Int J Adv Manuf Technol (2015) 77:1907–1915
DOI 10.1007/s00170-014-6511-2

ORIGINAL ARTICLE

Concurrent scheduling of manufacturing cells considering sequence-dependent family setup times and intercellular transportation times

Kourosh Halat · Reza Bashirzadeh

Received: 17 June 2014 / Accepted: 16 October 2014 / Published online: 23 November 2014

Abstract This paper deals with the problem of scheduling operations in cellular manufacturing systems. A concurrent approach is proposed for job shop cell scheduling while the objective is to minimize the makespan. An integer linear programming model is developed by considering exceptional elements, intercellular moves, intercellular transportation times, and sequence-dependent family setup times. A modification of the model with reduced number of functional constraints is then presented to enhance the efficiency of the model in terms of computational time. In order to efficiently solve real-size problems in a reasonable amount of time, a heuristic approach based on the genetic algorithm (GA) is also developed. The proposed GA is enhanced with problem-specific knowledge of the search space by using a specialized repair strategy, a decoding procedure based on the concept of active schedules and a problem-specific mutation operator which prevents the algorithm from searching redundant solutions. Results reveal that the modified mixed integer linear programming model significantly reduces the computational time and the proposed GA is capable of providing promising solutions to large-scale problems.

Keywords Cellular manufacturing systems · Scheduling · Intercellular transportation times · Exceptional elements · Genetic algorithm

K. Halat (✉) · R. Bashirzadeh
Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran
e-mail: k.halat@outlook.com

R. Bashirzadeh
e-mail: rbashirzadeh@kntu.ac.ir

1 Introduction

Cellular manufacturing (CM), as an application of group technology, is a production system which deals with the formation of manufacturing cells in a way that parts requiring similar production processes are assigned to same manufacturing cells [31]. There are several advantages associated with cellular manufacturing system (CMS) including reduction in inventory level, setup times, and cycle time as well as enhancement in quality control and flexibility Shankar and Vrat [25]. Implementing effective scheduling system is an important requisite for attaining the potential benefits of CMS [10]. The previous research devoted to cellular scheduling can be categorized into two main groups: flow line cell scheduling and job shop cell scheduling.

Most studies have been focused on the flow line cell scheduling in CMS. Sridhar and Rajendran [28] have developed a hybrid simulated annealing (SA) algorithm to deal with the problem of scheduling operations in a flow line cell with the objective of minimizing total flow time. Skorin-Kapov and Vakharia [26] have proposed a tabu search (TS) algorithm to solve the flow line cell scheduling while the objective is to minimize the makespan. They have shown that the proposed algorithm outperforms SA-based algorithm in terms of solution quality and computational time. Frazier [7] has investigated the performance of 14 different scheduling rules in flow line cell scheduling. Results show that the simple rules have better performances and the performance of the rules is dependent to set up to runtime ratio. Sridhar and Rajendran [29] have proposed a genetic algorithm (GA) for scheduling in flow line cells with multiple objectives, namely, minimizing makespan, total flow time, and machine idle time. They have shown that the proposed algorithm outperforms other multi-criterion heuristics available in the literature. Schaller et al. [23] have proposed heuristic algorithms

based on TS, GA, and two well-known heuristic algorithms in flow shop scheduling, i.e., NEH and CDS to solve the problem of scheduling in flow line-based CMS. The objective is to minimize the makespan and the family setup times are sequence independent. Reddy and Narendran [19] have developed heuristic algorithms to schedule jobs within a part family in a flow line manufacturing cell with the objective of improving the utilization of machines and to reduce the tardiness and the number of tardy jobs. França et al. [6] have developed evolutionary-based algorithms including GA and memetic algorithm (MA) to solve the problem with the objective of minimizing makespan considering sequence-dependent family setup times. Hendizadeh et al. [8] have developed metaheuristic algorithms based on TS to minimize makespan in a flow line manufacturing cell. The proposed algorithms exploit the concepts elitism and acceptance of worse moves form SA-based algorithm. Lin et al. [12] have studied the performance of non-permutation schedules in flow line cells. They have proposed and evaluated the effectiveness of three metaheuristic algorithms including SA, GA, and TS. Zandieh et al. [32] have proposed two metaheuristic algorithms based on GA and SA to solve the problem of scheduling part families and jobs in flexible flow line manufacturing cell in which one or more stages have two or more units of the same machine type. Results reveal that the proposed GA outperforms SA. Karimi et al. [11] have investigated the problem of scheduling in flexible flow line cell with sequence-dependent family setup times while the objectives are minimizing makespan and total weighted tardiness. They have proposed a three-phase algorithm based on GA to solve the problem. Celano et al. [3] have studied the problem of scheduling in permutation flow line cells with sequence-dependent family setup times while the objective is to minimize makespan and the interoperational buffer capacity is limited. They have proposed a GA to solve the problem and compared its performance with NEH and TS. Results show that the proposed approach improves line productivity in a company producing electronic devices. Behnamian et al. [2] have developed a hybrid metaheuristic algorithm based on particle swarm optimization (PSO), SA, and variable neighborhood search (VNS) to solve the flexible flow line cell scheduling problem. The objective is to minimize the sum of earliness and tardiness. They have shown the superiority of the proposed hybrid algorithm over GA and SA. Salmasi et al. [22] have proposed a mathematical formulation of flow line cell scheduling with sequence-dependent family setup times while the objective is to minimize total flow time. They have developed a TS algorithm and a hybrid ant colony optimization (ACO) algorithm to solve the problem. Results show that the proposed HACO algorithm outperforms TS. Solimanpur and Elmi [27] have proposed a mixed integer linear programming model for the flow line cellular manufacturing system considering bottleneck machines and intercellular moves with makespan criterion. They have developed a nested TS algorithm to solve the problem.

Some researchers have investigated the problem of scheduling operations in job shop cells. Mahmoodi et al. [15] have proposed dynamic scheduling heuristics for job shop cells emphasizing good due date performance and reducing overall setup times. Simulation results show that the proposed heuristics improve the performance of the CMS. Mahmoodi and Dooley [13] have proposed non-exhaustive heuristics for scheduling operations in a job shop cell. Simulation results show the superiority of the exhaustive heuristics over the non-exhaustive ones. Ruben et al. [21] have investigated the performance of single-stage and two-stage group scheduling heuristics in a job shop cell environment. They have also examined the impacts of setup to runtime ratio and cell load combined with the inter-arrival time variability factor. Simulation results reveal that in many situations, the performances of single-stage heuristics are similar to that of two-stage heuristics. Mahmoodi and Martin [14] have proposed a subfamily queue selection heuristic for the job shop cell scheduling problem while the objective is to minimize total major sequence-dependent setup times. The proposed heuristic is based on dynamically assessing variations in a subfamily's arrival rate. The efficiency of the heuristic in terms of flow time and due date performance has shown through simulation. Chen and Lee [4] have developed an exhaustive group scheduling algorithm based on bottleneck machines to improve load balance in job shop cells. Results indicate the advantages of the proposed procedure in a variety of experimental conditions. Tavakkoli-Moghaddam et al. [30] have proposed a nonlinear mathematical model for the problem of scheduling operations in a CMS considering inter-cell moves. The objectives are to minimize the makespan, intracellular moves, tardiness, and setup costs. They have also developed a scatter search (SS) algorithm to solve the problem. Elmi et al. [5] have proposed an integer linear programming model for the problem of scheduling operations in job shop CMS considering inter-cell moves with the objective of minimizing makespan. They have developed an SA-based algorithm with a block-based neighborhood structure to solve the problem. Table 1 provides a summary of the research in the area of cellular scheduling. Renna and Ambrico [20] have proposed three different models to deal with the design, reconfiguration, and scheduling of the CMS. They have considered the turbulent market conditions by developing a scenario-based mathematical model for designing the CMS. The proposed model is able to manage the reconfiguration of the machines in the manufacturing system at fixed time periods. Paydar et al. [17] have proposed

Table 1 Summary of the research in the area of cellular scheduling

No.	Reference	Problem	Criterion	Solution method	Descriptions
1	Sridhar and Rajendran [28]	Permutation flow line	Total flow time	Heuristic, SA	
2	Skorin-Kapov and Vakharia [28]	Permutation flow line	Makespan	TS	Sequence-independent setup times
3	Frazier [7]	Permutation flow line	Multi-criteria	Dispatching rules	
4	Sridhar and Logendran [29]	Permutation flow line	Makespan, flow time, idle time	GA	
5	Schaller [23]	Permutation flow line	Makespan	Heuristic, GA, TS	Sequence-independent setup times
6	Redddy and Narendran [19]	Permutation flow line	Time in system, tardiness, No. of tardy jobs	Non-exhaustive heuristics	Poisson job arrival
7	Franca et al. [6]	Permutation flow line	Makespan	GA, MA	Sequence-dependent setup times
8	Hendizadeh et al. [8]	Permutation flow line	Makespan	TS	Sequence-dependent setup times
9	Lin et al. [12]	Non-permutation flow line	Makespan, tardiness	SA, GA, TS	Sequence-dependent setup times
10	Zandieh et al. [32]	Flexible flow line	Makespan	SA, GA	Sequence-dependent setup times
11	Karimi et al. [11]	Flexible flow line	Makespan, total weighted tardiness	GA	Sequence-dependent setup times
12	Celano et al. [3]	Permutation flow line	Makespan	GA	Sequence-dependent setup times
13	Behnamian et al. [2]	Flexible flow line	Sum of earliness and lateness	PSO, SA, VNS	Sequence-dependent setup times
14	Salmasi et al. [22]	Permutation flow line	Total flow time	TS, ACO, Branch & Price	
15	Solimanpur and Elmi [27]	Non-permutation flow line	Makespan	TS	
16	Mahmoodi et al. [15]	Job shop	Due date, total setup times	Heuristic	Poisson job arrival
17	Mahmoodi and Dooly [13]	Job shop	Multi-criteria	Heuristic	Computer simulation
18	Ruben et al. [21]	Job shop	Multi-criteria	Heuristic	Computer simulation
19	Mahmoodi and Martin [14]	Job shop	Due date, flow time	Heuristic	Poisson job arrival
20	Chen and Lee [4]	Job shop	Makespan	Heuristic	Computer simulation
21	Tavakkoli-Moghaddam et al. [30]	Job shop	Makespan, intracellular moves, tardiness, setup costs	SS	Non-linear model
22	Elmi et al. [5]	Job shop	Makespan	SA	

a mathematical model for dynamic cellular manufacturing systems considering alternate process routings, sequence of operations, intra-cell layout, and duplicate machines. The objective is to minimize the total costs of material handling, cell reconfiguration, outsourcing, inventory holding, machine operating, and maintenance.

As can be seen from Table 1, most of the researches available in the literature have been devoted to flow line cell scheduling. Furthermore, very few studies have investigated job shop cell scheduling problem by considering exceptional elements and intercellular moves. Family setup times are one of the most important features of CMS which have great effects on the operations planning. Furthermore, exceptional elements are almost always inevitable in the CMS and profoundly affect the cellular scheduling. Due to the fact that the exceptional elements are transported among cells, the operations schedule of one cell is dependent to other cells. Thus, solving the cellular scheduling problems independently results in schedules which are not practically applicable to the CMS. Additionally, in most scheduling problems, the transportation times of parts are often considered to be negligible. In the CMS, although the intracellular transportation times can be ignored, intercellular transportation times are usually notable and significant. To the best of our knowledge, no work has presented a linear programming model to solve this problem regarding exceptional elements, intercellular moves, intercellular transportation times, and sequence-dependent family setup times. Considering these factors necessitates concurrent scheduling of manufacturing cells in CMS.

In this paper, a concurrent approach is proposed for scheduling operations in job shop cells with the objective of minimizing makespan. First, an integer linear programming model is developed in which exceptional elements, intercellular moves, intercellular transportation times, and sequence-dependent family setup times are considered. Then, an extension of the model with fewer functional constraints is proposed to enhance the efficiency of the model and reduce its computational time. Furthermore, a GA-based heuristic is developed to efficiently solve large-scale problems in a reasonable amount of time. In the proposed GA, a specialized repair strategy, a decoding procedure based on the concept of active schedules, and a problem-specific mutation operator for omitting redundant solutions from the search space are applied. Results reveal that the extended model significantly reduces the computational time and the proposed GA is able to provide promising solutions to real-size problems.

2 Mathematical models

In this section, an integer linear programming model for concurrent scheduling of job shop manufacturing cells is proposed. In the proposed model, the effects of exceptional elements, intercellular moves, intercellular transportation times, and sequence-dependent family setup times are considered. The objective is to minimize the makespan. The following definitions and notations are considered in the mathematical models.

i	Index for machines i=1,…, m
j	Index for jobs j=1,…, n
k	Index for cells k=1,…, K
l	Index for part families l=1,…, h
o	Index for operations o=1,…, O_j
m	Number of machines
n	Number of jobs
K	Number of cells
O_j	Number of operations of part j
p_{ij}	Processing time of part j on machine i
$t_{kk'}$	Transportation time from cell k to cell k′
$s_{ll'}$	Setup time for part family l' if processed immediately after part family l
x_{ik}	1, if machine i is located in cell k and 0 otherwise
y_{jl}	1, if job j belongs to part family l and 0 otherwise
r_{joi}	1, if operation o of job j is processed on machine i and 0 otherwise
C_{ji}	Completion time of job j on machine i
C_{max}	makespan
$z_{jj'i}$	1, if job j precedes job j' on machine i and 0 otherwise
w_{ji}	1, if job j is processed on machine i and 0 otherwise

The proposed model is as follows.

$$\text{Minimize } C_{\max} \tag{1}$$

Subject to:

$$\sum_{i=1}^{m} r_{joi} C_{ji} \geq \sum_{i'=1}^{m} r_{j,o-1,i'} \left(C_{ji'} + p_{ji'} + \sum_{k=1}^{K} \sum_{k'=1}^{K} x_{ik} x_{i'k'} t_{kk'} \right) \quad \forall j, o \geq 2 \tag{2}$$

$$C_{ji} \geq C_{j'i} + \sum_{l=1}^{h} \sum_{l'=1}^{h} y_{j'l'} y_{jl} s_{l'l} - M z_{jj'i} + p_{ji} - M\left(2 - w_{ji} - w_{j'i}\right) \quad \forall i, j, j' \tag{3}$$

$$C_{j'i} \geq C_{ji} + \sum_{l=1}^{h} \sum_{l'=1}^{h} y_{jl} y_{j'l'} s_{ll'} - M\left(1 - z_{jj'i}\right) + p_{j'i} - M\left(2 - w_{ji} - w_{j'i}\right) \quad \forall i, j, j' \tag{4}$$

$$C_{ji} \geq p_{ji} \quad \forall i, j \tag{5}$$

$$C_{\max} \geq C_{ji} \quad \forall i, j \tag{6}$$

$$z_{jj'i},\ w_{ji},\ w_{j'i} = 0 \text{ or } 1 \quad \forall i, j, j' \tag{7}$$

Equation (1) is the objective function which minimizes makespan. Equation (2) ensures that each part is processed based on the defined precedence of its operations and intercellular transportation times. Equations (3) and (4) guarantee that at most, one part is processed by each machine at a time considering family setup times. Equation (4) imposes that completion time of an operation is not less than its processing time. Equation (6) calculates the makespan and Eq. (7) defines binary variables.

By defining a new variable $q_{jj'i}$ as the surplus variable of Eq. (3), it can be rewritten as follows.

$$M z_{jj'i} + \left(C_{ji} - C_{j'i}\right) - \sum_{l=1}^{h} \sum_{l'=1}^{h} y_{j'l'} y_{jl} s_{l'l} - p_{ji} + M\left(2 - w_{ji} - w_{j'i}\right) = q_{jj'i} \tag{8}$$

Accordingly, Eq. (4) is rewritten as follows.

$$q_{jj'i} \leq 5M - p_{ji} - p_{j'i} - \sum_{l=1}^{h}\sum_{l'=1}^{h} y_{j'l'} y_{jl} s_{l'l} - \sum_{l=1}^{h}\sum_{l'=1}^{h} y_{jl} y_{j'l'} s_{ll'} - 2Mw_{ji} - 2Mw_{j'i} \quad (9)$$

Now, by replacing Eqs. (3) and (4) with Eqs. (8) and (9), an extension of the proposed model is obtained. In the extended model, upper-bound constraints of type of Eq. (9) are added to the model while functional constraints of type of Eq. (4) are omitted by increasing the number of variables. Upper-bound constraints are known to be handled efficiently by the bounded simplex algorithm.

3 Proposed genetic algorithm

The problem of scheduling manufacturing cells involves a complex shape of search space. Since job shop scheduling problem (JSP) is a particular case of this problem, the complexity of the problem under consideration is at least of the same order of JSP. Recall that JSP is NP-hard when the number of machines exceeds 2. Thus, the problem of scheduling manufacturing cells is NP-hard as well. Due to the complexity of the problem, a GA is developed to efficiently search the solution space and find optimal or near-optimal solutions to the problem. GAs are known to be effective search techniques that have been successfully applied to solve different optimization problems. However, when the search space is extremely large, as it is the case for the problem of scheduling manufacturing cells, the efficiency of a common search technique such as GA highly depends on the incorporating problem-specific knowledge to the search algorithm. In the proposed GA, a specialized repair strategy, a decoding procedure based on the concept of active schedules, and a problem-specific mutation operator for omitting redundant solutions from the search space are applied. The main concepts of the proposed algorithm are discussed below.

3.1 Representation and feasibility

Representation is the first of GA implementation in which a typical solution is transformed to the format of a chromosome string. In the proposed algorithm, an operation-based representation scheme is adopted. The chromosome is encoded as permutation of integers from 1 to total number of operations to be scheduled in a way that each gene stands for an operation. Using permutation-based representation makes the chromosome adaptable to different operators and easily comprehensible [16]. For example, consider the problem of scheduling 21 operations of seven parts to be processed on six machines while the number of cells and the number of part families are two. Process routings of parts are shown in Fig. 1 in which rows and columns of the matrix stand for machines and parts, respectively.

Figure 2 presents an instance chromosome for this problem which is a permutation from 1 to 21.

The chromosome presented in Fig. 2 is a feasible chromosome because it does not violate the defined precedence of operations. However, a not-all permutation-based chromosomes are feasible. It is possible that a permutation be inconsistent with the process routings of the parts; thus, the resulted chromosome is infeasible. In the case of infeasibility, a straightforward repair strategy is applied. The stepwise procedure is as follows.

Step 1. Consider the first part and *do*

- 1–1) Determine all operations related to the part
- 1–2) Reorder the operations based on the process routing of the part
- 1–3) Replace the determined operations in the chromosome with the ordered ones

Step 2. If all parts have been considered, stop; else, consider the next operation and repeat 1–1 to 1–3

3.2 Initialization and fitness evaluation

Initialization is the process of generating an initial population with a desired population size. In the proposed algorithm, the initial population is a set of randomly generated solutions. If a randomly generated solution is infeasible, the repair strategy described in Section 3.1 is utilized to guarantee the feasibility of solutions.

For individual x, fitness function is calculated as follows.

$$f(x) = G - C_{\max}(x) \quad (10)$$

where $C_{\max}(x)$ is the makespan calculated for individual x and G is the maximum makespan in the current population. In order to calculate the makespan, the sequence represented by the chromosome must be decoded into a feasible schedule. A simple way to obtain the schedule is to directly put the

	P1	P2	P3	P4	P5	P6	P7
M1	0	0	3	1	3	0	0
M2	1	1	0	0	0	3	4
M3	0	0	1	0	2	0	0
M4	0	0	4	2	1	0	3
M5	2	3	2	0	0	2	1
M6	0	2	0	0	0	1	2

Fig. 1 Process routings of parts

18	6	19	3	12	4	15	10	1	5	7	13	16	11	8	14	20	17	21	9	2

Fig. 2 An example of a feasible chromosome

operations in the schedule according to their order in the chromosome. The resulted schedule will then be a semi-active schedule in which no operation can be processed earlier unless the order of processing on machines is changed. The set of semi-active schedules is very large and low quality in terms of makespan. The performance of the algorithm can be dramatically enhanced if the search process is guided to explore a smaller set of schedules without excluding the optimal solution. The set of active schedules is a small subset of semi-active schedules which includes the optimal solution. A schedule is active if no job can be processed earlier without delaying the process of any other job Pinedo [18]. In the proposed algorithm, a decoding procedure is applied to obtain active schedules from a chromosome. The procedure is a modified version of the algorithm proposed by Arkat et al. [1]. The procedure is based on finding the idle time intervals on machines before putting an operation into the schedule. For each machine, a set of gaps based on idle time intervals is defined and then updated whenever an operation is appended to the schedule.

Notation:

o_{ij} Operation of part j which is processed on machine i
p_{ij} Processing time of operation o_{ij}
r_j Earliest possible time for processing part j
MG_i Set of gaps on machine i

Algorithm:

Step 1 Set $MG_i=[0, +\infty]$ and $r_j=0$ for all parts
Step 2 for $k=1$ to total number of operations do
2–1) Consider operation o_{ij} at kth position of the chromosome
2–2) Find $[e, f]\in \mathrm{MG}_i$ such that $\min\{f-r_j, f-e\}\geq t_{[e, f]}+p_{ij}$ where, $t_{[e, f]}$ is family setup time needed for part j if processed in gap $[e, f]$
2–3) Among gaps found in step 2–2 select $[e^*, f^*]$ with earliest e
2–4) Set $e^*+t_{[e^*, f^*]}$ as the start time and $e^*+p_i+t_{[e^*, f^*]}$ as the completion time of operation o_{ij}
2–5) Remove $[e^*, e^*+p_i+t_{[e^*, f^*]}]$ from MG_i and update r_j for all parts considering intercellular transportation times

3.3 Selection

The selection mechanism in the proposed algorithm is based on the roulette wheel approach. In the roulette wheel selection mechanism, as a fitness proportional approach, individuals with better fitness have higher chances to be carried over to the next generation. An elitist strategy is also adopted to keep the best solution of each generation and directly transform it to the next generation.

3.4 Crossover

Crossover is the main genetic operator which combines two selected parents from the population to produce two offspring. In the proposed algorithm, order crossover is applied which is suited to genes that represent permutation [9]. The main idea of order crossover is to preserve the relative order of operations in the parents and convey it to the offspring.

3.5 Mutation

Mutation operator keeps the diversity of the population by making subtle changes in the chromosomes. Mutation operator in the proposed algorithm is based on the swap mutation while preventing from searching redundant solutions. In swap mutation, two genes are selected randomly and their corresponding operations are exchanged. According to the representation scheme of the solutions in the proposed algorithm, simple swap mutation may result in the same solution despite the change in the chromosome. This fact, which two different chromosomes may yield in same solutions, is called redundancy. In order to increase the efficiency of the proposed algorithm, a procedure is applied to prevent searching redundant solutions. Redundancy can occur in two cases. The first case of redundancy is when the two selected operations belong to the same part. According to the repair procedure described in Section 3.1, exchanging the position of these two operations does not change the solution. The second case is when the processes of the selected operations are on different machines and none of the operations located between those operations in the chromosome string are processed on one of the machines which process the selected operations. In other words, suppose that operations o_{ij} and o_{uv} are selected. Redundancy occurs if (1) $j=v$ or (2) $i\neq u$ and all operations located between o_{ij} and o_{uv} in the chromosome are not processed on machines i or u. The procedure described below enhances the performance of the algorithm by omitting redundant solutions from the search area.

Table 2 Comparison between the proposed mathematical models

No.	No. of machines	No. of parts	No. of cells	No. of part families	Model 1		Model 2	
					Solution	Time (s)	Solution	Time (s)
1	6	7	2	2	111	1	111	1
2	5	7	2	3	187	1	187	1
3	6	8	3	2	175	12	175	5
4	7	8	2	3	146	4	146	2
5	6	8	2	3	133	3	133	1
6	6	8	3	3	157	2	157	1
7	6	10	3	2	142	17	142	5
8	5	10	2	3	218	265	218	142
9	6	10	2	2	146	18	146	4
10	6	12	2	2	188	71	188	39
11	6	12	3	2	174	25	174	13
12	10	12	4	4	103	28	103	8
13	10	15	3	3	119	31	119	15
14	15	15	4	4	114	33	114	19

Step 1 Set $R=1$

Step 2 While $R=1$, repeat the following sub-steps

2–1) Select operations o_{ij} and o_{uv} located in positions p_1 and p_2 at random

2–2) If $j=v$, go to 2–1

2–3) If $i=u$, set $R=0$; else

for $k=p_1+1$ to p_2-1

If the operation located in the kth position of the chromosome is processed on machines i or u, set $R=0$

Step 3 Exchange the position of operations o_{ij} and o_{uv}

4 Computational results

In this section, the results of two experiments are described. The first experiment is conducted to compare the performance of the two proposed mathematical models. The second experiment deals with investigating the efficiency of the proposed genetic algorithm.

4.1 Performance of the mathematical models

In this section, the performance of the first mathematical model is compared to the modified model in terms of computational time. The modified model refers to the model incorporating the technique of reducing functional constraints described in Section 2. Models have been solved using LINGO 8.0 software on 2.00-GHz PC with 2.00 GB of RAM. The numerical examples have been generated using a random generation procedure. The processing time of operations are selected from [2, 65] randomly. The process routing of each part is determined by randomly choosing machines and arranging them at random. Family setup times and intercellular transportation times are selected randomly from [5, 50] to [5, 70], respectively. The problem instances have been generated in a way to be consistent, as far as possible, with the real-world situations. The intervals, from which the random numbers are selected, have been considered wide enough to include different possible settings for the problem. This yields to a wide range of problem instances with different settings on which the efficiency of the proposed algorithm is tested. It is noteworthy that the real-world problems are usually more highly structured than randomly generated problems, which provides a computational advantage [24].

Fourteen small- and medium-sized problem instances have been generated and then solved by the proposed mathematical models. Table 2 shows the data related to the problem instances and the results obtained by the first model (model 1) and the model with reduced functional constraints (model 2).

As can be seen from Table 2, both mathematical models are capable of finding optimal solutions for the small- and medium-sized problem instances. However, there is a significant difference between the computational times and model 2 performs much better than model 1. The average improvement in computational time is 48 %. Thus, as expected, the

Table 3 Initial levels for the parameters

Parameters	Level (−)	Center	Level (+)
Population size	20	85	150
Crossover rate	0.75	0.85	0.95
Mutation rate	0.05	0.175	0.3
No. of generations	20	85	150

Table 4 Performance of the proposed GA over small and medium-sized instances

No.	Model 2		Proposed GA		
	Solution	Time (s)	Best	Average	Time (s)
1	111	1	111	111	1
2	187	1	187	187	1
3	175	5	175	175	1
4	146	2	146	146	1
5	133	1	133	133	1
6	157	1	157	157	1
7	142	5	142	142	2
8	218	142	218	218	5
9	146	4	146	146	1
10	188	39	188	188	3
11	174	13	174	174	2
12	103	8	103	103	1
13	119	15	119	119	2
14	114	19	114	114	2

model with reduced number of functional constraints has a better performance in terms of computation time.

4.2 Performance of the proposed GA

The proposed GA has been coded in MATLAB and run on a 2.00-GHz PC with 2 GB of RAM. The parameters of the algorithm, including population size, crossover rate, mutation rate, and number of generations, have been set using statistical methods. In order to examine the sensitivity of the performance of the algorithm to the parameters, a 2^4 factorial design with five runs at the center point has been conducted. The algorithm has been run on some randomly generated problem instances with different sizes. The initial values considered for the levels of the parameters are shown in Table 3.

For each problem instance, the average solution over 10 runs of the proposed algorithm has been obtained for experimental runs of the factorial design. According to the calculated standardized effects for different problem instances, the parameters that have greater effects on the GA performance have been identified. Then, these parameters have been examined more attentively in further experiments for setting the parameters. The following results have then been obtained: population size 50, crossover rate 0.9, mutation rate 0.2, and number of generations 50.

The proposed algorithm has been run 10 replicates for each small- and medium-sized problem instances. The best and the average of the solutions have been reported in Table 4.

As can be seen from Table 4, the proposed algorithm is able to find optimal solutions in all replicates in a very short amount of time. The key point to achieve such remarkable results is the use of a decoding procedure which reduces the search space without excluding the optimal solution along with a strategy that prevents the algorithm from searching redundant solutions. In order to evaluate the performance of the algorithm on real-sized problems, five large-scale problem instances have also been generated and solved by the proposed algorithm. These problems have also been solved using the model with reduced number of functional constraints (model 2) described in Section 2. Data related to large-sized examples and the obtained results are shown in Table 5. The LINGO software has been interrupted after 5 h. For the problems that the optimal solution has been obtained in 5 h, the solution is depicted by an asterisk. For other problem, the reported solution is an upper bound to the optimal solution.

According to the results shown in Table 5, the proposed algorithm is capable of finding the optimal solutions for the first two problem instances in a very little amount of time. Note that these problems have been optimally solved using the proposed mixed integer linear programming model with reduced number of functional constraints. For problems 3 to 5, the proposed model is not able to find the optimal solution in 5 h. The solution obtained by the proposed algorithm is always better than the upper bound found by the mathematical model. Furthermore, the average of solutions obtained over 10 runs of the algorithm is very close to the optimal solution. Thus, the proposed GA is a fast and robust solution method for the problem of concurrent scheduling of job shop cells.

Table 5 Performance of the proposed GA over large-sized instances

No.	No. of machines	No. of parts	No. of cells	Model 2		Proposed GA		
				Solution	Time (s)	Best	Average	Time (s)
1	18	25	5	137*	12963	137	137	22
2	18	27	5	189*	10748	189	189	19
3	20	25	5	157	18000	147	147.2	38
4	20	27	5	192	18000	187	188.1	41
5	22	27	5	184	18000	177	177.6	34

5 Conclusion

In this paper, a concurrent approach is proposed for job shop cell scheduling considering sequence-dependent family setup times and intercellular transportation times. An integer linear programming model with the objective of minimizing makespan has been proposed to solve the problem. The proposed model is able to consider the effects of exceptional elements, intercellular moves, intercellular transportation times, and sequence-dependent family setup times. A modification of the model with reduced number of functional constraints has also been presented to enhance the efficiency of the model in terms of computational time. A genetic algorithm (GA) enriched with problem-specific knowledge of the search space has then been proposed to deal with real-size problems. The proposed algorithm uses a specialized repair strategy, a decoding procedure based on the concept of active schedules, and a problem-specific mutation operator which prevents the algorithm from searching redundant solutions. Computational results show the superiority of the modified model in terms of computational time. Finally, the robustness and efficiency of the proposed GA has been shown through computational experiments on some large-scale problem instances.

References

1. Arkat J, Farahani MH, Ahmadizar F (2012) Multi-objective genetic algorithm for cell formation problem considering cellular layout and operations scheduling. Int J Comput Integr Manuf 25(7):625–635
2. Behnamian J, Zandieh M, Fatemi Ghomi SMT (2010) Due windows group scheduling using an effective hybrid optimization approach. Int J Adv Manuf Technol 46:721–735
3. Celano G, Costa A, Fichera S (2010) Constrained scheduling of the inspection activities on semiconductor wafers grouped in families with sequence-dependent set-up times. Int J Adv Manuf Technol 46: 695–705
4. Chen YC, Lee CE (2001) A bottleneck-based group scheduling procedure for job-shop cells. J Chin Inst Ind Eng 18(5):1–12
5. Elmi A, Solimanpour M, Topaloglu S, Elmi A (2011) A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. Comput Ind Eng 61: 171–178
6. França PM, Gupta JN, Mendes AS, Moscato P, Veltink KJ (2005) Evolutionary algorithms for scheduling a flow shop manufacturing cell with sequence dependent family setups. Comput Ind Eng 48(3): 491–506
7. Frazier GV (1996) An evaluation of group scheduling heuristics in a flow-line manufacturing cell. Int J Prod Res 34(4):959–976
8. Hamed Hendizadeh S, Faramarzi H, Mansouri SA, Gupta JN, ElMekkawy T (2008) Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. Int J Prod Econ 111(2):593–605
9. Haupt RL, Haupt SE (2004) Practical genetic algorithms. 2nd edition, Wiley
10. Hitomi K, Ham I (1976) Operations scheduling for group technology applications. Annals CIRP 25(1):419–422
11. Karimi N, Zandieh M, Karamooz HR (2010) Bi-objective group scheduling in hybrid flexible flowshop: a multi-phase approach. Expert Syst Appl 37:4024–4032
12. Lin SW, Ying KC, Lee ZJ (2009) Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times. Comput Oper Res 36(4):1110–1121
13. Mahmoodi F, Dooley KJ (1991) A comparison of exhaustive and non-exhaustive group scheduling heuristics in a manufacturing cell. Int J Prod Res 29(9):1923–1939
14. Mahmoodi F, Martin GE (1997) A new shop-based and predictive scheduling heuristic for cellular manufacturing. Int J Prod Res 35(2): 313–326
15. Mahmoodi F, Dooley KJ, Starr PJ (1990) An investigation of dynamic group scheduling heuristics in a job shop manufacturing cell. Int J Prod Res 28(9):1695–1711
16. Naderi B, Fatemi Ghomi SMT, Aminnayeri M, Zandieh M (2010) A contribution and new heuristics for open shop scheduling. Comput Oper Res 37:213–221
17. Paydar MM, Saidi-Mehrabad M, Kia R (2013) Designing a new integrated model for dynamic cellular manufacturing systems with production planning and intra-cell layout. Int J Appl Deci Sci 6(2): 117–143
18. Pinedo M (2002) Scheduling: theory, algorithms, and systems. Prentice Hall, New Jersey
19. Reddy V, Narendran TT (2003) Heuristics for scheduling sequence-dependent set-up jobs in flow line cells. Int J Prod Res 41(1):193–206
20. Renna P, Ambrico M (2014) Design and reconfiguration models for dynamic cellular manufacturing to handle market changes. Int J Comput Integr Manuf. doi:10.1080/0951192X.2013.874590
21. Ruben RA, Mosier CT, Mahmoodi F (1993) A comprehensive analysis of group scheduling heuristics in a job shop cell. Int J Prod Res 31(6):1343–1369
22. Salmasi N, Logendran R, Skandari MR (2010) Total flow time minimization in a flowshop sequence-dependent group scheduling problem. Comput Oper Res 37:199–212
23. Schaller J (2000) A comparison of heuristics for family and job scheduling in a flow-line manufacturing cell. Int J Prod Res 38(2): 287–308
24. Schmidt CW, Grossmann IE, Blau GE (1998) Optimization of industrial scale scheduling problems in new product development. Comput Chem Eng 22:1027–1030
25. Shankar R, Vrat P (1998) Post design modeling for cellular manufacturing system with cost uncertainty. Int J Prod Econ 55(1): 97–109
26. Skorin-Kapov A, Vakharia J (1993) Scheduling a flow-line manufacturing cell: a tabu search approach. Int J Prod Res 31(7): 1721–1734
27. Solimanpur M, Elmi A (2013) A tabu search approach for cell scheduling problem with makespan criterion. Int J Prod Econ 141: 639–645
28. Sridhar J, Rajendran C (1993) Scheduling in a cellular manufacturing system: a simulated annealing approach. Int J Prod Res 31(12):2927–2945
29. Sridhar J, Rajendran C (1996) Scheduling in flowshop and cellular manufacturing systems with multiple objectives—a genetic algorithm approach. Prod Plan Control 7(4):374–382
30. Tavakkoli-Moghaddam R, Javadian N, Khorrami A, Gholipour-Kanani Y (2010) Design of a scatter search method for a novel multi-criteria group scheduling problem in a cellular manufacturing system. Expert Syst Appl 37:2661–2669
31. Wemmerlov U, Hyer NL (1986) Procedures for the part family, machine group identification problem in cellular manufacturing. J Oper Manag 6(2):125–147
32. Zandieh M, Dorri B, Khamseh AR (2009) Robust metaheuristics for group scheduling with sequence-dependent setup times in hybrid flexible flow shops. Int J Adv Manuf Technol 43:767–778