A bee performing a random search to find a new food source is named a scout bee. In this study, it is assumed that the number of employed bees is equal to the number of onlooker bees. The amount of nectar for each food source corresponds to a solution to the addressed problem.

The generic pseudo-code of the ABC algorithm is given as follows:

```
Initialization
(1) Set the input parameters, involving the number of food sources ($NFS$), scout iteration ($limit$) and the termination condition.
(2) Initialize population with randomly generated individuals.
Loop
(3) Employed bee stage. The sub-stages applied for population are as follows:
(3.1) For each pair of food sources, create two new food sources with utilizing one-point crossover and mutation operators shown in Section 4.3.1.
(3.2) Evaluate each individual by the fitness function and choose two food sources.
(3.3) Improve two new food sources with the aid of the local search introduced in Section 4.6.
(3.4) If the fitness values of the new food sources are better than existing food sources, update the population; otherwise, set $trialscout = trialscout + 1$, $trialscout$ is the number of abandoning the non-improvement.
(4) Onlooker bee stage. The sub-stages applied for population are as follows:
(4.1) Choose two food sources by utilizing the binary tournament selection.
(4.2) Improve two new food sources based on local search presented in Section 4.6.
(4.3) If the fitness values of the new food sources are better than existing food sources, update the population; otherwise, set $trialscout = trialscout + 1$.
(5) Scout bee stage. If a food source has not been improved ($trialscout > limit$), abandon it, create a new food source randomly.
(6) Record the best solution
(7) If the termination condition is reached, go to Step 8; otherwise, go to Step 3.
End Loop
(8) Output the best solution found so far.
```

Because the ABC algorithm has proven effectiveness and outperforms most of the
commonly employed algorithms regarding distinguished solution performance [31]-[10], such as GA and particle swarm optimization (PSO), ABC is included in this study for comparison purposes.

The solution representation scheme and the fitness function, presented in Sections 4.1 and 4.2 respectively, are also used for the ABC algorithm.

4.5. Simulated annealing algorithm. The simulated annealing (SA) algorithm was introduced by Metropolis [11], and applied to combinatorial optimization problems by Kirkpatrick [33]. The SA is a single-solution based metaheuristic, and it simulates a real-life annealing process. This annealing process entails heating and gradual cooling of a metal to obtain a strong crystalline structure [56].

The SA algorithm has been implemented for optimization problems over the past few decades, and has demonstrated efficiency for discrete optimization problems [[1], [51], and [12]]. Therefore, the SA is included in this study for comparison purposes. The solution representation scheme and the fitness function, presented in Sections 4.1 and 4.2 respectively, are also used for the SA algorithm.

The generic pseudo-code of the SA algorithm is given as follows:

1. Set the input parameters, including initial temperature \( \text{initialtemp} \), cooling rate \( \text{coolingrate} \) and epoch length \( \text{epoch} \) and the termination condition.
2. Initialize a randomly generated solution \( s \) and set \( T = \text{initialtemp}, T \) is the current temperature.
3. Loop
4. Evaluate individual by the fitness function \( f(s) \).
5. Generate a random neighborhood solution \( s' \) with using swap operator and calculate the fitness value \( f(s') \) and set \( \text{Iteequ} = 1, \text{Iteequ} \) is the current iteration for the epoch condition.
6. Set \( \Delta = f(s') - f(s), \Delta \) is the difference between fitness (objective) function values.
7. If \( \Delta \leq 0 \) or \( e^{-\frac{\Delta}{T}} > \text{rand}(0, ..., 1) \) update the solution; otherwise, continue with the existing solution \( s \), set \( \text{Iteequ} = \text{Iteequ} + 1 \). (\( T \) is the current temperature)
8. If \( \text{Iteequ} \geq \text{epoch} \) (\( \text{epoch} \) is the maximum number of iteration for epoch condition), go to Step 8; otherwise, go to Step 4.
9. Set \( T = T \times \text{coolingrate} \) and go to Step 3.
10. If the termination condition is satisfied, go to Step 10; otherwise go to step 4. End Loop
11. Output the best solution.

4.6. Local search. In this study, a local search procedure is adapted to improve the performance of the ABC algorithm. With the aid of the local search, a better solution can be explored from the neighborhoods of an existing solution. The swap and the insertion operators are applied sequentially once at each iteration of local search to generate neighborhood solutions. The stages of the local search are performed for all solutions until the maximum number of iterations \( \lambda \) is reached.

The generic pseudo-code of local search is represented as follows:
Initialization
(1) Set $i=1$. 
Loop
(2) Generate a neighborhood solution by applying both swap and insertion operators sequentially once. 
(3) Evaluate neighborhood solution by the fitness function (objective). 
(4) If the neighborhood solution is better than the existing solution with respect to fitness function, update the solution; otherwise, continue with the existing solution. 
(5) Set $i = i + 1$, if $i$? (is the maximum iteration for the local search), stop; otherwise, go to Step 2. 
End Loop
(5) Output the solution.

5. Computational experiments. In order to evaluate the performance of the selected metaheuristics, it is necessary to compare them with each other for the batch scheduling problem in MHCMS.

To compare the performances of the proposed algorithms, experiments are conducted on real production data provided by a manufacturing company in the pipeline industry. The minimum and maximum cell cycle times, operation times, and total walking times are taken from the study of Yilmaz et al. [62], and are listed in Table 5, where the missing operations are denoted by MO.

The parts families, which are classified as DN 25-32-40-50, DN 65-80, DN 100-125-150 and DN 200-250-300 according to their nominal diameters, are produced in the manufacturing system.

There are seven different types of machines in the cells: roller, TIG welding, horizontal forming, vertical forming, reel, cutting, and closing machines.

The algorithms were coded in MATLAB software and tested on the same computer, a 2.4 GHz Intel(R) Core i7-3630QM CPU with 16 GB of RAM.

5.1. Experimental data. The experiments take three factors into consideration: the number of batches on each cell (NBEC), the number of employees for each skill level (NESL), and the algorithms. The experimental factors, along with their levels, are listed in Table 6. The number of batches on each cell is denoted by NB, and listed in Table 5. For the comparison analysis, the batch sizes were randomly generated from a discrete uniform distribution in the interval $U[1, 20]$. The total number of employees in the system was equal to nine. A benchmark of $3 \times 10=30$ test problems was generated and solved by each of the algorithms. Each experiment was repeated ten times with different random seeds.

The level of the NESL factor affects the distribution of the number of employees in the manufacturing system according to skill levels. For instance, when the NESL level is equal to 1, the employees are equally distributed across the skill levels. Similarly, when the NESL level is equal to 10, each employee has a senior skill level in the system.

The skill levels (namely junior, normal, and senior) of employees affect the manual operation times on the cells. For example, when the skill level of an employee is assumed to be junior, the manual operation times given in Table 5 are divided by 0.63 (skill level coefficient). The determined skill level coefficients are similar to the values in the study of Costa et al. [15], and are listed in Table 7.
Table 5. Cell cycle times, batch sizes, operation times and total walking times

| i | $c_{x_1}$ | $c_{p_1}$ | $q_i$ | $c_{x_{2im}}$ | $c_{x_{im}}$ | $c_{x_{2im}}$ | $c_{x_{im}}$ | $c_{x_{2im}}$ | $c_{x_{im}}$ | $a_i$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 957 | 450 | 5 | 16 | 150 | MO | MO | 450 | 155 | 108 |
| 2 | 622 | 468 | 4 | 16 | 150 | 110 | 358 | MO | 160 | 108 |
| 3 | 915 | 432 | 10 | 15 | 144 | MO | MO | 432 | 148 | 105 |
| 4 | 590 | 455 | 5 | 15 | 144 | 100 | 355 | MO | 152 | 105 |
| 5 | 887 | 420 | 15 | 15 | 140 | MO | MO | 420 | 145 | 96 |
| 6 | 555 | 438 | 8 | 15 | 140 | 88 | 350 | MO | 145 | 96 |
| 7 | 741 | 319 | 15 | 12 | 135 | MO | MO | 319 | 130 | 84 |
| 8 | 508 | 355 | 5 | 12 | 135 | 75 | 280 | MO | 138 | 84 |
| 9 | 626 | 250 | 10 | 9 | 120 | MO | MO | 250 | 117 | 71 |
| 10 | 436 | 272 | 10 | 9 | 120 | 60 | 212 | MO | 117 | 71 |
| 11 | 472 | 155 | 15 | 8 | 108 | MO | MO | 155 | 88 | 66 |
| 12 | 363 | 186 | 9 | 8 | 108 | 26 | 160 | MO | 102 | 66 |
| 13 | 525 | 182 | 12 | 8 | 115 | MO | MO | 182 | 95 | 71 |
| 14 | 414 | 210 | 8 | 8 | 115 | 50 | 160 | MO | 112 | 71 |
| 15 | 617 | 226 | 14 | 11 | 131 | MO | MO | 226 | 102 | 85 |
| 16 | 469 | 258 | 8 | 11 | 131 | 58 | 200 | MO | 120 | 85 |
| 17 | 755 | 318 | 10 | 14 | 145 | MO | MO | 318 | 115 | 94 |
| 18 | 541 | 360 | 5 | 14 | 145 | 80 | 280 | MO | 134 | 94 |
| 19 | 940 | 431 | 5 | 18 | 153 | MO | MO | 431 | 139 | 118 |
| 20 | 653 | 490 | 15 | 18 | 153 | 120 | 370 | MO | 155 | 118 |
| 21 | 1070 | 485 | 10 | 25 | 168 | MO | MO | 485 | 163 | 135 |
| 22 | 739 | 545 | 8 | 25 | 168 | 135 | 410 | MO | 175 | 135 |
| 23 | 1202 | 528 | 10 | 34 | 185 | MO | MO | 528 | 188 | 161 |
| 24 | 857 | 607 | 5 | 34 | 185 | 147 | 460 | MO | 211 | 161 |

Table 6. Experimental factors and their levels

| Factor | Level |
|---|---|
| Number of Batches on Each Cell (NBEC) *(Problem size factor)* | 1 NB \ 2 \[5 \times \text{NB}\] \ 3 \[10 \times \text{NB}\] |
| Number of Employees for each Skill Level (NESL) | Junior Normal Senior |
| 1 | 33% 33% 33% |
| 2 | 66% 33% 00% |
| 3 | 66% 00% 33% |
| 4 | 100% 00% 00% |
| 5 | 00% 66% 33% |
| 6 | 33% 66% 00% |
| 7 | 00% 100% 00% |
| 8 | 33% 00% 66% |
| 9 | 00% 33% 66% |
| 10 | 00% 00% 100% |
5.2. **Evaluation metric.** In this study, the response variable used for the comparison analysis is the \( RPD \), calculated according to equation 25. The lower bound (\( LB \)), which is computed using Theorem 3.5, is used as the best solution for a given class of problem according to the following formula:

\[
RPD\% = \frac{\text{solution-LB}}{\text{LB}}
\]  

(25)

The \( RPD \) value, which is a percentage deviation of solutions versus \( LB \), is computed for each instance. Obviously, the smaller values of the \( RPD \) are more desirable than the larger values.

5.3. **Parameters tuning.** The results for all metaheuristics are susceptible to the parameters used. Hence, the best combination of parameter values should be determined through repeated simulations [46]. In this study, the primary estimation values of the parameters are obtained through repetitive experiments. One parameter is varied each time, and the others are kept fixed. The best-known combination of parameter values is experimentally achieved as a result of a series of experiments performed with different parameter values. The same benchmark set used for the comparison analysis is employed for the parameter tuning analysis. In particular, the skill level of each employee is assumed to be normal. Each experiment is repeated 10 times with different random seeds. In order to achieve the convergence of each algorithm, the algorithms were terminated after 50 iterations without improvement to the best solutions. The maximum number of iterations without improvements is determined as the termination condition for the parameter tuning analysis, and named \( \text{termination}_{-1} \).

A two-stage adjustment method has been conducted to determine the parameters for ABC, GA, and SA, respectively. The method consists of two stages: an initial stage and improvement stage. In the initial stage, the algorithms are run to determine promising parameter values, as marked in bold in Table 8, for each NBEC level. The method continues to the improvement stage after obtaining the first parameter values from the initial phase. The improvement stage attempts to achieve the best combination of parameter values by running the algorithms in the neighborhoods of the promising values obtained in the initial stage, for each NBEC level. The best combination of parameter values achieved at the end of the improvement stage is denoted in bold in Table 9.

Table 10 shows the maximum computational times, in seconds, of the metaheuristic for each NBEC level.

To make an unbiased comparison, a limited computational time is used as a termination condition for comparison analysis [47, 23] and named as \( \text{termination}_{-2} \), which is fixed to “15 × Level of problemsize factor (NBEC) × (coefficient of NBEC)\(^{\frac{1}{2}}\)” seconds. The formula for the limited CPU time is determined using the values in Table 10.

5.4. **Computational results.** In this section, we present the results of experiments performed with the metaheuristics for the addressed problem. First, in order
Table 8. The promising values of the parameters for the metaheuristics

| Algorithm | Notation | Values                  | Combination |
|-----------|----------|-------------------------|-------------|
| GA        | PS       | `0.2,0.4,0.6,0.8`       | NBEC=1 60 80 |
|           | pcross   | `0.1,0.2,0.3,0.4`       | NBEC=2 0.2 0.2 |
|           | pmutation| `2,4,6,8,10`            | NBEC=3 0.8 0.6 |
| ABC       | NFS      | `20,40,60,80,100`       | NBEC=1 40 60 80 |
|           | λ        | `2,4,6,8,10`            | NBEC=2 2 4 4 |
|           | limit    | `2,4,6,8,10`            | NBEC=3 6 6 4 |
| SA        | initialtemp | `10^3 \times (1,3,5,7)` | NBEC=1 1 3 5 |
|           | coolingrate | `0.9,0.95,0.99`       | NBEC=2 0.99 0.99 0.99 |
|           | epoch    | `5,10,15,20`            | NBEC=3 10 15 15 |

Table 9. The tuned values of the parameters for the metaheuristics

| Algorithm | Notation | Combination |
|-----------|----------|-------------|
| GA        | PS       | `30,40,50` | NBEC=1 40 60 80 |
|           | pcross   | `0.7,0.8,0.9`       | NBEC=2 0.2 0.2 |
|           | pmutation| `0.25,0.3,0.35`       | NBEC=3 0.15 0.15 |
| ABC       | NFS      | `30,40,50` | NBEC=1 40 60 80 |
|           | λ        | `1,2,3`      | NBEC=2 3.45 3.45 |
|           | limit    | `5,6,7`      | NBEC=3 5.67 5.67 |
| SA        | initialtemp | `750,1000,1250` | NBEC=1 4000 5000 6000 |
|           | coolingrate | `0.98,0.99`     | NBEC=2 0.98 0.98 |
|           | epoch    | `8,10,12`     | NBEC=3 13 13.15 |

Table 10. The maximum computational times

|               | ABC | GA | SA |
|---------------|-----|----|----|
| NBEC=1        | 14.87 | 12.68 | 9.16 |
| NBEC=2        | 53.40 | 46.07 | 38.73 |
| NBEC=3        | 98.25 | 95.53 | 83.68 |

To evaluate the effectiveness of the local search procedure, it is excluded from the GA and ABC algorithms, and these algorithms without local search are denoted by GAWLS and ABCWLS, respectively. Then, these algorithms are also implemented for the problem using the same parameter values obtained for the GA and ABC algorithms, in order to make the comparison fairer. Therefore, the number of runs in the experimentation is equal to 1500 (30 × 5 × 10). Table 11 reports the medians of the relative percentage deviation (RPD) obtained by the algorithms, including GA and ABCWLS, for each class of problem, together with the computational time expressed in seconds. In addition, the medians of the RPD values for each NBEC level are presented in the last rows of Table 11, in order to emphasize the differences among them. The RPD values reported in Table 11 are obtained by considering that the total number of employees in the system is equal to nine.
As seen in Table 11, for the majority of the problems the RPD values for each algorithm range from 20% to 40%, depending on the NBEC level. As the NBEC level increases, considerable increases are observed in the RPD values of the algorithms. This is not caused by the poor performance of the algorithms, but rather by the lower bound and the limited number of employees. The developed lower bound equation (Section 3.2) uses the minimum cycle time for each cell, which is possible when the maximum number of employee assignments is conducted in each cell. However, if there is smaller number of employees in the system than the sum of the maximum number of employees that can be assigned to each cell, as computed

| NESL×NBEC | LB | SA | GA | ABC | ABCWLS | GAWLS | CPU |
|-----------|----|----|----|-----|--------|-------|-----|
| 1×1       | 23580 | 17.66 | 16.96 | 18.82 | 18 | 17.14 | 15 |
| 2×1       | 18989 | 19.34 | 20.01 | 16.08 | 20.04 | 21.39 | 15 |
| 3×1       | 22263 | 17.99 | 18.97 | 16.62 | 18.83 | 19.05 | 15 |
| 4×1       | 23196 | 20.22 | 19.23 | 17.4 | 18.55 | 20.13 | 15 |
| 5×1       | 22223 | 18.05 | 19.18 | 16.02 | 18.99 | 18.25 | 15 |
| 6×1       | 22097 | 18.05 | 19.18 | 16.02 | 18.99 | 18.25 | 15 |
| 7×1       | 20685 | 20.57 | 18.87 | 18.53 | 18.32 | 19.08 | 15 |
| 8×1       | 24351 | 19.93 | 21.03 | 16.56 | 17.36 | 21.58 | 15 |
| 9×1       | 19621 | 17.45 | 18.6 | 18.57 | 18.92 | 20.4 | 15 |
| 10×1      | 23961 | 20.42 | 20.57 | 15.97 | 20.02 | 21.16 | 15 |

**Medians**

| 1×2 | 123093 | 34.99 | 30.13 | 26.88 | 27.85 | 32.15 | 51.2 |
| 2×2 | 116383 | 33.24 | 31.81 | 29.88 | 28.44 | 32.73 | 51.2 |
| 3×2 | 125259 | 28.87 | 27.55 | 29.16 | 29.28 | 29.71 | 51.2 |
| 4×2 | 114564 | 30.38 | 28.49 | 28.59 | 29.36 | 29.73 | 51.2 |
| 5×2 | 108065 | 28.37 | 30.16 | 29.74 | 29.65 | 32.76 | 51.2 |
| 6×2 | 121750 | 30.88 | 30.67 | 29.75 | 30.49 | 32.6 | 51.2 |
| 7×2 | 119274 | 31.85 | 31.92 | 27.21 | 31.93 | 33.58 | 51.2 |
| 8×2 | 118577 | 30.56 | 29.65 | 28.24 | 32.1 | 29.16 | 51.2 |
| 9×2 | 104521 | 29.57 | 30.03 | 25.62 | 32.12 | 30.22 | 51.2 |
| 10×2 | 111100 | 29.45 | 29.11 | 29.65 | 31.03 | 35.54 | 51.2 |

**Medians**

| 1×3 | 242399 | 36.45 | 35.13 | 34.05 | 36.48 | 39.72 | 96.9 |
| 2×3 | 249240 | 40.28 | 39.55 | 36.66 | 36 | 41.11 | 96.9 |
| 3×3 | 238045 | 39.79 | 36.61 | 34.87 | 36.76 | 39.92 | 96.9 |
| 4×3 | 246187 | 37.21 | 38.62 | 35.04 | 38.24 | 39.51 | 96.9 |
| 5×3 | 229842 | 36.44 | 35.87 | 36.48 | 39.46 | 41.09 | 96.9 |
| 6×3 | 228375 | 36.34 | 36.01 | 36.14 | 38.63 | 39.45 | 96.9 |
| 7×3 | 211540 | 37.34 | 36.91 | 35.92 | 39.04 | 38.87 | 96.9 |
| 8×3 | 225562 | 40.26 | 40.27 | 36.21 | 37.61 | 41.78 | 96.9 |
| 9×3 | 232299 | 39.72 | 38.53 | 33.32 | 36.2 | 42.21 | 96.9 |
| 10×3 | 237518 | 39.76 | 37.6 | 34.73 | 36.86 | 41.06 | 96.9 |

**Medians**

| 1×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 2×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 3×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 4×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 5×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 6×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 7×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 8×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 9×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |
| 10×2 | 30.816 | 29.952 | 28.472 | 30.225 | 31.818 | 51.2 |

**Medians**

| 1×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 2×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 3×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 4×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 5×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 6×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 7×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 8×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 9×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |
| 10×3 | 38.359 | 37.51 | 35.342 | 37.528 | 40.472 | 96.9 |

**Table 11.** Median of RPD values and computational time of algorithms for 9 employees
by the equation 21, this means that there is a limited number of employees in the system. Thus, it can be concluded that the MHCMS system we consider has a limited number of employees. Because the lower bound equation does not take into account the number of employees in the MHCMS, the $R_P D$ values are large, and these values become larger with the increase in the number of batches assigned to the cells.

In order to evaluate the quality of the lower bound with respect to the total number of employees in the system, it is assumed that the MHCMS has 15 employees, which implies that there are no employee restrictions in the system. Three different problems are solved under the assumption that the total number of employees in the system is 15, and the computational results are presented in Table 12.

| NESL × NBEC | LB     | SA    | GA    | ABC   | ABCWLS | GAWLS | CPU |
|-------------|--------|-------|-------|-------|--------|-------|-----|
| 7 × 1       | 20685  | 3.68  | 3.45  | 3.11  | 3.53   | 3.65  | 15  |
| 7 × 2       | 119274 | 5.11  | 4.9   | 4.65  | 5.07   | 5.15  | 51.2|
| 7 × 3       | 228375 | 6.32  | 5.98  | 5.6   | 5.92   | 6.44  | 96.9|

Table 12. Median of $R_P D$ values and computational time of algorithms for 15 employees

To obtain better $R_P D$ values, the results given in Table 12 show that the total number of employees should be taken into consideration in the lower bound equation. However, the improvement in the lower bound equation is not included in this study. The equation can be improved by considering this fact in future studies.

Explanations and graphs are presented below for the case in which the total number of employees in the system is nine. Figure 5 shows the box plot of the $R_P D$ for each algorithm, from which it is not clear which algorithm outperforms the others. The reason for this is that the box-plots are drawn by taking all of the NBEC and NESL levels into account together. Thus, further analyses are necessary to reach a conclusion regarding the performances of algorithms according to the factor levels.

The interactions of the independent factors of NBEC and NESL for the algorithms are examined using two separate graphs, which summarize the results. The NBEC factor has been taken into consideration to examine the variations in the performances of the algorithms according to the number of batches processed in the cells. Figure 6 presents the $R_P D$ results obtained by the algorithms for each NBEC level. It can be seen from Figure 6 that the ABC algorithm exhibits the same trends for different NBEC levels, and the performance of the ABC algorithm is noticeably better when the level of NBEC is greater than one. From these results, it can be deduced that the ABC algorithm performs better when the number of batches processed in the cells increases, which makes the employee timetabling problem more difficult to solve for systems with a limited number of employees.

It can also be seen that the SA, GA, and ABCWLS algorithms exhibit similar tendencies, and they produce a slightly better performance than the GAWLS algorithm for various levels of NBEC. In addition, the ABCWLS and GAWLS algorithms are slightly better than the SA and GA algorithms with respect to consistency.

Figure 7 is also included to this section, to examine the effects of NESL levels on the performances of the algorithms.
Figure 7 shows the interaction between the algorithms and the NESL levels increasing from 1 to 10. According to this figure, all metaheuristics exhibit the same trends for the different levels of NESL. In addition, the ABC algorithm shows a better performance than the other algorithms regarding consistency along the levels of NESL.

Figures 5 and 6 show that the ABC algorithm implemented in this study clearly exhibits a superior performance than the other algorithms for each level of the NBEC and NESL factors, because the local search procedure significantly boosts ABCs performance. Although the local search also enhances GAs performance, ABC provides better outcomes. Thus, it can be concluded that the local search procedure is more effective for ABCs performance than for GA. The conclusions reached from 5 and 6 are also supported by the ANOVA analysis results presented in Table 13. Once ABC has been identified as the best algorithm for coping with the batch scheduling problem in MHCMS, the effect of the NESL levels on ABCs performance should also be examined. Therefore, we decided to answer the following question by interpreting Figure 8, which represents ABCs performance for each NESL level.

- Which NESL levels have a positive effect on ABCs performance with regard to the criteria of the consistency, median, and upper whisker?
When the NESL level is equal to 1, the $RPD$ results are more consistent compared with the other levels. In other words, if there is an equal number of employees at each skill level, then the most consistent results are achieved.

If the NESL levels are compared with each other with respect to the median, the best $RPD$ results occur in the case that the NESL level is equal to 9. Once 66% of the employees have a senior skill level and 33% of them have a normal skill level in the system, 50% of the $RPD$ values are lower than the midpoint.

When the upper whiskers are considered, the smallest of the maximum $RPD$ results is obtained in the case that the NESL level is equal to 10, which means that each employee in the system has the senior skill level.

It can be inferred from the above information that the employee skill levels affect ABCs performance, and there are trade-offs among NESL levels according to different performance criteria. In addition, the box plot for the NESL factor shows the distributional characteristics of all of the levels. Thus, it is likely that different interpretations can be produced in terms of different criteria. The ANOVA analysis is performed to obtain statistical evidence for the conclusions reached from Figures 5 and 6. The results are analyzed by three-way completely randomized ANOVA, with interactions where the factors are NBEC, NESL, and the algorithms. The response variable used in the ANOVA method is the $RPD$. The interactions provide us with an opportunity to examine the manner in which the algorithms depend on the
The normality assumption is tested by a residual analysis, which indicates departures from normality. However, the ANOVA method is robust to violations of this assumption. In addition, when the group sizes are equal to each other, ANOVA is also robust to violations of the homoscedasticity assumption [47]. Therefore, we decided to use the ANOVA method to evaluate the different factor effects on the RPD, even though the assumptions are violated.

As previously mentioned, the results in Table 13 confirm that the effects of NBEC and the algorithms are separately statistically significant. Furthermore, no interaction effects between any factors are observed to be statistically significant.
6. **Conclusions.** In this paper, the batch scheduling problem has been examined for MHCMS in a DRC setting, considering SWA, with the objective of the average cell response time. The development of an integer linear optimization model is described for the problem. Because the considered problem is NP-hard in the strong sense, three metaheuristics have been proposed, and compared with each other. For comparison purposes, computational experiments were conducted using a set of test instances that were generated based on real production data. Finally, numerical results demonstrated the effectiveness and efficiency of the ABC algorithm compared with the other algorithms. The ABC algorithm outperformed the GA and SA algorithms regarding the solution quality, especially for large-sized problem instances.

Because this paper deals with a batch scheduling problem in a real CMS, the findings of the study could be used by engineering managers for effective scheduling applications in production.

Regarding future research directions, employee skills can be evaluated as zero and one instead of at multiple levels. Furthermore, the operation times can be considered as stochastic or fuzzy in place of deterministic. Finally, this study could be extended to other manufacturing systems, such as a hybrid manufacturing system (HMS).
REFERENCES

[1] M. O. Adamu and A. Adewumi, Comparing the performance of different meta-heuristics for unweighted parallel machine scheduling, *South African Journal of Industrial Engineering*, **26** (2015), 143–157.

[2] A. O. Adewumi and M. M. Ali, A multi-level genetic algorithm for a multi-stage space allocation problem, *Mathematical and Computer Modeling*, **51** (2010), 109–126.

[3] A. O. Adewumi B. A. Sawyerr and M. M. Ali, A heuristic solution to the university timetabling problem, *Engineering Computations*, **26** (1984), 972–984.

[4] R. G. Askin and Y. Huang, Forming effective worker teams for cellular manufacturing, *International Journal of Production Research*, **39** (2001), 2431–2451.

[5] A. Azadeh, M. Sheikhalishahi and M. Koushan, An integrated fuzzy DEA-Fuzzy simulation approach for optimization of operator allocation with learning effects in multi products CMS, *Applied Mathematical Modelling*, **37** (2013), 9922–9933.

[6] A. N. Balaji and S. Porselvi, Artificial immune system algorithm and simulated annealing algorithm for scheduling batches of parts based on job availability model in a multi-cell flexible manufacturing system, *Procedia Engineering*, **97** (2014), 1524–1533.

[7] J. T. Black and S. L. Hunter, *Lean Manufacturing Systems and Cell Design*, Society of Manufacturing Engineers, 2003.

[8] G. Celano, A. Costa and S. Fichera, Scheduling of unrelated parallel manufacturing cells with limited human resources, *International Journal of Production Research*, **46** (2008), 405–427.

[9] V. I. Cesani and H. J. Steudel, A study of labor assignment flexibility in cellular manufacturing systems, *Computers & Industrial Engineering*, **48** (2005), 571–591.

[10] W. D. Chang, Nonlinear CSTR control system design using an artificial bee colony algorithm, *Simulation Modelling Practice and Theory*, **31** (2013), 1–9.

[11] W. D. Chang, Equation of state calculations by fast computing machines, *The Journal of Chemical Physics*, **21** (1953), 1087–1092.

[12] S. Chetty and A. O. Adewumi, Three new stochastic local search metaheuristics for the annual crop planning problem based on a new irrigation scheme, *Journal of Applied Mathematics*, **2013** (2013), Article ID 158538, 14 pages.

[13] D. A. Coley, *An Introduction to Genetic Algorithms for Scientists and Engineers*, World Scientific Pub. Co. Inc., 1999.

[14] P. Cortes, J. Larraneta, L. Onieva, J. M. García and M. S. Caraballo, Genetic algorithm for planning cable telecommunication networks, *Applied Soft Computing*, **1** (2001), 21–33.

[15] A. Costa, F. A. Cappadonna and S. Fichera, Joint optimization of a flow-shop group scheduling with sequence dependent set-up times and skilled workforce assignment, *International Journal of Production Research*, **52** (2014), 2696–2728.

[16] A. Costa, F. A. Cappadonna and S. Fichera, A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times, *The International Journal of Advanced Manufacturing Technology*, **69** (2013), 2799–2817.

[17] A. Costa, S. Fichera and F. A. Cappadonna, A genetic algorithm for scheduling both jobs families and skilled workforce, *International Journal of Operations and Quantitative Management*, **19** (2013), 221–247.

[18] R. L. Daniels, B. J. Hoopes and J. B. Mazzola, Scheduling parallel manufacturing cells with resource flexibility, *Management Science*, **42** (1996), 1260–1276.

[19] S. R. Das and C. Canel, An algorithm for scheduling batches of parts in a multi-cell flexible manufacturing system, *International Journal of Production Economics*, **97** (2005), 247–262.

[20] D. J. Davis, H. V. Kher and B. J. Wagner, Influence of workload imbalances on the need for worker flexibility, *Computers & Industrial Engineering*, **57** (2009), 319–329.

[21] E. B. Edis, C. Oguz and I. Ozkarahran, Parallel machine scheduling with additional resources: Notation, classification, models and solution methods, *European Journal of Operational Research*, **230** (2013), 449–463.

[22] G. Egilmez, B. Erenay and G. A. Suer, Stochastic skill-based manpower allocation in a cellular manufacturing system, *Journal of Manufacturing Systems*, **33** (2014), 578–588.

[23] B. Fahimnia, H. Davarzani and A. Eshragh, Planning of complex supply chains: A performance comparison of three meta-heuristic algorithms, *Computers & Operations Research*, **89** (2018), 241–252.
[24] J. Fan, M. Cao and D. Feng, Multi-objective dual resource-constrained model for cell formation problem, In Management of Innovation and Technology IEEE International Conference, (2010), 1031–1036.
[25] J. W. Fowler, P. Wirojanagud and E. S. Gel, Heuristics for workforce planning with worker differences, European Journal of Operational Research, 190 (2008), 724–740.
[26] J. H. Holland, Genetic algorithms, Scientific American, 267 (1992), 66–72.
[27] M. P. Hottenstein and S. A. Bowman, Cross-training and worker flexibility: A review of DRC system research, The Journal of High Technology Management Research, 9 (1998), 157–174.
[28] H. Hyer and U. Wemmerlov, Reorganizing the Factory Competing Through Cellular Manufacturing, Productivity Press, 2002.
[29] J. B. Jensen, The impact of resource flexibility and staffing decisions on cellular and departmental shop performance, European Journal of Operational Research, 127 (2000), 279–296.
[30] N. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters, Journal of the Franklin Institute, 346 (2009), 328–348.
[31] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing, 8 (2008), 687–697.
[32] D. Karaboga and E. Kaya, An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training, Applied Soft Computing, 49 (2016), 423–436.
[33] S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, Journal of Statistical Physics, 34 (1984), 975–986.
[34] R. Kolisch, Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, European Journal of Operational Research, 90 (1996), 320–333.
[35] X. Li and L. Gao, An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem, International Journal of Production Economics, 174 (2016), 93–110.
[36] Y. Li, X. Li and J. N. D. Gupta, Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search, Expert Systems with Applications, 42 (2015), 1409–1417.
[37] C. Liu, J. Wang and J. Y. T. Leung, Worker assignment and production planning with learning and forgetting in manufacturing cells by hybrid bacteria foraging algorithm, Computers & Industrial Engineering, 96 (2016), 162–179.
[38] K. F. Man, K. S. Tang and S. Kwong, Genetic Algorithms: Concepts and Designs, Springer-Verlag London, Ltd., London, 1999.
[39] B. L. Miller and D. E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex Systems, 9 (1995), 193–212.
[40] J. Miltenburg, One-piece flow manufacturing on U-shaped production lines: A tutorial, IIE Transactions, 33 (2001), 303–321.
[41] D. Naso, M. Surico, B. Turchiano and U. Kaymak, Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete, European Journal of Operational Research, 177 (2007), 2069–2099.
[42] F. Niakan, A. Baboli, T. Moyaux and V. Botta-Genoulaz, A new multi-objective mathematical model for dynamic cell formation under demand and cost uncertainty considering social criteria, Applied Mathematical Modelling, 40 (2016), 2674–2691.
[43] B. A. Norman, W. Tharmmapornphilas, K. L. Needy, B. Bidanda and R. C. Warner, Worker assignment in cellular manufacturing considering technical and human skills, International Journal of Production Research, 40 (2002), 1479–1492.
[44] M. W. Park and Y. D. Kim, A systematic procedure for setting parameters in simulated annealing algorithms, Computers & Operations Research, 25 (1998), 207–217.
[45] D. Quagliarella, Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications, World John Wiley & Son Ltd., 1998.
[46] S. Ramesh, S. Kannan and S. Baskar, Application of modified NSGA-II algorithm to multi-objective reactive power planning, Applied Soft Computing, 12 (2012), 741–753.
[47] I. Ribas and R. Companys, Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization, Computers & Industrial Engineering, 87 (2015), 30–39.
[48] S. I. Satoglu, M. B. Durmusoglu and T. Ertay, A mathematical model and a heuristic approach for design of the hybrid manufacturing systems to facilitate one-piece flow, International Journal of Production Research, 48 (2010), 5195–5220.
[49] S. I. Satoglu and N. C. Suresh, A goal-programming approach for design of hybrid cellular manufacturing systems in dual resource constrained environments, Computers & Industrial Engineering, 56 (2009), 560–575.

[50] J. E. Schaller, J. N. D Gupta and A. J. Vakharia, Scheduling a flowline manufacturing cell with sequence dependent family setup times, European Journal of Operational Research, 125 (2000), 324–339.

[51] T. Sousa, T. Soares, H. Morais, R. Castro and Z. Vale, Simulated annealing to handle energy and ancillary services joint management considering electric vehicles, Electric Power Systems Research, 136 (2016), 383–397.

[52] G. A. Suer and O. Alhwari, Operator assignment decisions in a highly dynamic cellular environment, Operations Management Research and Cellular Manufacturing Systems: Innovative Methods and Approaches, (2011), 258–294.

[53] G. A. Suer and C. Dagli, Intra-cell manpower transfers and cell loading in labor-intensive manufacturing cells, Computers & Industrial Engineering, 48 (2005), 643–655.

[54] G. A. Suer and R. R. Tummaluri, Multi-period operator assignment considering skills, learning and forgetting in labour-intensive cells, International Journal of Production Research, 46 (2008), 469–493.

[55] R. Suri, Quick response manufacturing: A competitive strategy for the 21st century, In Proceedings of the 2002 POLCA Implementation Workshop, 141 (2002).

[56] E. G Talbi, Metaheuristics: From Design to Implementation, John Wiley & Son Ltd., 2009.

[57] M. Unal, A. Ak, V. Topuz and H. Erdal, Optimization of PID Controllers Using ant Colony and Genetic Algorithms, Springer, Heidelberg, 2013.

[58] S. Venkataramanaiah, Scheduling in cellular manufacturing systems: An heuristic approach, International Journal of Production Research, 46 (2008), 429–449.

[59] J. X. Wang, Cellular Manufacturing: Mitigating Risk and Uncertainty, CRC Press, Boca Raton, USA, 2015.

[60] U. Wemmerlöv and N. L. Hyer, Cellular manufacturing in the US industry: A survey of users, International Journal of Production Research, 27 (1989), 1511–1530.

[61] J. Xu, X. Xu and S. Q. Xie, Recent developments in Dual Resource Constrained (DRC) system research, European Journal of Operational Research, 215 (2011), 309–318.

[62] O. F. Yilmaz, E. Cevikcan and M. B. Durmusoglu, Scheduling batches in multi hybrid cell manufacturing system considering worker resources: A case study from pipeline industry, Advances in Production Engineering & Management, 11 (2016), 192–206.

[63] O. F. Yilmaz, B. Oztaysi, M. B. Durmusoglu and S. C. Oner, Determination of material handling equipment for lean in-plant logistics using fuzzy analytical network process considering risk attitudes of the experts, International Journal of Industrial Engineering: Theory, Applications and Practice, 24 (2017), 81–122.

Received August 2016; revised June 2017.

E-mail address: omer.yilmaz@yalova.edu.tr
E-mail address: durmusoglu@itu.edu.tr