Int J Adv Manuf Technol (2010) 48:701–717
DOI 10.1007/s00170-009-2322-2

ORIGINAL ARTICLE

Designing integrated cellular manufacturing systems with scheduling considering stochastic processing time

Vahidreza Ghezavati · Mohammad Saidi-Mehrabad

Received: 10 July 2009 / Accepted: 15 September 2009 / Published online: 1 October 2009

Abstract This paper addresses a new mathematical model for cellular manufacturing problem integrated with group scheduling in an uncertain space. This model optimizes cell formation and scheduling decisions, concurrently. It is assumed that processing time of parts on machines is stochastic and described by discrete scenarios enhances application of real assumptions in analytical process. This model aims to minimize total expected cost consisting maximum tardiness cost among all parts, cost of subcontracting for exceptional elements and the cost of resource underutilization. Scheduling problem in a cellular manufacturing environment is treated as group scheduling problem, which assumes that all parts in a part family are processed in the same cell and no inter-cellular transfer is needed. Finally, the nonlinear model will be transformed to a linear form in order to solve it for optimality. To solve such a stochastic model, an efficient hybrid method based on new combination of genetic algorithm (GA), simulated annealing (SA) algorithm, and an optimization rule will be proposed where SA and optimization rule are subordinate parts of GA under a self-learning rule criterion. Also, performance and robustness of the algorithm will be verified through some test problems against branch and bound and a heuristic procedure.

Keywords Cellular manufacturing · Uncertainty modeling · Stochastic processing time · Hybrid genetic algorithm · Cell scheduling · Stochastic programming

V. Ghezavati (✉) · M. Saidi-Mehrabad
Department of Industrial Engineering,
Iran University of Science and Technology,
P.C. 16844 Tehran, Iran
e-mail: ghezavati@iust.ac.ir

1 Introduction

Group technology (GT) is a management theory that aims to group products with similar process or manufacturing characteristics, or both. Cellular manufacturing (CM) can be proposed as a practical application of GT that determines groups of machines based on similarity of the parts processed by them. The basic purpose of CM is to identify machine cells and part families concurrently and to assign part families to machine cells in order to minimize the intercellular and intracellular costs of parts [1]. Scheduling jobs in individual cells is an operational feature that should be determined at the design stage. Because design stages are so difficult, however, integrating scheduling decisions with CF decision is often seen as just a more complication.

This review focuses on the uncertain studies that are relevant to the uncertainty planning of cellular manufacturing system (CMS) problems; however, a survey of certain conditions will be presented.

The literature on the design of cellular manufacturing system is quite extensive in certain and determined situations. Past researches in studying CMSs design and implementation have been predominantly focused on the CF decision in certain conditions. In the context of the research reported here, research work dealing with the uncertainty aspects of CMS design is presented.

A majority of cases studied CMS problem in uncertain situations can be classified into three branches: (1) fuzzy approach, (2) stochastic optimization, and (3) heuristic procedures. The most common planning approach developed to resolve uncertainty in CMS problems can be introduced as fuzzy approach with many researches proposed previously. Papaioannou and Wilson [2] proposed CMS problems analysis where coefficients in objective function and constraints are considered as fuzzy coeffi-

cients. Also, in an interesting research, Hentschel et al. [3] introduced a cellular recycling system in which processing time for parts due to fluctuation of products usage are uncertain and described by fuzzy parameters. Shanker and Vrat [4] in order to forecast future variations applied fuzzy approach in formulation process. Szwarc et al. [5] considered uncertainty in machines' capacity located in cells. In this way, in CMS problem, capacity of machines is uncertain and is resolved by fuzzy approach. Ravichandran and Chandra Sekhara Rao [6] regarded as fuzzy uncertainty in part-machine matrix. Based on this research, a new similarity coefficient matrix is defined with 0 to one elements.

Although stochastic programming (SP) is applicable in many areas in production planning, but in CMS problem, this approach is usually applied just for handling changes in demand while many other factors can be stochastic. Some researches deal with aggregated CMS problem with tactical decisions such as production planning where demand of products is stochastic are handled [7, 8]. Also, some studies such as [9] made layout decisions concurrently with CMS where demand was stochastic. Many contents are also focused on CMS problem in dynamic situations and stochastic demand in which demand is changed from a period to another period [10, 11]. Yang and Deane [12] presented a formulation in which set-up time was considered as a part of processing time. The aim of this problem was to decrease set-up time. In this study, since amount of set-up time decreased was uncertain, and then total processing time was uncertain, too. In order to analysis this problem they proposed a M/G/1 queuing model for the manufacturing framework. Another aspect which is considered stochastic in the literature is machines' availability during production plan. Some researchers considered CMS problem as a probabilistic programming in which the time between two sequence failures follows exponential distribution [13, 14]. Finally, Markov chain and queuing theory were applied to handle uncertainty in machines' availability [15, 16].

In the last branch, some studies have been developed to define heuristic methods of solving aggregated CMS problem in uncertain situations such as [10, 17]. In some cases, heuristic algorithms considering uncertainty in processing time are proposed by Asgharpour and Javadian [18] and Andres et al. [19].

Literature survey in certain conditions can be described as follows. There exist many researches in certain situations for designing CMS in different areas such as cell formation integrated with scheduling (Taylor and Ham [20], Logendran and Nudtasomboon [21], Solimanpur et al. [22], Aneja and Kamoun [23], Lockwood et al. [24]), considering exceptional elements in CF (Tsai et al. [25], Vakharia and Kaku [26], Mahdavi et al. [27]), some works apply meta-heuristics and heuristics to solve large scale problems are more practical and appealing real-case problems (Xiaodan Wu et al. [28], Venkataramanaiah [29], Sridhar and Rajendran, [30], Mahmoodi and Martin [31]).

Saad [32] proposed an integrated approach to redesign CM system considering emphasis on redesign aspects. His approach used simulation based on scheduling module.

Panchalavarapu and Chankong [33] formulated a version of CMS problem as a nonlinear programming incorporating with assembly considerations. The goal of their model was to determine assignment of parts, machines and in addition, subassemblies to manufacturing cells.

Table 1 points up summery of the literature reports.

In practice, costs, demands, processing times [3, 14, 34], set-up times, and the other inputs to classical CMS problems may be highly uncertain so that it can have impact on results sensitively. So, development models for cellular manufacturing problem under uncertainty can be novel area for researchers and belongs to a relatively new class of CMS and scheduling problems that were not researched well in the literature. In this way, random parameters can be either continues or described by discrete scenarios. If probability information is known, uncertainty is described using a (discrete or continuous) probability distribution on the parameters, otherwise, continuous parameters are normally limited to lie in some predetermined intervals. Scenario-based planning is an approach in which uncertainty is captured by determining a number of possible future states by decision makers. The goal is to find solutions which perform well under all scenarios [35].

Uncertainty in processing time can be applied for many real applications in advanced manufacturing systems and also described discrete or continues form based on the properties of the application and case study. For example, it may be possible that design of parts is changed during production process (postdesign definition [36]). So, these changes can vary design aspects and finally can fluctuate processing time in production plan. The main point is that these changes in design of parts are not certain events in the future. So, in order to design cellular manufacturing effectively changes must be predicted as some discrete scenarios at the beginning of the planning phase. Hence, in industries such as automobile manufacturing, this formulation can be applied. On the other hand, in some cases such as condition-based maintenance for a system where after each inspection based on the degree of deterioration maintenance is performed, the time needed for maintenance is uncertain [14]. Also, in an interesting research, Hentschel et al. [3] introduced a cellular recycling system in which processing time for parts due to fluctuation of products usage are uncertain and described by fuzzy parameters. Three last cases illustrate situations in which processing time is uncertain, and therefore, our decision process can be applied in real-world conditions.

In any SP problem using uncertain parameters, one must decide which decision variables are first stage and which

Table 1 Summery of the literature survey results

Reference	Year	Condition	Decision
Papaioannoua and Wilson [2]	2008	Fuzzy	Uncertainty in coefficients
Hentschel et al. [3]	1995	Fuzzy	Uncertainty in processing time
Shanker and Vrat [4]	1999	Fuzzy	Uncertainty in forecasting future variations
Szwarc et al. [5]	1997	Fuzzy	Uncertainty in machines' capacity
Ravichandran and Chandra Sekhara Rao [6]	2001	Fuzzy	Uncertainty in part-machine matrix
Song et al. [7]	1991	Stochastic	Uncertainty in demand of products
Hurley and Clay Whybark [8]	1999	Stochastic	Uncertainty in demand of products
Balakrishnan and Cheng [10]	2005	Stochastic	Uncertainty in dynamic demands
Balakrishnan and Cheng [11]	2007	Stochastic	Uncertainty in dynamic demands
Yang and Deane [12]	1993	Stochastic	Uncertainty in processing time
Kuroda and Tomita [13]	2005	Stochastic	Uncertainty in machines' availability
Hosseini [14]	2000	Stochastic	Uncertainty in machines' availability
Balakrishnan and Cheng [10]	2005	Uncertain	Heuristic approaches
Asgharpour and Javadian [17]	2004	Uncertain	Heuristic approaches
Sun and Yih [18]	1996	Uncertain	Heuristic approaches
Andres et al. [19]	2007	Uncertain	Heuristic approaches
Taylor and Ham [20]	1981	Uncertain	CF + GS
Logendram and Nudtasomboon [21]	1991	Certain	CF + GS
Solimanpur et al. [22]	2004	Certain	CF + GS
Lockwood et al. [24]	2000	Certain	CF + scheduling
Aneja and Kamoun [23]	1999	Certain	CF + GS
Tsai et al. [25]	1997	Certain	CF + EE
Vakharai and Kaku [26]	1993	Certain	CF + EE
Mahdavi et al. [27]	2008	Certain	CF + EE
Xiaodan Wu et al. [28]	2006	Certain	Meta-heuristics
Venkataramanaiah [29]	2007	Certain	Heuristic approaches
Sridhar, J. and Rajendran [30]	1993	Certain	Heuristic Approaches
Mahmoodi and Martin [31]	1997	Certain	Heuristic approaches

are second stage; that is, which variables must be set now and which may be set after the uncertainty has been resolved [34, 37]. In other words, which variables should be set at the beginning of the planning period and which of them must be set after uncertainty is realized. In stochastic CMS problem which we are interested, CF decisions must be made now (design variable), before it is known which scenario will come to pass due to its strategic impact, while scheduling decisions are determined in future after uncertainty has been realized (control variable).

This paper introduces a new mathematical model to design cellular manufacturing systems under uncertainty which is described by discrete scenarios, each with a specified probability of occurrence. In this model, total cost consists of expected maximum tardiness cost among parts, subcontracting and outsourcing costs for exceptional elements and underutilization costs. This model can trade off between subcontracting costs and scheduling costs for each operation of parts in order to determine cells of machines and part families. In other words, each operation for a part can be considered either to be outsourced, therefore, we pay subcontracting cost for it and so, it needn't to be reconsidered in group scheduling or it can be processed in cells and we should pay scheduling costs. We call this model as the stochastic cell formation problem (SCFP).

2 Model formulation

In this section, we describe a new mathematical model for SCFP which we are interested. It is assumed that processing time of parts on machines is uncertain and is described by discrete scenarios. We have a set of scenarios; each of them occurred with probability p_s. Due to having multiscenarios in the problem, we must have group scheduling for cells in each scenario which is based on processing time of the parts. In this formulation, we minimize total expected cost, included expected maximum tardiness costs among parts,

total cost of subcontracting costs for outsourcing exceptional elements, and the cost of resource underutilization that occurred when the parts, which have no need to be operated on a machine, were placed together in a same cell.

We consider the following assumptions in scheduling a problem:

1. All parts are available to process at the beginning of the planning period.
2. Due date of each part is determined and certain events.
3. While an operation starts on a machine, it cannot be interrupted before completion of the process.
4. Set-up time for parts is sequence independent and is considered as a portion of the processing time.
5. Machines are available during the planning period and they are not failing.

We consider the following assumptions in a cell formation phase:

1. Processing time for each part on each machine is stochastic and described by set of discrete scenarios where probability of occurring each scenarios is p_s.
2. Each part has a number of operations which is determined by machine-part matrix.
3. Cost of subcontracting and underutilization is known and deterministic.
4. Under utilization, cost will occur if machines will have low similarity located in a same cell.

2.1 Notation

Indexes

i Part index
j Machine index
k Cell index
s Scenario index

Parameters

a_{ij} $= \begin{cases} 1 & \text{If part i require to be processed on machine j.} \\ 0 & \text{Otherwise.} \end{cases}$
c_i Penalty cost of subcontracting for part i.
u_{ij} Cost part i not utilizing machine j.
$M_{\max}$ Maximum number of machines permitted in a cell.
C_u Maximum number of cells permitted.
p_s Probability of scenario s occurs.
t_{ijs} Processing time for part i on machine j in scenario s.
DD_i Due Date of part i.
pc Penalty cost for unit time delayed.

Decision variables

x_{ik} $= \begin{cases} 1 & \text{If part i processed in cell } k. \\ 0 & \text{Otherwise.} \end{cases}$

y_{jk} $= \begin{cases} 1 & \text{If machine j assigned to cell k.} \\ 0 & \text{Otherwise.} \end{cases}$

$Z_{is[r]}$ $= \begin{cases} 1 & \text{If part i assigned to sequence}[r] \text{ in scenario s.} \\ 0 & \text{Otherwise.} \end{cases}$

$F_{[r]ks}$ The time in which process of part with sequence $[r]$ ends in cell k and scenario s.
$FD_{[r]ks}$ Due date of part with sequence $[r]$ in cell k in scenario s.
$L_{[r]ks}$ Tardiness of part with sequence $[r]$ in cell k in scenario s.
ML_s Maximum tardiness occurred in scenario s.
D_{iks} Total processing times of part i needs to be processed in cell k and scenario s.
$T_{[r]ks}$ Total processing times of a part with sequence $[r]$ assigned to cell k in scenario s.

CF decisions are scenario independent—they must be made before occurring scenarios and, they are made based on similarity in processing parts and are independent to quantity of processing time. Scheduling decisions are scenario dependent, thus Z, D, T, FD, L, ML, and F variables are indexed by scenario since they should be made after we realized scenario and which processing time is occurred.

2.2 Model description

$$\text{Minimize } Z = \sum_s pc \times p_s \times ML_s + \sum_k \sum_j \sum_i c_i a_{ij} x_{ik}\left(1 - y_{jk}\right) + \sum_k \sum_j \sum_i u_{ij}\left(1 - a_{ij}\right) x_{ik} y_{jk} \quad (1)$$

Subject to:

$$\sum_k x_{ik} = 1 \qquad \forall i \quad (2)$$

$$\sum_k y_{jk} = 1 \qquad \forall j \quad (3)$$

$$\sum_r Z_{is[r]} = 1 \qquad \forall i, s \quad (4)$$

$$\sum_i x_{ik} Z_{is,[r+1]} \le \sum_i X_{ik} Z_{is[r]} \qquad \forall k, s, r \quad (5)$$

$$D_{iks} = \sum_j a_{ij} t_{ijs} x_{ik} y_{jk} \qquad \forall i, k, s \quad (6)$$

$$\sum_i x_{ik} Z_{is[r]} \le 1 \qquad \forall r,s,k \tag{7}$$

$$T_{[r]ks} = \sum_i Z_{is[r]} D_{iks} \qquad \forall k,s,r \tag{8}$$

$$F_{[r]ks} = \sum_{r=1} \sum_{\alpha=1}^{r} T_{\alpha ks} \qquad \forall k,s,r \tag{9}$$

$$FD_{[r]ks} = \sum_i x_{ik} \times Z_{is[r]} \times DD_i \qquad \forall k,s,r \tag{10}$$

$$L_{[r]ks} = \max\left\{0, F_{[r]ks} - FD_{[r]ks}\right\} \qquad \forall k,s,r \tag{11}$$

$$ML_s = \max\left\{L_{[r]ks} : k = 1,\ldots,C \text{ and } [r] = 1,\ldots,P\right\} \qquad \forall s \tag{12}$$

$$\sum_j y_{jk} \le M_{\max} \qquad \forall k \tag{13}$$

$$x_{ik}, y_{jk}, Z_{isr} \sim (0.1) \tag{14}$$

$$D_{iks},\ T_{rks},\ F_{rks},\ FD_{rks} \ge 0 \tag{15}$$

The objective function (1) minimizes total cost made of expected tardiness cost of parts in all scansions, subcontracting cost as well as the cost of resource underutilization. Set constraint (2) says that each part must be assigned to a single cell. Set constraint (3) states that each machine can be assigned only to one cell. Set constraint (4) ensures that each part in each scenario must be assigned to a unique sequence. Set constraint (5) indicates that in each cell and each scenario parts must be ordered sequential (for example, if part 'a' is assigned to sequence 1, next part must be assigned to sequence 2). Set constraint (6) computes total processing time of part i in cell k in scenario s. D_{iks} variable gets non-zero value when part i needs to be processed on machine j and both of them assigned to same cell. (Note that, this variable will be zero for the other cells which part is not assigned to them). Set constraint (7) guarantees that in each cell at most one part can be assigned to each sequence. Set constraint (8) computes total processing time of a part with sequence $[r]$ assigned to cell k in scenario s. Set constraint (9) computes total flow time (or the time in which all process are completed) of a part assigned to sequence $[r]$ in cell k in scenario s. Since in group scheduling definition processing time of all operations for each part are summed to make a single process (in other words, for each part all operations are assumed to be grouped into a single operation), so process of a part with sequence $[r]$ cannot be started unless grouped operation (or all operations) of a part with sequence $[r\text{-}1]$ is finished. Thus, a part with sequence $[r]$ has to wait in a cell (flow time) until all operations of parts with sequence [1] to $[r]$ are completed. Set constraint (10) computes due date of a part assigned to sequence $[r]$ in cell k in scenario s. set constraint (11) computes tardiness of parts in cells. Set constraint (12) specifies maximum tardiness per each scenario among all cells and parts. Set constraint (13) specifies maximum number of machines allowed in any cell. Set constraints (14) and (15) determine type of variables.

2.3 Linearization of the proposed model

Unfortunately, the proposed model is nonlinear, and nonlinear models are usually much harder to solve for optimality than linear models. We reformulate the model as a mixed-integer linear programming model by introducing new sets of variables. In this way, different types of nonlinear terms are appeared in formulation. In some terms such as objective function, set constraints (5), (6), (7), and (10), and (10) there are two binary variables which are multiplied (this problem is defined by quadratic 0–1 problem). In other cases such as set constraint (8), a continuous variable is multiplied to a binary variable (this problem in defined in the literature as mixed 0 to one quadratic problem). Also, set constraints (11) and (12) are nonlinear too. In each case, different approach will be applied to linearization.

2.3.1 Linearization quadratic 0-1 problem

Given a quadratic 0-1 term $z = x_1 \times x_2$ where x_1, x_2 are binary variables. This term forces that z must be 0 if and only if at least one x gets 0. On the other hand, z must be 1 if and only if both variables get 1. This term can be transformed to a set of linear auxiliary constraints. Based on this transformation, the original 0 to one quadratic program can then be solved directly by the branch-and-bound method. This nonlinearity is appeared in set constraints (5), (6), (7), and (10). To linearize this term applied auxiliary constraints are as follows:

$$z = x_1 \times x_2 \Leftrightarrow \begin{cases} z \le x_1 \\ z \le x_2 \\ z \ge x_1 + x_2 - 1 \end{cases}.$$

Proof) Glover [38].

2.3.2 Linearization quadratic mixed 0 to one problem

Given a quadratic mixed 0 to one term $z = x \times y$ where x is a binary variable and y is a continuous variable with upper bound units. This term can be transformed to a set of linear auxiliary constraints. Based on this transformation, the original mixed 0 to one quadratic program can then be solved directly by the branch-and-bound method. This nonlinear term is appeared in set constraint (8). To linearize this term applied auxiliary constraints are as follows:

$$z = x \times y \Leftrightarrow \begin{cases} z \le y \\ z \le U \times x \\ z \ge y - U(1-x) \end{cases}.$$

The proof is proposed by [39].

2.3.3 Linearization Max {0, x}

In order to linearize such term as Max {0, x}, an additional variable and two auxiliary constraints must be used to replace with this term. Such nonlinear term is appeared in set constraints (11) and (12). Linearization procedure is performed as follows:

$$\begin{array}{l} \text{Minimize} \quad Z \\ \text{ST}: \\ Z = \text{Max}\{0, x\} \end{array} \Rightarrow \begin{array}{l} \text{Minimize} \quad Z \\ \text{ST}: \\ Z \ge 0 \\ Z \ge x \end{array}$$

2.3.4 Linear model

To linearize the proposed model, above techniques are applied as follows:

- XK_{ijk} added to replace the x_{ik} and y_{jk} (quadratic pure 0 to one, technique 2.3.1 applied).
- XZ_{iksr} added instead of x_{ik} and Z_{isr} (quadratic pure 0 to one, technique 2.3.1 applied).
- ZD_{iksr} added to replace with Z_{isr} and D_{iks} (quadratic mixed 0 to one, technique 2.3.2 applied)
- Set constraints (11) and (12) will be linear employing technique proposed in 2.3.3

Thus, final version of the linear model presents as follows:

$$\text{Minimize } Z = \sum_s pc \times p_s \times ML_s + \sum_k \sum_j \sum_i c_i a_{ij}\left(x_{ik} - XY_{ijk}\right) + \sum_k \sum_j \sum_i u_{ij}\left(1 - a_{ij}\right) XY_{ijk} \tag{14}$$

Subject to: unaltered set constraints (2)–(3)–(4)–(9)–(11)–(12)–(13)

Set constraint (5) changes as:

$$XZ_{iks,[r+1]} \le XZ_{iks[r]} \qquad \forall i,k,s,r \tag{15}$$

$$XZ_{iks[r]} \le x_{ik} \qquad \forall i,k,s,r \tag{16}$$

$$XZ_{iks[r]} \le Z_{is[r]} \qquad \forall i,k,s,r \tag{17}$$

$$x_{ik} + Z_{is[r]} - XZ_{iks[r]} \le 1 \qquad \forall i,k,s,r \tag{18}$$

Set constraint (6) changes as:

$$D_{iks} = \sum_j a_{ij} t_{ijs} XY_{ijk} \qquad \forall i,k,s \tag{20}$$

$$XY_{ijk} \le x_{ik} \qquad \forall i,k,j \tag{21}$$

$$XY_{ijk} \le y_{jk} \qquad \forall i,k,j \tag{22}$$

$$x_{ik} + y_{jk} - XY_{ijk} \le 1 \qquad \forall i,k,j \tag{23}$$

Set constraint (7) changes as:

$$\sum_i XZ_{iks[r]} \le 1 \qquad \forall r,s,k \tag{24}$$

Set constraint (8) changes as:

$$T_{[r]ks} = \sum_i ZD_{iks[r]} \qquad \forall k,s,r \tag{25}$$

$$ZD_{iks[r]} \le D_{iks} \qquad \forall i,k,s,r \tag{26}$$

$$ZD_{iks[r]} \le M \times Z_{is[r]} \qquad \forall i,k,s,r \tag{27}$$

$$ZD_{iks[r]} \ge D_{iks} - M\left(1 - Z_{is[r]}\right) \qquad \forall i,k,s,r \tag{28}$$

Set constraint (10) changes as:

$$FD_{[r]ks} = \sum_i XZ_{iks[r]} \times DD_i \qquad \forall k,s,r \tag{29}$$

Set constraint (11) changes as:

$$L_{[r]ks} \geq F_{[r]ks} - FD_{[r]ks} \qquad \forall r,k,s \tag{30}$$

Set constraint (12) changes as:

$$ML_s \geq L_{[r]ks} \qquad \forall r,k,s \tag{31}$$

$$L_{[r]ks} \geq 0 \qquad ML_s \geq 0 \tag{32}$$

In the above linear model, new set constraints (21), (22), and (23) guarantee linearization of $x_{ik} \times y_{jk}$, new set constraints (16), (17), and (18) guarantee linearization of $x_{ik} \times Z_{isr}$. Also, new set constraints (26), (27), and (28) guarantee linearization of $Z_{isr} \times D_{iks}$. Finally, new set constraints (30), (31), and (32) guarantee linearization of constraints (11) and (12). The other altered and no altered constraints are provided as mentioned. Also, M is a positive large number.

3 Solution approach

It is known that cellular manufacturing problems are nondeterministic polynomial-time (NP)-hard that cannot be solved in reasonable computational time. In this paper, we proposed a nonlinear and stochastic model for the cellular manufacturing design integrated with scheduling. By knowing that, scheduling problem is extremely hard to optimally solve in large sizes. Thus, we apply aggregation of genetic algorithm and simulated annealing methods under an optimization rule (earliest due date) and also a novel self-learning rule (SLR) to achieve the best solutions for such complicated model. In this way, employment of earliest due date (EDD) rule, an optimization technique in scheduling theory as a subordinate part of hybrid genetic algorithm simplify decision making in scheduling phase. Also, employment of a novel self-learning rule intensifies efficiency of the aggregation between genetic algorithm (GA) and simulated annealing (SA). Generally, genetic algorithm defines the main framework for this algorithm. Also, both simulated annealing and EDD rule work as subordinate parts of the genetic algorithm in order to improve performance of the approach under a novel self-learning rule.

3.1 Optimization rule: earliest due date

Since scheduling our objective is to compute maximum tardiness of parts in all cells and scenarios and try to minimize it, an optimization method for scheduling is considered as a subordinate part of hybrid method. This rule is based on earliest due date technique approved by theory of neighbor pairs and minimizes maximum tardiness among all parts. This rule is applied as: in order to minimize maximum tardiness among parts, sequence of parts in each cell must be planned so that the part with the earliest due date will be processed earlier.

Suppose that in a given solution, if n parts are assigned to the same cell, a total situation for scheduling decisions is $n!$ where from this quantity only one of them is optimal for that given assignment. Therefore, if scheduling is determined randomly complete, the best solution in scheduling will be found with probability of one/$n!$. So, if we apply EDD rule to make decisions in scheduling phase, probability of achieving the best solution in scheduling will increase from one/$n!$ to one. Consequently, having an effective algorithm to solve combinational problem made, we employ EDD method as a subprocedure of hybrid genetic algorithm.

3.2 A novel self-learning rule

In order to the better aggregation of GA and SA process and improve quality of solutions, an SLR which is updated after each generation is performed. In this criterion, a matrix is defined as denoting the best elements in the best solutions obtained so far. In other words, this criterion always remains the best information about suboptimal solutions and then notifies decision maker about solution space characteristics. The introduced matrix should be updated when a new population is completed. Thus, a parent is selected for further process if only it has a high similarity with the defined matrix among all parents. By this way, suitable elements in the best solutions found up to now are identified and continued in the next generation. The specific structure will be discussed in the rest of the content.

3.3 Hybrid method

In this section, we describe an efficient hybrid method based on genetic algorithm, simulated annealing, and EDD rule under an SLR criterion. Applying metaheuristics methods is one of the most common approaches which can enables researchers to solve NP-hard problems in large-sized cases and achieve suboptimal solutions. In this way, genetic algorithm and simulated annealing are the most popular approaches which have attracted researchers more than the others. Thus, if GA and SA are combined in a unique algorithm, a hybrid method with special advantages which none of the previous algorithms does not have in all of them will be simultaneously made. In order to combine them, we employ SA as a subordinate part of GA instead of mutation process. Thus, in GA process, to find the best neighborhood solutions for each selected parent, SA algorithm will produce a new child in population. Also, a novel SLR mentioned earlier will be applied to increase

efficiency of the aggregated method. Using such aggregation and utilizing rule, quality of solutions will be improved significantly and the entire feasible solution will be checked. Schema of this process is illustrated in Fig. 1, which will then be describe in detail in the rest of the paper.

3.3.1 Chromosome structure

Each solution represents a feasible solution which consists of a matrix divided in two parts. The first part of the matrix consists of assignment arrays. The length of this array is equal to: summation of the number of parts by the number of machines. The assign array represents the assignments of the parts and machines to cells. If part i is assigned to cell k, its entry in the assign array has value k in node i. In addition, if machine j is assigned to cell k, node (j + no. of parts) will get one. The second part located below the first part consists of integer elements. They represent the sequencing of each part in each scenario in cells where the part was assigned. Figure 2 illustrates a sample of the solution scheme.

3.3.2 Initial population

The first generation is created by initializing the population of chromosomes, $X^k = [x_i^k]$, k=1, 2, . . ., *pop_size* from the feasible region $\{(X)|g_i(x_i) \leq 0, x_i = 1, 2, \ldots, n\}$, randomly. Five steps will be performed to achieve each chromosome in initial population:

Step I: Assignment of machines

In this step, we assign machines to cells according to capacity of cells. To do this procedure, for each machine, a random number from one to number of cells is generated, and a cell will be selected by chance. If the selected cell has free capacity, then we assign the machine to it. Otherwise, this step is repeated until a feasible solution is found.

Step II: Assignment of parts

In this step, we assign parts to cells, randomly. To perform this, for each part, a random number from one to number of cells is generated, and a part is assigned to the selected cell.

Step III: Based on mentioned EDD rule, scheduling decisions will be determined. Thus, in each cell a part which has an earliest due date has priority and is processed earlier that leads to minimization maximum tardiness for parts in all cells.

Step IV: Total processing time for each part in each scenario in a cell assigned to it in previous step is computed.

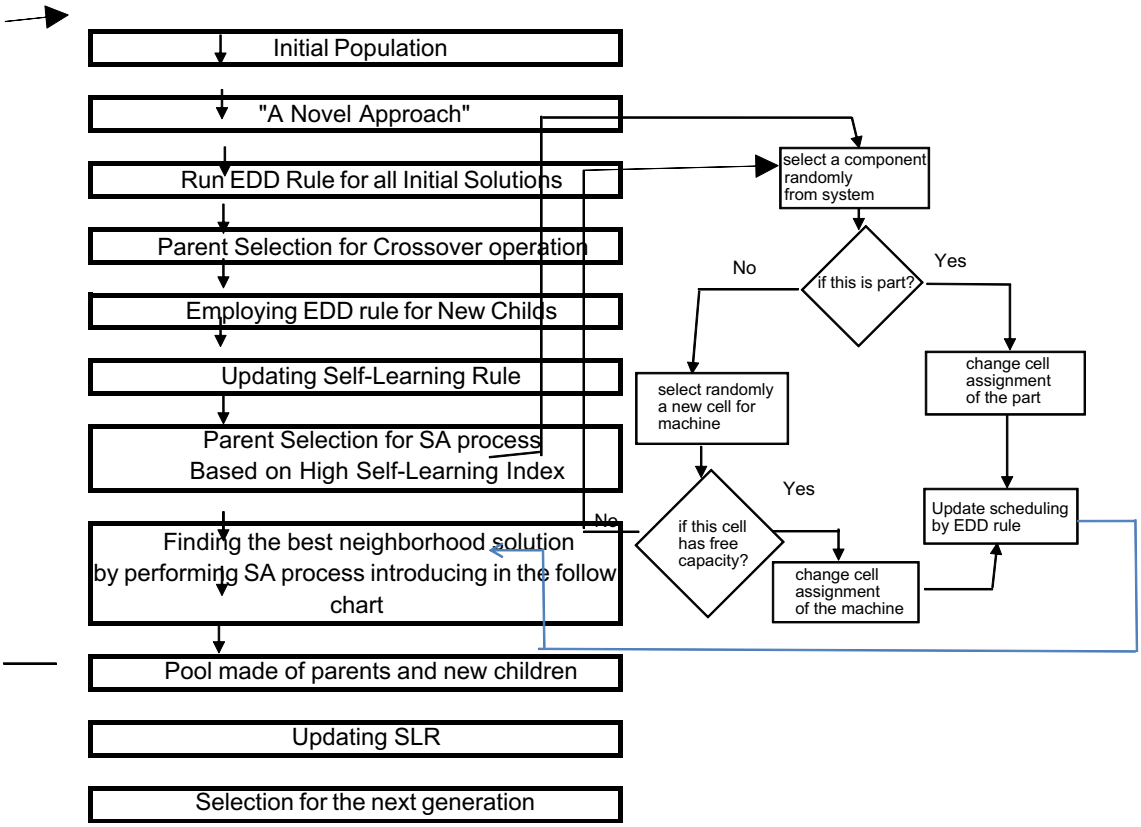

Fig. 1 Structure of hybrid solution procedure

Fig. 2 Sample of solution scheme

	Part 1	Part 2	.....	Part m	Machine 1	Machine 2	.....	Machine n
The First Part (Assignment of Elements)	3	2	.....	2	1	2	.....	3
The Second Part								
Sequencing in Scenario 1	1	2	.....	2	-	-	-	-
Sequencing in Scenario 2	2	1	.....	3	-	-	-	-
					-	-	-	-
Sequencing in Scenario s	1	2	.....	3	-	-	-	-

Step V: Based on the previous step, total processing time of a part with sequence [r] assigned to cell k in scenario s is determined. These details are the main variables and applied for computation the other variables based on the equations in model formulation. Details of steps IV and V are illustrated in Fig. 3.

3.3.3 Fitness function

The rank-based evaluation function is defined as the objective function for chromosome k=1,2,..., *pop_size*

3.3.4 Crossover operator

We combine the chromosomes V_k, k=1,2,..., *pop_size* by crossover operation. In order to determine the parents for crossover operation, the following process is repeated for k= 1 to *pop_size*: generating a random real number r from the interval (0, 1), the chromosome V_k will be selected as a parent provided that $r<Pc$; where the parameter Pc is the probability of crossover. Then selected parents $V_1', V_2', V_3', \ldots$ are grouped to the pairs $(V_1', V_2'), (V_3', V_4'), \ldots$. without loss of generality. For all parts and machines a random real number λ from the open interval (0, 1) is generated. Since each item may be assigned to different cells in each parent, thus in an offspring, each part or machine must be assigned to one of the cells assigned previously in its parent, randomly with a probability of 0.5 based on the value of λ. The crossover operator on V_2' and V_1' producing one child X is illustrated in Fig. 4:

```
Sub scheduling setting
 For i = 1 to no. parts
   For k = 1 to no. cells
     For s = 1 to no. scenarios
       For j = 1 to no. machines
          D_iks = D_iks + a_ij × t_ijs × x_ik × y_jk
       Next j
     Next s
   Next k
 Next i

For r = 1 to maximum no. of sequences
   For k = 1 to no. cells
     For s = 1 to no. scenarios
       For i = 1 to no. parts
          T_rks = T_rks + Z_isr × D_iks
       Next i
     Next s
   Next k
 Next r

End sub
```

Fig. 3 Pseudocode of computing scheduling variables

3.3.5 Defining self-learning rule

In order to better select parents for SA process, working instead of mutation operator and improving quality of offsprings, an SLR is employed. As it is described earlier, a matrix is defined denoting the number of times in which each part or machine is assigned to each cell in the best solutions up to now. In other words, after, the best solution is selected in each generation. In this solution, if part i is assigned to cell k, then in defined matrix array $[i,k] = [i,k] + 1$. Also, if machine j is assigned to cell k, then in defined matrix array $[\text{no. of parts} + j, k] = [\text{no. of parts} + j, k] + 1$. The defined matrix should be updated when a new population is completed. Thus, a parent is selected for SA process if only it has a high similarity with the defined matrix in assignment (or with high score) among all parents. In Fig. 5, sample of this matrix considering five parts, three machines, and three cells is illustrated. For example, array [2, 3] which is equal to four denotes that part three is

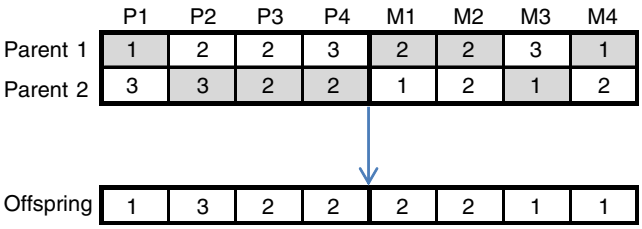

Fig. 4 Sample of crossover operator with four parts, four machines, and three cells

	cell 1	cell 2	cell 3
part 1	2	0	2
part 2	1	1	2
part 3	0	4	0
part 4	3	1	0
part 5	1	0	3
machine 1	2	0	2
machine 2	0	3	1
machine 3	1	2	1

Fig. 5 Sample of defined matrix performance in self-learning rule

assigned to cell two in the best solutions for four times so far. For this purpose, similarity and score for each solution is measured as the procedure defined in Fig. 6.

3.3.6 Simulated annealing instead of mutation operator

In this section, simulated annealing process will be performed to achieve the best neighborhood solution for each selected parent based on the self-learning index.

– Initialization of SA parameters

In this step, we describe procedure of setting the initial parameters of SA. Initial and final values for the control parameter temperature, named as T_i and T_f, respectively, are determined as follows.

- Initial temperature: due to significant influence of the objective function on the initial temperature, some illustrative examples are given. Also, its value should be large enough so that probability of acceptance of new solutions at the initial temperature T_0 reaches at least 80%. In this way, we generate 100 solutions randomly and compute their objective functions defined as g_i. We compute the difference between two sequential solutions as Δf_i. Therefore, we compute an initial temperature as that of Safaei et al. [40]:

$$\Delta = \Delta f_{\max} \rightarrow \qquad T_0 = \frac{\Delta}{-Ln(0.8)}$$

Pseudocode of initial temperature is shown in Fig. 7.

- Final temperature: this temperature is set to be:

$$T_f = 0.08 \times T_0$$

Also, the cooling rate (a) is considered to be a constant.

- Mechanism of generating feasible neighborhood solution:

To generate neighborhood solution for each selected parent, we modify current solution by one of the two following moves, randomly:

Move type 1: select a part randomly and change cell assignment of the selected part.

Move type 2: select a machine randomly and then find the number of the other cells having free

```
For each solution a score computed showing amount
of similarity of solution with SLR as follows:
For i = 1 to no. of parts
    For k = 1 to no. of cells
        Score = score + [i, k]×x_ik
    Next k
Next i

For j = 1 to no. of machines
    For k = 1 to no. of cells
Score = score + [no.of parts + j, k]×y_jk
    Next k
Next j
```

Fig. 6 Pseudocode of procedure computing score for each solution in self-learning rule

```
Sub initial temperature
  For i = 1 to 100
    Do
      Generate random solution
      Let g_i = fitness function of new solution
    End Do
  Next i
  For j =2 to 100
    Do
      Let Δf_j = |g_j − g_{j−1}|
    End Do
  Next j
  Let Δf_max = max {Δf_2 ,.., Δf_100 }
  Let Δ = Δf_max
Prob(Acceptance at T_0) ≥ 0.8 ⇒
e^{−Δ/T_0} ≥ 0.8 ⇒ −Δ/T_0 ≥ Ln(0.8) ⇒ T_0/Δ ≥ 1/(−Ln(0.8))
⇒ T_0 ≥ Δ/(−Ln(0.8))
End sub
```

Fig. 7 Pseudocode of the initial temperature

capacity so that the selected machine can be assigned to one of them. Finally, a cell from the candidate cells is found by chance and the machine is assigned to it.

- After performing move procedure, we update scheduling for new neighborhood solution by EDD rule.

Since the other parts of SA algorithm are described for many times in the literature, so to summarize this part of HGA, it is sufficient to present only the title of the other aspects of the proposed SA are similar to the classical in the literature [41]. So, we only list them as follows where they are applied in coding the algorithm:

- Evaluation of current solution and neighborhood solution
- Examination of acceptance condition
- Update counters
- Adjusting temperature ($T_{i+1} = T_i \times \alpha$)
- Stopping criteria

3.3.7 Selection process for the next generation in HGA

The selection process is based on selecting 50% from the best chromosomes and other 50% randomly. Thus we obtain *pop_size* copies of chromosomes, denoted by V_k.

3.4 Benchmark heuristic procedure

To measure efficiency of the proposed HGA, we compare HGA solutions with branch and bound solutions which obtained by Lingo 8 solver and also with a benchmark heuristic procedure that has introduced in the literature by [42]. This algorithm has two phases where in phase I part families based on maximization of similarity coefficients are determined and in phase II scheduling and the other tactical decisions based on previous phase are presented. For the proposed model, we apply this heuristic procedure using same structure and solve the model in two steps. In the first step, assignment of machines to cells and part families are determined. In this way, we use a similarity coefficient ($S_{j,l}$) based on part-machine matrix presented in the literature by Jaccard [43] for any pair of machines. Therefore, according to the maximizing of similarity of machines assigned to the same cell, we determine cells of machines. For the pair of machines, similarity is defined as the number of parts that operated by both machines divided by number of parts that operated by at least one of them:

$$S_{j,l} = \frac{\sum_i a_{ij}a_{il}}{\sum_i \left(a_{ij} + a_{il} - a_{ij}a_{il}\right)} \tag{25}$$

$$a_{ij} = \begin{cases} 1 & \text{If part } i \text{ require to be processed on machine } j. \\ 0 & \text{Otherwise.} \end{cases} \tag{26}$$

Using the preceding definition, the model which determined the cell of machines is as:

$$\text{Max } w = \sum_k \sum_j \sum_{,l \neq j} S_{jl} y_{jk} y_{lk} \tag{27}$$

S.t:

$$\sum_k y_{jk} = 1 \qquad \forall j \tag{29}$$

$$\sum_i y_{jk} \leq M_{\max} \qquad \forall j \tag{30}$$

$$y_{jk} \sim (0,1) \qquad \forall j,k \tag{31}$$

The objective function maximizes summation of similarities in cells. In this way, similarity between machines i and j is computed when both machines assigned are to the same cell and will be 0 otherwise. Using defined objective, cells of machines will be determined with maximum similarity. The first constraint ensures that each machine is assigned to one cell and the second constraint guarantees the number of machines assigned to the same cell will not exceed the determined upper bound.

By solving this model, machine cells are determined and it enables us to find assignment of parts to the part families. For this purpose, number of operations for each part in each cell is computed and part is assigned to a cell which has maximum number of operations. To do this, we define an equation which computes the mentioned criteria.

$$\text{NO}_{i,k} = \sum_j a_{ij} y_{jk} \tag{32}$$

$\text{NO}_{i,k}$ Number of operation of part i in cell k.
y_{jk} A 0 to one variable that gives a value of one if machine j assigned to cell k and 0 otherwise. (This variable is computed in the previous section).

After this, we set part assignment variables as shown in Fig. 8:

In the second step of the algorithm, we make group scheduling decisions for each cell like [42]. This procedure performs a similar manner like EDD procedure executed in initialization phase and steps III to IV in section 3.3.2 which used to determine GS decisions.

For $i = 1$ to parts
$c = \{c \mid NO_{ic} \quad is \quad \max(NO_{ik})\}$
$x_{ic} = 1$
Next i

Fig. 8 Pseudocode of assignment process

4 Numerical experiments

In order to verify the performance of the HGA approach, we have solved 21 random instances. We have considered a time criterion to determine the number of solved instances. For medium-sized problems, we have started from a small-sized problem and increased gently the size of problem by a specific rule until the exact approach could not reach the optimum solution within a predetermined run time. In a similar fashion, for large-sized problems, we have started from the largest medium-sized problem and increased the size of the problem by a specific rule until the branch and bound algorithm could not reach the feasible solution within a run time. These problems are generated randomly based on consideration of similar data in the literature. All algorithms considered in this paper are coded in Visual Basic 6 and run on a Pentium IV PC with 3 GHz CPU and 512 MB RAM. The associated results are compared with global solutions obtained by the Lingo 8 software which uses branch and bound algorithm to solve such problem.

For HGA parameter settings, some scenarios are applied as follow:

- Pc: four scenarios: 0.75, 0.80, 0.85, and 0.95.
- Pm is based on the self-learning rule index.
- Population size: two scenarios: 20 and 30.
- No. of generation: two scenarios: 250 and 300.

Also, the SA parameter setting is shown in Table 2. Parameter K is number of iteration in each temperature to achieve equilibrium. Also, parameter N is maximum number of consecutive temperature trails when it reaches to the predetermined value the algorithm is stopped. Parameter K is set to 150 empirically.

In the first section of computational examples, we generate small-sized problems and solve them with three methods: branch and bound algorithm to obtain global solutions, proposed HGA and benchmark heuristic procedure, and then compare HGA solutions with the other methods. In the second part, we present some medium-sized problems and since optimal solver cannot achieve optimal solutions in a reasonable run time, to measure efficiency of the proposed HGA for medium-sized problems, we set time constraint for Lingo solver and compare HGA solutions with the best solutions found by Lingo in a limited time. In this way, each example is allowed to be solved within 5,400 s (1.5 h). However, because of the time-consuming computation, the introduced model cannot be optimally solved within this time. Thus, to solve the medium-sized problems, we consider a possible interval for optimum value of objective function (F^*) that constructed by the F^{bound} and F^{Best} values that are proposed by Lingo software where $F^{bound} \leq F^* \leq F^{Best}$. Based on the Lingo software's documents, the F^{Best} determines the best feasible solution found so far. Also, F^{bound} determines the bound on the objective function. This bound is a limit on how far the solver will be able to improve the objective. At some point, these two values may become very close. Given that the best objective value can never exceed the bound, the fact that these two values are close determines that Lingo's current best solution is either the optimal solution, or very close to it. At such a point, the user can interrupt the solver and accept with the current best solution in order to save additional computation time. As we said before, we interrupt the solver within 5,400 s (this procedure is like that of Ching-Ter and Chi-Chiao [40]).

In the third part, we generate some large-sized problems, and due to the inability of Lingo software to find a feasible solution within 5,400 s, we just use the proposed benchmark heuristic procedure to obtain the best solutions and thus efficiency of the proposed HGA in large-sized problems can be measured by comparing HGA solutions with benchmark heuristic solutions in the literature.

Above discussions are summarized as follows:

- A problem which can be solved optimality by B&B within 5,400 s is small.
- A problem which can be only feasible by B&B within 5,400 s is medium.
- A problem which cannot be feasible by B&B within 5,400 s is large.
- In small-sized problems, HGA is compared against Global solutions and heuristic solutions.
- In medium problems, HGA is compared against the best solutions by B&B algorithm and heuristic solutions.
- In large problems, HGA is compared against only heuristic solution.

We start to solve small-sized problems and increase size of problems based on specific rule until we reach medium-sized problems. As mentioned earlier, in medium-sized problems, Lingo software cannot achieve global solutions

Table 2 Information for SA parameter setting

K	α	T0	Tf
150	0.9	$T_0 = \frac{\max\{\Delta f_i\}}{-Ln(0.8)}$	0.08×T0

in a predetermined run time. Finally, for large-sized problems in which Lingo cannot find feasible solution in a predetermined time, validation of HGA algorithm is based on only benchmark algorithm. These problems are generated randomly based on consideration of similar data in literature. In each class, processing times in scenario 1 have been randomly generated using a uniform distribution in the rage of U (2, 10). Also, processing time in scenario 2 are generated randomly distributed in the range of {processing time in scenario $1 + (-2) + U[0, 4]$}. Number of scenarios for describing uncertainty is considered two. The number of operation for each part is random integer in the range of U (2, 6). Due date for parts are also generated from interval U (50, 75), randomly. According to the mentioned intervals, average of total processing time for each part determined as: average operation time × average number of operation. Thus, for each part, average processing time will be equal to $6 \times 4 = 24$. Consequently, a due date of distribution of U (50, 75) implies that the ratio of mean due date to mean job processing time is approximately 3:1 (62.5:24); and therefore average number of parts with no tardiness will be three.

4.1 Effectiveness of HGA in the small-sized problems

In this section, we present seven numerical examples with small size to compare the proposed HGA with global solutions and benchmark heuristic solutions. Moreover, each numerical example is solved using a branch and bound algorithm with the Lingo 8.0 software. Table 3 illustrates the characteristics of numerical examples and summarizes the results of the proposed algorithms.

It appears that the entire objective functions obtained by HGA algorithm, benchmark heuristic procedure and global optimums don not have any difference in small-sized problems. We present a parameter, called the percent error (i.e., (global optimum—HGA objective function)/global optimum, where the global objective value is obtained by branch and bound. The last column, named "error", in Table 3 shows the above errors. In Table 3, it is concluded that there is no percentage error when different small-sized problems are selected. It implies that the proposed hybrid genetic algorithm and heuristic procedures are effective to solve the presented model in this class of problems. Table 3 can show us another result. As mentioned earlier, a problem which cannot be solved optimally in 5,400 s is in class of medium-sized problems. So, problems with greater scale rather than example M1 in Table 3 are to be in the class of medium- and large-sized problems.

4.2 Effectiveness of the proposed HGA in the medium-sized problems

In this section, to illustrate effectiveness of the proposed HGA, we generate some instances with medium size and solve them by both B&B algorithm and benchmark heuristic procedure. As we described earlier, for these problems Lingo solver which uses branch-and-bound method to solve the model cannot get us global solutions in maximum run time (5,400 s). Therefore, in this section, solutions obtained by HGA are compared with the best objective function of branch-and-bound method within limited time to measure effectiveness of the introduced HGA algorithm in medium-sized problems. Also, we solve these problems by benchmark heuristic procedure to learn more about behavior of this algorithm in solving problems. Table 4 presents results for these problems.

In the examples in Table 4, performance evaluation is based on the best objective function found in 5,400 s using B&B algorithm. This table indicates that the comparison

Table 3 Effectiveness of hybrid GA in small-sized problems

Prob. No.	Problem info.					B&B solution	HGA solution	Heuristic solution	$T_{B\&B}$ (seconds)	T_{HGA}	Error percent
	No. of parts	No. of machines	No. of scenarios	No. cells	Max machine allowed in each cell						
S1		4×3×1×3			2	28	28	28	1	<1	0.00%
S2		4×4×1×3			2	33	33	33	1	<1	0.00%
S3		4×3×2×3			2	27	27	27	17	<3	0.00%
S4		5×4×1×3			3	39	39	39	58	<3	0.00%
S5		4×4×2×3			3	40.5	40.5	40.5	37	<3	0.00%
S6		5×4×2×3			3	66	66	66	812	<3	0.00%
S7		7×5×1×3			3	115	115	115	3,015	<3	0.00%
M1		8×6×1×3			3	122 (Best Solution)			>5,400		

Table 4 Effectiveness of hybrid GA in medium-sized problems

Problem No.	Problem information					B&B solution	HGA solution	Heuristic solution	$T_{B\&B}$ (seconds)	T_{HGA}	Error percent
	No. of parts	No. of machines	No. of scenarios	No. of cells	Max machine allowed in each cell						
M1		8×6×1×3			3	122	101	101	>5,400	4	17.15%
M2		9×6×1×3			3	309	238	238	>5,400	9	22.86%
M3		9×7×1×3			4	345	275	292	>5,400	13	20.21%
M4		10×8×2×3			4	385	280	311	>5,400	17	27.35%

between the results obtained by B&B, benchmark algorithm and HGA, and characteristics of the examples. The last column shows the percentage of gap between B&B and HGA solutions which is computed with the ratio $(F_{Best}(B\&B) - HGA(solution))/HGA(solution) \times 100$. As shown in the last row, the average value of gap is 21.93% which implicates to a better performance of HGA rather than the best solution of B&B algorithm in limited time. Indeed, from Table 4 when the scale of the problems is increased, solutions obtained by benchmark heuristic procedure is to be laid between the solutions of B&B and HGA solutions $\left(Z^{Best}_{B\&B} \geq Z^{Best}_{Heuristic} \geq Z^{Best}_{HGA}\right)$ and therefore, HGA has a better performance rather than benchmark heuristic algorithm in medium-sized problems, too. These comparisons show efficiency of the presented HGA in medium-sized problems. To clarify the problem attempted in this paper, we consider a typical CMS with three cells. In this system, ten parts are to be scheduled on eight machines. The processing time of each part on each machine and also final solution for problem M4 in Table 4 achieved by HGA approach are given in Table 5.

4.3 Efficiency of HGA in large-sized problems

In this section, we present six numerical examples with large size to compare proposed HGA with the benchmark heuristic solutions. Since Lingo solver cannot obtain feasible solutions for these problems within maximum run time (5,400 s), we obtain the best solutions only by benchmark method and it will be the basis to measure performance of the presented HGA. Table 6 shows results of this class of problems.

To evaluate performance and efficiency of HGA in large-sized problems, performance measurement is based on solutions obtained from benchmark algorithm. Table 5 summarizes this comparison and the other characteristics of solved examples. In this section, we define a measurement and named it "improvement percent" computed by $(heuristicOFV - -HGAOFV)/HGAOFV \times 100$. The solution results presented here and also the last row show that average value of improvement percent which is 6.19% implicates to a better performance of HGA rather than heuristic procedure in large-sized problems. In other words, HGA

Table 5 Processing time, cells and part families for problem M4

Machines		Parts																			
		5		6		9		10		1		2		4		3		7		8	
		I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]	I[a]	II[b]
Cell 1	A	9	9	3	4	5	4	3	4												
	D	3	1	7	8			4	2												
	E	3	2	9	9	8	7			3	2										
Cell 2	C	4	5							5	3	4	4	3	1			7	5		
	G									3	3	7	8	5	3						
Cell 3	B															9	7	3	2	4	2
	F	9	9													5	4	9	10	7	5
	H															3	2	4	3		

[a] Processing time in scenario 1

[b] Processing time in scenario 2

Table 6 Effectiveness of hybrid GA in large-sized problems

Problem No.	Problem information					Heuristic solution	HGA solution	Improvement percent
	No. of parts	No. of machines	No. of scenarios	No. cells	Max machine allowed in each cell			
L1		15×10×2×3			5	976	945	3.15%
L2		20×13×1×3			6	2,108	1,972	6.45%
L3		30×19×2×4			6	4,836	4,601	4.85%
L4		25×16×2×4			6	6,114	5,769	5.65%
L5		35×22×1×4			6	16,180	14,701	9.14%
L6		40×25×2×5			6	25,239	23,790	5.74%
L7		45×25×1×5			6	26,501	24,829	6.31%
L8		40×28×2×5			7	29,151	27,143	6.89%
L9		50×28×2×6			7	34,981	32,456	7.22%
L10		50×30×2×6			7	41,978	39,379	6.19%
							Average	6.16%

solutions are better about 6.19% than heuristic solutions which imply HGA is so effective to solve large-sized problems.

4.4 Robustness of the proposed HGA

In Table 7, we compare solutions obtained by HGA approach for problem with 30 parts, 19 machines, two scenarios, and four cells when different parameters in HGA approach are taken with the same generations as a stopping rule. It appears that all the minimal costs differ little from each other. In order to account for it, we present a parameter, called the percent error, i.e., (objective value—the best objective value)/the best objective value, where the best objective value is the least one of all the ten minimal costs obtained above. The last column named by "error" in Table 7 is just this parameter. From Table 7, the percent error does not exceed 2.10% when different parameters for HGA algorithm are selected, which implies that the hybrid genetic algorithm is robust to the initial parameter settings and effective to solve the model.

Therefore, the variation of initial parameters in the algorithm has slight impact on the optimal objective, which implies that the algorithm designed in this paper is much robust.

5 Results and discussion

In this paper, we introduced a notation of SCFP considering stochastic processing times where described by discrete scenarios. A conceptual framework and a mathematical model were proposed. Then mathematical method applied to linear the proposed model. Also, a hybrid genetic algorithm was introduced to solve the model. We divided our computational experiments to four parts. In the first part, both hybrid genetic algorithm and a heuristic

Table 7 Robustness of the proposed approach

	GA parameters			SA parameters		Total cost	Error
	Pop_size	Pc	Generation	*a*	K		
1	20	0.95	250	0.90	150	4,652	1.10%
2	20	0.80	300	0.95	200	4,642	0.90%
3	20	0.75	250	0.85	150	4,661	1.30%
4	20	0.90	300	0.90	200	4,654	1.15%
5	20	0.75	250	0.95	150	4,665	1.40%
6	30	0.85	300	0.85	200	4,679	1.70%
7	30	0.80	250	0.95	150	4,680	1.72%
8	30	0.95	300	0.85	200	4,698	2.10%
9	30	0.90	250	0.90	150	4,601	0.00%
10	30	0.85	300	0.85	200	4,692	1.97%

procedure in the literature were able to find optimal solutions. In the second part, we compared HGA and heuristic solutions with the best solutions obtained by Lingo using branch and bound algorithm. Numerical examples showed that HGA algorithm had a better performance than the other approaches. Also, heuristic solutions were to lie between branch and bound and HGA solutions. In the third part, Lingo solver cannot get us feasible solution and therefore we compared HGA algorithm with proposed heuristic procedure. Computational experiments showed that in large-sized problems HGA works better than heuristic algorithm. In the last part, robustness of the algorithm was validated which shows informs that HGA algorithm was not sensitive to the initial settings. Our contributions research field consists of: considering stochastic parameters which yield to more flexibility and practical aspects in real world cases, integrating cell formation problem with scheduling aspects, linearization of the model and presenting a hybrid genetic algorithm which had successful performance in any size of problem.

For future research, we suggest three directions:

- Development of the model under more and the other stochastic parameters such as costs, processing routes and machine availability.
- Considering this problem as a multiobjective model which considers CF decisions in one objective and scheduling in the other objective.
- Aggregating proposed model with the other production aspects like layout problem considerations. These remain a critical issue for future study.

References

1. Heragu S (1997) Facilities design. PWS publishing company, Boston, p 316
2. Papaioannou G, Wilson JM (2008) Fuzzy extensions to integer programming of cell formation problem in machine scheduling. Ann Oper Res 166:1–19
3. Hentschel C, Seliger G, Zussman E (1995) Grouping of used products for cellular recycling systems. CIRP Ann Manuf Technol 44 (1):11–14
4. Shanker R, Vrat P (1999) Some design issues in C.M. using the fuzzy programming approach. Int J Prod Res 37(11):2545–2563
5. Szwarc D, Rajamani D, Bector CR (1997) Cell formation considering fuzzy demand and machine capacity. Int J Adv Manuf Technol 13(2):134–147
6. Ravichandran KS, Chandra Sekhara Rao K (2001) A new approach to fuzzy part family formation in CMS. Int J Adv Manuf Technol 18(8):591–597
7. Song S, Hitomi K (1991) Determining the planning horizon and group part family for flexible cellular manufacturing. Nippon Kikai Gakkai Ronbunshu, C Hen/Trans Jpn Soc Mech Eng C 57 (542):3364–3371
8. Hurley SF, Clay Whybark D (1999) Inventory and capacity trade-off in a manufacturing cell. Int J Prod Econ 59(1):203–212
9. Tavakkoli-Moghaddam R, Javadian N, Javadi B, Safaei N (2007) Design of a facility layout problem in CMS with stochastic demand. Appl Math Comput 184(2):721–728
10. Balakrishnan J, Cheng CH (2005) Dynamic C.M. under multi-period planning horizon. J Manuf Technol Manage 16(5):516–530
11. Balakrishnan J, Cheng CH (2007) Multi-period planning and uncertainty issues in C.M.: a review and future directions. Eur J Oper Res 177(1):281–309
12. Yang J, Deane RH (1993) Setup time reduction and competitive advantage in a closed manufacturing cell. Eur J Oper Res 69 (3):413–423
13. Kuroda M, Tomita T (2005) Robust design of a cellular—line production system with unreliable facilities. Comput Ind Eng 48 (3):537–551
14. Hosseini MM (2000) An inspection model with minimal and major maintenance for a system with deterioration and poison failures. IEEE Trans Reliab 49(1):88–98
15. Gupta SM, Kavusturucu A (1998) Modeling of finite buffer cellular manufacturing systems with unreliable machines. Int J Ind Eng Theory Appl Pract 5(4):265–277
16. Simeu-Abazi Z, Sassine C (1999) Maintenance integration in manufacturing systems using stochastic Petri nets. Int J Prod Res 37(17):3927–3940
17. Asgharpour MJ, Javadian N (2004) Solving a stochastic cellular manufacturing model using genetic algorithm. Int J Eng Trans A: Basics 17(2):145–156
18. Sun Y-L, Yih Y (1996) An intelligent controller for manufacturing cell. Int J Prod Res 34(8):2353–2373
19. Andres C, Lozano S, Adenso-Diaz B (2007) Disassembly sequence planning in a disassembly cell. Robot Comput Integr Manuf 23(6):690–695
20. Taylor JF, Ham I (1981) The use of a micro computer for grouping scheduling. In: Proceedings of the 9th North American Manufacturing Research Conference (NAMRC), Society of Manufacturing Engineers, pp 483–491
21. Logendran R, Nudtasomboon N (1991) Minimizing the makespan of a group scheduling problem: a new heuristic. Int J Prod Econ 22:217–230
22. Solimanpur M, Vrat P, Shankar R (2004) A heuristic to minimize makespan of cell scheduling problem. Int J Prod Econ 88:231–241
23. Aneja YP, Kamoun H (1999) Scheduling of parts and robot activities in a two machine robotic cell. Comput Oper Res 26:297–312
24. Lockwood WT, Mahmoodi F, Ruben RA, Mosier CT (2000) Scheduling unbalanced cellular manufacturing systems with lot splitting. Int J Prod Res 38(4):951–965
25. Tsai CC, Chu CH, Barta T (1997) Analysis and modeling of a manufacturing cell formation problem with fuzzy integer programming. IIE Trans 29(7):533–547
26. Vakharia AJ, Kaku BK (1993) Redesigning a cellular manufacturing system to handle long-term demand changes: a methodology and investigation. Decis Sci 24(5):909–917
27. Mahdavi I, Javadi B, Fallah-Alipour K, Slomp J (2007) Designing a new mathematical model for cellular manufacturing system based on cell utilization. Appl Math Comput 190:662–670
28. Wu XD, Chu CH, Wang YF, Yan WL (2006) Concurrent design of cellular manufacturing systems: a genetic algorithm approach. Int J Prod Res 44(6):1217–1241
29. Venkataramanaiah S (2007) Scheduling in cellular manufacturing systems: an heuristic approach. Int J Prod Res 99999(1):1–21
30. Sridhar J, Rajendran C (1993) Scheduling in a cellular manufacturing system: a simulated annealing approach. Int J Prod Res 31 (12):2927–2945
31. Mahmoodi F, Martin GE (1997) A new shop-based and predictive scheduling heuristic for cellular manufacturing. Int J Prod Res 35 (2):313–326

32. Saad SM (2003) The reconfiguration issues in manufacturing systems. J Mater Process Technol 138:277–283
33. Panchalavarapu PR, Chankong V (2005) Design of cellular manufacturing systems with assembly considerations. Comput Ind Eng 48(3):449–469
34. Ghezavati VR, Jabal-Ameli MS, Makui A (2009) A new heuristic method for distribution networks considering service level constraint and coverage radius. Expert Syst Appl 36(3):5620–5629 Part 1
35. Mobasheri F, Orren LH, Sioshansi FP (1989) Scenario planning at southern California Edison. Interfaces 19(5):31–44
36. Shanker R, Vrat P (1998) Post design modeling for CMS with cost uncertainty. Int J Prod Econ 55(1):97–109
37. Snyder LV (2006) Facility location under uncertainty: a review. IIE Trans 38:537–554
38. Glover F, Woolsey L (1974) Converting the 0–1 polynomial programming problem to a 0–1 linear program. Oper Res 22:180–182
39. Chang C-T, Chang C-C (2000) A linearization method for mixed 0–1 polynomial programs. Comput Oper Res 27:1005–1016
40. Safaei N, Saidi-Mehrabad M, Jabal-Ameli MS (2008) A hybrid simulated annealing for solving an extended model of dynamic cellular manufacturing system. Eur J Oper Res 185:563–592
41. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
42. Jeon G, Leep HR (2006) Forming part families by using genetic algorithm and designing machine cells under demand changes. Comput Oper Res 33:263–283
43. Heragu S (1997) Facilities design. PWS publishing company, Boston, p 303