RAIRO-Oper. Res. 55 (2021) S167–S194
https://doi.org/10.1051/ro/2019075

**RAIRO Operations Research**
www.rairo-ro.org

# NESTED BI-LEVEL METAHEURISTIC ALGORITHMS FOR CELLULAR MANUFACTURING SYSTEMS CONSIDERING WORKERS' INTEREST

Bardia Behnia[1], Babak Shirazi[1,*], Iraj Mahdavi[1]
and Mohammad Mahdi Paydar[2]

**Abstract.** Due to the competitive nature of the market and the various products production requirements with short life cycles, cellular manufacturing systems have found a special role in manufacturing environments. Creativity and innovation in products are the results of the mental effort of the workforces in addition to machinery and parts allocation. Assignment of the workforce to cells based on the interest and ability indices is a tactical decision while the cell formation is a strategic decision. To make the correct decision, these two problems should be solved separately while considering their impacts on each other classically. For this reason, a novel bi-level model is designed to make decentralized decisions. Because of the importance of minimizing voids and exceptional element in the cellular manufacturing system, it is considered as a leader at the first level and the assignment of human resources is considered as a follower at the second level. To achieve product innovation and synergy among staff in the objective function at the second level, increasing the worker's interest in order to cooperate with each other is considered too. Given the NP-Hard nature of cell formation and bi-level programming, nested bi-level genetic algorithm and particle swarm optimization are developed to solve the mathematical model. Various test problems have been solved by applying these two methods and validated results have been shown the efficiency of the proposed model. Also, real experimental comparisons have been presented. These results in contrast with previous works have been shown the minimum amount of computational time, cell load variation, total intercellular movements, and total intracellular movements of this new method. These effects have an important role in order to the improvement of cellular manufacturing behavior.

**Mathematics Subject Classification.** 90Bxx.

Received February 19, 2018. Accepted August 14, 2019.

## 1. Introduction

In the contemporary business environment, values change and the ability to adapt is very valuable. Nowadays, in companies quests for efficiency, comprehensive quality, client satisfaction, and improved quality of working life, manufacturers experience a variety of new forms. To improve their performance, manufacturers seek advantages such as smaller size, more flexible operations, new and creative products. In the past, manufacturers addressed

*Keywords.* Cellular manufacturing, bi-level programming, NBL-GA, NBL-PSO, workers' interest.

[1] Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran.
[2] Department of Industrial Engineering, Babol Noshirvani University of Technology, Babol, Iran.
*Corresponding author: shirazi_b@icloud.com

Article published by EDP Sciences

customer preferences by their large-scale production. This trend continued, perhaps due to the low diversity of products. Today, however, due to the high level of customer expectations caused by increased consumers' knowledge and awareness, customers will determine the product characteristics. For this reason and according to the globalization trend and increased emphasis on customer requirements, significant changes have occurred in the business environment and many manufacturers focus on customer-oriented production. Companies and manufacturers seek to increase customer satisfaction, the number of loyal customers and competitive advantages in the market with well-designed products based on customer desires. With the growth in the economy and expansion of manufacturers, the companies tasks and the markets in which they operate are becoming increasingly more complicated. Thus, the employees' activities in these organizations must be specialized more than ever, and experts in a company need to learn how to arrange team works. Another important issue is the intense economic competition. In today's economy, leaders in most industries often utilize large savings and huge growth. In such circumstances, attaining the best employees' performance has been very important, and employees are expected to be more skilled in their respective fields. Therefore, a higher number of professional constraints depend on the employees' expertise and interest [1]. Nowadays, many manufacturing systems are based on group technology. Parsing a manufacturing system to subsystems for lighter controlling and planning is the main idea of group technology. The cellular manufacturing system is one of the most famous applications of group technology [26]. The benefits of both job shop and flow shop appeared in a cellular manufacturing system. In cellular manufacturing systems, the part family is a group of parts with the same processing requirements and processed by various machines in a cell. Betterment of the production control, productivity, and decreased setup times, material handling charges are awaited in a cellular manufacturing system [17, 19]. In designing a cellular manufacturing system one of the first and important problems is the cell formation which known as Part-Machine grouping problem. Minimizing parts movement between cells and in each cell, which known as intercell and intracell movements are the main objectives of the cell formation problem. All equipments and skills which required for a product should be located on a cell based on the cellular manufacturing policies. By decreasing intercell and intracell movements, production costs cut. Moreover, the role of the worker in manufacturing systems cannot be refused. Nowadays both technical and human characteristics are considering for continuous success and performance of cellular manufacturing system. Figure 1 has been shown the schema of a cellular manufacturing system considering workers' interest.

Many of the earliest researches on cellular manufacturing systems focused just on technical problems such as part family grouping, cell balancing, and processing routes. Goncalves and Rezende [15] provided a method with compounding a genetic algorithm and novel local algorithm to determine product family and machine cell [15]. Albadawi *et al.* [2] provided a mathematical model with two phases for solving manufacturing cell formation problem. Machine cells and part allocation to the cells respectively have been done in the first and second phase [2]. Mahdavi *et al.* [24] proposed a mathematical model for the cell formation problem based on the concept of using cells in cellular manufacturing systems. This model aimed to minimize the number of voids and exceptional elements in the cells. They also planned an effective method for solving the mathematical model based on the genetic algorithm [24]. Anvari *et al.* [3] developed a particle swarm optimization algorithm (PSO) for minimizing voids and exceptional elements with determining machine group and the coefficient of commensurate similarity [3]. Paydar *et al.* [25] proposed a mathematical model for specifying part families and machine groups at the same time in the cell formation problem. Minimizing intracellular empty spaces and exceptional elements at the same time are the main objectives of their model. The optimal number of cells is determined by the model, and the number of manufacturing cells consideration as a decision variable is the most important advantage of this model [25]. Elbenani *et al.* [13] proposed a local search method for solving the cell formation problem in which each cell contains at least one machine and one part. Their proposed method successively uses resonance strategy to optimize the answer locally, and destruction strategy for gaining a new resolution from the previous answer [13]. Brown [8] tries to dilute the total cost in cellular manufacturing systems through the number of similar machinery and minimizing intracellular movements. He developed mathematical models by working on long-term bearable cell formation [8]. Moreover, some researches focused on human factors in the cellular manufacturing problem. Fazakerley [14] studied group technology problems

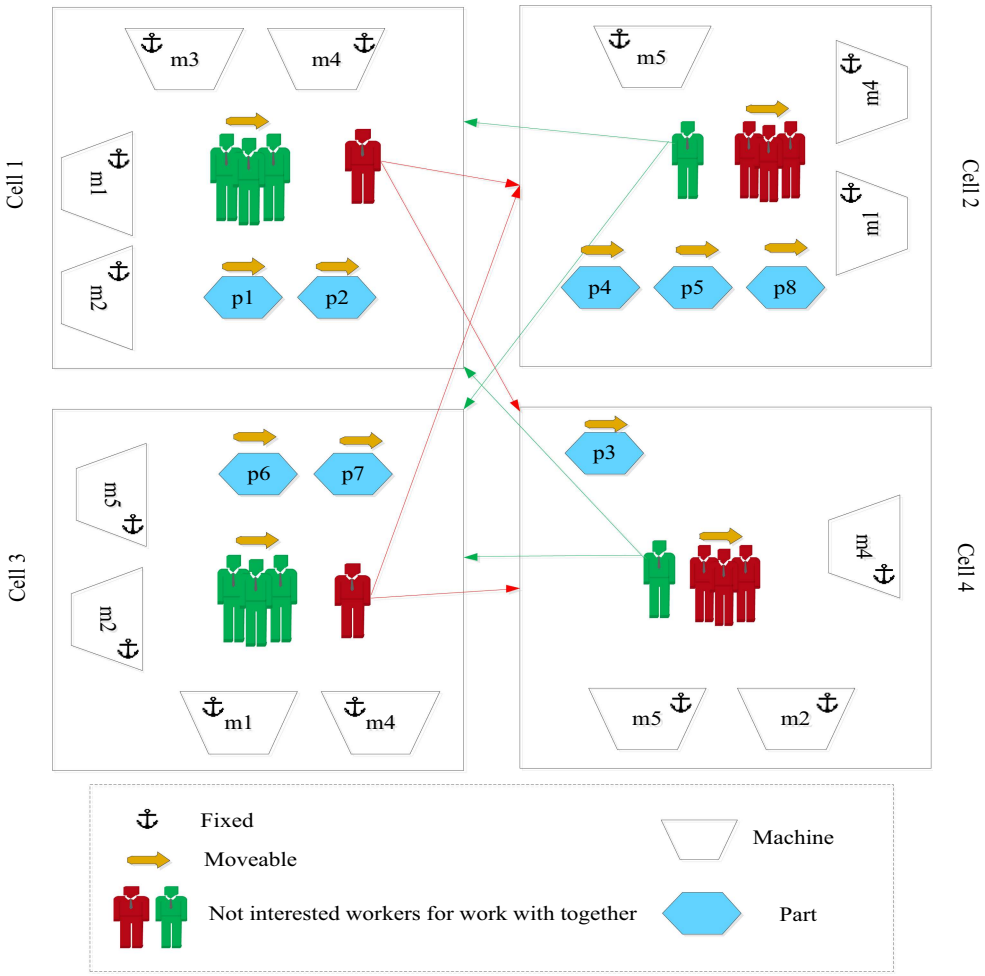


FIGURE 1. The schema for the cellular manufacturing system considering workers' interest [6].

for human resources like loss of group feeling, vulnerability, and hesitancy [14]. Shafer *et al.* [30] studied about cellular manufacturing system and its negative effect on worker's manners and conceptions. They believed that organizational commitment and job satisfaction cause derivation in cellular manufacturing systems [30]. Udo and Ebiefung [33] studied about human factors and they're relevant to the success of cellular manufacturing system. According to their study, top management, training, and self-interest have an affirmative effect on manufacturing and group technology efficiency and improvement [33]. Egilmez *et al.* [12] studied worker allocating to various cells. They proposed a mathematical model based on the abilities of the workers for allocating the worker to different cells with a random approach [12]. The nature of cellular manufacturing systems in manufacturing products in mid-variety and mid-volume, the product mix variation is not too far-fetched. Bootaki *et al.* [7] studied about configuring manufacturing cells when product mix variation occurs. The part-machine incidence matrix should be changed because of product mix variation. They formulated the problem with two different criteria which considered to workers' relation together and each worker experts [7]. Sakhaii *et al.* [27] considered decreasing the machinery moving costs, lack costs and worker cost. They resolved the problem in a cellular manufacturing system with untrustworthy machinery using mixed integer programming model [27]. Azadeh *et al.* [4] proposed a mathematical model for the dynamic cellular manufacturing system. They repre-

TABLE 1. Literature review of the cellular manufacturing system and workforces' assignment.

| Author(s) | Movement cost | | Workforces assignment | | | | CSL | Decision making method | | NML | Solution method |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Intra-cell | Inter-cell | Skill | WLB | Interest | Stochastic | | CN | DCN | | |
| Resende and Goncalves [15] | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | GA |
| Albadawi *et al.* [2] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | IP |
| Mahdavi *et al.* [24] | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | GA |
| Anvari *et al.* [3] | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | PSO |
| Paydar el al. [25] | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | Lingo |
| Elbenani *et al.* [13] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | Hybrid GA and LSA |
| Brown [8] | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | Lingo |
| Egilmez *et al.* [12] | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | Simulated annealing |
| Bootaki *et al.* [7] | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | NSGA-II |
| Sakhaii *et al.* [27] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | MILP |
| Azadeh *et al.* [4] | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | NSGA-II and MOPSO |
| This paper | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | NBL-GA and NBL-PSO |

**Notes.** CN: Centralized decision making, DCN: Decentralized decision making, WLB: Workload balancing, CSL: Cell size limitation, NML: Number of Machine Limitation, IP: Integer programming, MILP: Mixed-integer linear program.

sented a multi-objective model by taking human factors which three objectives are considered in their model as minimizing the total cost of the dynamic cellular manufacturing system, minimizing operators' decision-making inconsistency style and workload balancing of operators' in cells. They solved nine test problems for verifying and validating their proposed model [4]. A summary of past researches and the contribution of this work is shown in Table 1. The purpose of cellular manufacturing problems is the best layout for the facilities. Planning facilities can be centralized or decentralized. With various sizes of facilities and in centralized planning, manufacturing policies are designed in an integrated paradigm; but in decentralized cases, this activity could change depending on the requirements and policies taken for different facilities even in one unit. Organizations should adapt themselves to the changes of the environment regards the current dynamic and constantly changing business environment and seek innovation and competitive advantages for survival. It is necessary to raise the sense of collaboration and coordination between the workers to sustain an advanced and active formation. Workers' optimize the group performance collaboration results in a more efficient and realistic exchange of knowledge and consecutively. In this case, worker planning in cellular manufacturing should be focused on promoting cooperation interactions between the staff. Thus, the optimal design of cells based on workers interests has important role in improvement of cellular manufacturing behavior. The remainder of this paper is organized as follows. Section 2 presents problem formulation; Section 3 develops a nested bi-level genetic algorithm (NBL-GA); then in Section 4, nested bi-level particle swarm optimization algorithm (NBL-PSO) is provided in detail. Individual solution representation has been considered in Section 5. Sections 6 and 7 discuss Taguchi's experimental design and computational results. Finally, conclusions and future research direction are listed in the last section.

## 2. Problem formulation

In this section, for formulating the problem, a bi-level mathematical model designed to achieve an optimal solution for worker planning in cellular manufacturing systems. A bi-level optimization problem defined as follows:

$$\begin{cases} \underset{x\in R^n, y\in R^m}{\text{Min}} F(x,y) \\ \text{subject to } G(x,y) \le 0 \\ \quad \begin{cases} \underset{y\in R^m}{\text{Min}} \quad f(x,y) \\ \text{subject to } g(x,y) \le 0 \end{cases} \end{cases}$$

$$F, f : R^n \times R^m \to R$$

$$G, g : R^n \times R^m \to R^p$$

$F$ is the objective function of the upper-level, which known as a leader and $f$ is the objective function of the lower-level which known as a follower. $G$ and $g$ are respectively the sets of constraints of the upper-level and the lower-level problems [32].

Since there are several objectives that are optimized at a different level, the possible use of multi-objective optimization should be discussed here. What is the main difference between multi-objective and multi-level optimizations? A multi-objective problem doesn't quite optimize all objectives simultaneously; rather, it treats all objectives as equally important and will give a trade-off curve or Pareto front. At some points of that curve, a trade-off in favor of objective1 is made, at others, in favor of other objectives. All points along the curve are feasible for the same set of constraints, and this set of constraints does not depend on either objective. While multi-level programming cares about one objective ($f(x)$), and the optimum value of $f(x)$ over a set constraints and lower level model is obtained which happens to be defined using another optimization (the lower level program). Here exactly one optimal solution, though perhaps many optimizers are searched [32].

Cell formation is a strategic decision and worker planning is a tactical decision. These two aspects cannot be planned centrally, and decision making in this regard is decentralized. Aiming to make a decentralized and integrated decision, a bi-level approach has been provided here. The desired goals of the optimization model are voids and exceptional elements and workers' assignment minimization. Two objective functions in two levels are considered. The leader addresses a more important issue, which is reducing the number of voids and exceptional elements. In the second level or follower, promoting a sense of synergy between the workers has been taken into consideration to maintain an innovative and dynamic organization in the long term.

## Sets

$C$ = Number of cells.
$M$ = Number of machine types.
$P$ = Number of part types.
$W$ = Number of workers.

## Parameters

$\text{LM}_c$ = Lower bound of cell $c$ in terms of the number of machine types;
$\text{LP}_c$ = Lower bound of cell $c$ in terms of the number of part types;
$\text{LW}_c$ = Lower bound of cell $c$ in terms of the number of workers;
$\text{UW}_c$ = Upper band of cell $c$ in terms of the number of workers;

$$A_{pm} = \begin{cases} 1 & \text{if machine type } m \text{ required for part type } p, \\ 0 & \text{otherwise} \end{cases}$$

$$B_{pmw} = \begin{cases} 1 & \text{if part type } p \text{ can be processed on machine type } m \text{ with worker } w, \\ 0 & \text{otherwise} \end{cases}$$

$$R_{ww'} = \begin{cases} 1 & \text{if worker } w \text{ interested in working with worker } w', \\ 0 & \text{otherwise.} \end{cases}$$

### Decision variables

$$x_{mc} = \begin{cases} 1 & \text{if machine type } m \text{ is assigned to cell } c, \\ 0 & \text{otherwise} \end{cases}$$

$$y_{pc} = \begin{cases} 1 & \text{if part type } p \text{ is assigned to cell } c, \\ 0 & \text{otherwise} \end{cases}$$

$$z_{wc} = \begin{cases} 1 & \text{if worker } w \text{ is assigned to cell } c, \\ 0 & \text{otherwise} \end{cases}$$

$$d_{pmwc} = \begin{cases} 1 & \text{if part } p \text{ is processed on machine type } m \text{ with worker } w \text{ in cell } c, \\ 0 & \text{otherwise.} \end{cases}$$

### Mathematical model

*Leader level*

$$\text{Min} = \sum_{c=1}^{C} \left[ \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{w=1}^{W} y_{pc} x_{mc} z_{wc} - \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{w=1}^{W} y_{pc} x_{mc} z_{wc} d_{pmwc} \right] \tag{2.1}$$

$$+ \sum_{p=1}^{P} \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{w=1}^{W} [y_{pc} x_{mc} (1 - z_{wc}) d_{pmwc}] \tag{2.2}$$

$$+ \sum_{p=1}^{P} \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{w=1}^{W} [2 \times x_{mc} (1 - y_{pc}) (1 - z_{wc}) d_{pmwc}] \tag{2.3}$$

$$+ \sum_{p=1}^{P} \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{w=1}^{W} [x_{mc} (1 - y_{pc}) z_{wc} d_{pmwc}] \tag{2.4}$$

The first term in the leader objective function, *i.e.* (2.1), the total number of voids minimization, the terms (2.2)–(2.4) calculate the number of exceptional elements for parts which computed based on the status, availability of corresponding worker and machine.

The constraints are:

$$\sum_{m=1}^{M} x_{mc} \geq \text{LM}_c \qquad \forall c; \tag{2.5}$$

$$\sum_{p=1}^{P} y_{pc} \geq \text{LP}_c \qquad \forall c; \tag{2.6}$$

$$d_{pmwc} \leq B_{pmw} x_{mc} \qquad \forall p, m, w, c; \tag{2.7}$$

$$\sum_{c=1}^{C} \sum_{w=1}^{W} d_{pmwc} = A_{pm} \qquad \forall p, m; \tag{2.8}$$

$$\sum_{c=1}^{C} y_{pc} = 1 \qquad \forall p; \tag{2.9}$$

$$x_{mc}, y_{pc}, z_{wc}, d_{pmwc} \in \{0, 1\} \qquad \forall p, m, w, c. \tag{2.10}$$

In the lower-level objective function of this mathematical model and in each cell, we are looking for worker interest maximization in order to work together. Inequality (2.5) controls the assignment of a minimum number of machines to a cell. Inequality (2.6) controls the minimum number of parts which processed in each cell.

Inequality (2.7) ensures that when machine type $m$, is not in cell $c$, then $d_{pmwc}$ is equal to zero. Equation (2.8) ensures that if the part $p$ need to process by machine $m$, there is a cell and just one like $c$, which contains this machine and worker $w$ who work on it to process part $p$ on this cell. Equation (2.9) ensures that a specific part is assigned to one cell only. Finally, equation (2.10) suggests that $x_{mc}$, $y_{pc}$, $z_{wc}$ and $d_{pmwc}$ are binary decision variables.

*Follower level*

$$\text{Max} = \sum_{w=1}^{W}\sum_{w'=1}^{W}\sum_{c=1}^{C} R_{ww'} z_{wc} z_{w'c}. \tag{2.11}$$

The constraints are:

$$\sum_{w=1}^{W} z_{wc} \geq LW_c \qquad \forall c; \tag{2.12}$$

$$\sum_{w=1}^{W} z_{wc} \leq UW_c \qquad \forall c; \tag{2.13}$$

$$\sum_{c=1}^{C} z_{wc} = 1 \qquad \forall w; \tag{2.14}$$

$$\sum_{p=1}^{P}\sum_{m=1}^{M} d_{pmwc} \geq z_{wc} \qquad \forall w, c. \tag{2.15}$$

In the follower objective as the second level, *i.e.* (2.11), we are looking for maximizing the interest of working together between workers who work in a special cell. At this level of the mathematical model, constraints (2.12) and (2.13) control the assignment of minimum and maximum workers to a cell. Equality (2.14) ensures that a specific worker is assigned to one cell only. Equation (2.15) defined to ensure that if worker $w$ assigned to cell $c$ there is at least one part like $p$ in this cell which processed on machine type $m$by working the worker $w$on that machine. Cell formation problems classified as NP-hard problems [25]. Bi-level programming has an inherent framework and even with contiguous variables in the linear model to solve strongly NP-hard problems [32]. Several solution techniques have been proposed for solving this kind of problems. The difficulty consideration of multi-level programming, metaheuristic algorithms are the most useful suggested methods. Metaheuristic algorithm classes can be arranged according to the type of strategies as shown in Figure 2.

For solving the proposed mathematical model, this paper developed nested approaches with genetic algorithm (GA) and particle swarm optimization (PSO) algorithm which is explained in the next section.

## 3. Nested bi-level genetic algorithm (NBL-GA)

Genetic algorithm is common optimization tools for engineering problems which was introduced by John Holland from the University of Michigan in 1975 [18]. Genetic algorithms are special types of evolutionary algorithms that utilize biological anabolic techniques such as inheritance and mutation [26]. In fact, genetic algorithms use the principles of Darwin's natural selection to find the optimum formula for estimating or matching patterns. Genetic algorithms are programming techniques that make use of genetic evolution as a problem-solving scheme [29]. The problem to be solved is the input and the solutions are coded per a scheme which is called the fitness function that evaluates every candidate. Two search operators are present in this algorithm: Crossover and Mutation. Mutation creates a neighborhood based on the offspring while crossover selects two solutions as the parents and creates two offspring solutions by combining them thus searching for

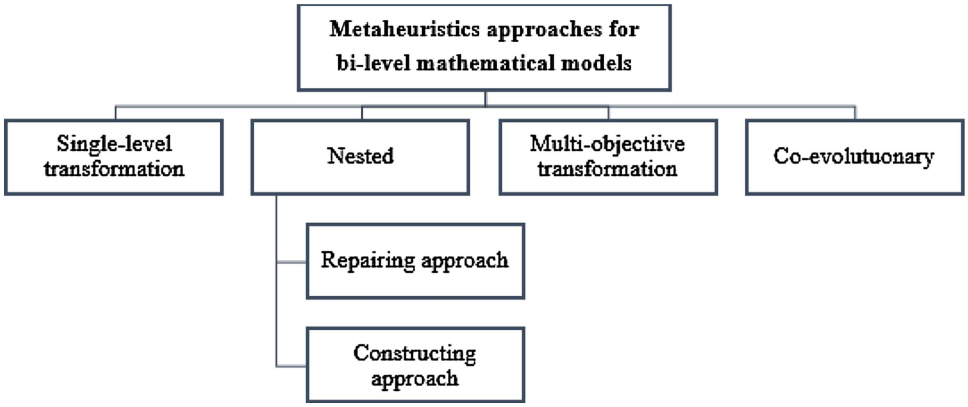


FIGURE 2. Metaheuristics solving approach for bi-level mathematical models.

```
parent=_generate_parent()
pbest=get_fitness(parent)
display(pbest)
if compare_best_optimal(pbest,optimal):
    return pbest
flag=true
gens=0
while flag:
  offspring=_mutate(pbest,gens)
  produce(offspring)
  offbest=get_fitness(offspring)
  if pbest>=offbest:
      continue
  display(offbest)
  if compare_best_optimal(offbest,optimal):
      return offbest
  pbest=offbest
  parent=offspring
  gens=gens+1
```

FIGURE 3. The executable code of the genetic algorithm (with Python programming language).

the possibility space of the solution. The algorithm performs the focus and variety of metaheuristics phases blindly in the solution space. The executable code of the genetic algorithm with the mentioned steps is as follows (Fig. 3).

The genetic algorithm, as one of the most well-known metaheuristic algorithms, can be molded into different forms for various problems. Given that many researchers have utilized nested metaheuristic algorithms for solving multi-level problems as accurate methods such as Karush–Kuhn–Tucker (KKT) [16, 21, 22, 32]. Therefore, this type of algorithm was adopted for solving the two-level problem in this study. The nested bi-level genetic algorithm executable code has been shown in Figure 4.

```
(x,y,d)=_generate_parent_upper_level()
(z,d)=_generate_parent_lower_level()
pbest_upper_level=get_fitness_upper_level(x,y,d)
pbest_lower_level=get_fitness_lower_level(z,d)

if compare_best_optimal(pbest_upper_level,optimal_upper_level):
      if compare_best_optimal(pbest_lower_level,optimal_lower_level):
            return F1(pbest_upper_level,pbest_lower_level)
            return F2(pbest_lower_level)
flag=true
gens=0
while flag:
   pbestt_upper_level=_crossover(pbest_upper_level,gens)
   offspring_upper=_mutate(pbestt_upper_level,gens)
   produce(offspring_upper)
   offbest_upper_level=get_fitness_upper_level(offspring_upper)
   if pbest_upper_level>=offbest_upper_level:
         continue
   if compare_best_optimal(offbest_upper_level,optimal_upper_level):
         co_evolve_lower_layer(offbest_upper_level)
         pbestt_lower_level=_crossover(pbest_lower_level,gens)
         offspring_lower=_mutate(pbestt_lower_level,gens)
         produce(offspring_lower)
         offbest_lower_level=get_fitness_lower_level(offspring_lower)
         if pbest_lower_level>=offbest_lower_level:
               continue
         if compare_best_optimal(offbest_lower_level,optimal_lower_level):
               co_evolve_lower_layer(offbest_upper_level)
               return F1(offbest_upper_level)
               return F2(pbest_lower_level)

   pbest_upper_level=offbest_upper_level
   pbest_lower_level=offbest_lower_level
   (x,y,d)=offspring_upper
   (d,z)=offspring_lower
   gens=gens+1
```

FIGURE 4. The executable code of the NBL-GA (Customized genetic algorithm with Python programming language).

### 3.1. Operators

As illustrated in Figure 4, several crossover and mutation operators are developed in the related field. Here, a single-point and double-point crossovers are selected and for mutation, three operators include swap, inversion, and displacement are applied [5, 23, 28].

### 3.2. Parent selection mechanism

Two stages in the development cycle are present in which competition occurs based on fitness, selecting candidates for reproduction and selecting candidates to make it to the next generation. Selecting the parent is to distinguish individuals based on their fitness, which means to permit better individuals to be parents for the next generation. Selecting parent along with a reproduction selection mechanism improve the character of the population. Parent selection is a possibility-based process in the genetic algorithm. Hence, candidates of higher fitness are more probable to be selected as the parent. Nevertheless, lower fitness candidates have a lower but a positive probability of being chosen. If the process was not possible-based, the search would have become greedy and the algorithm could get trapped in local optimums. The roulette wheel is one of the most common selection algorithms and in this paper has been used for parent selection.

## 4. Nested bi-level particle swarm optimization (NBL-PSO)

The particle swarm optimization (PSO) algorithm was proposed by Kennedy and Eberhart in 1995 [11]. This is a search algorithm which is modeled based on the social behavior of bird flocks. This algorithm was first used to discover the patterns governing the synchronous flight of birds with their sudden changes of the path and for the optimum transformation of the flock, in which particles flow in the search space [9]. Dislocation of particles in the search space is governed by the knowledge and experience of their own and their neighbors'. Therefore, the status of the other masses of particles is effective on the search for one particle. The result of modeling this social behavior is a search procedure in which particles tend to successful regions. Particles learn from one another and approach their best neighbors based on the knowledge they acquire. The algorithm is established along the principle that each particle adjusts its location in the search space considering the best location and the best fix in its neighborhood. Finally, the pseudo code of the bi-level nested particle swarm algorithm is presented in Figure 5.

### 4.1. Initial generation

There are several entities in the PSO algorithm that are called particles and are scattered in the search space of the function. Each particle computes the target function in the position. Then, analyzing the information about its current and previous locations along with data from one or more particles of the best population, it picks out a way to go. A step of the algorithm ends as all the particles present in the population choose a direction to go. These steps are repeated until the desirable solution is found. Each particle has a certain competency which is calculated by a competency function. In the bird's flock model, the closer particle to the target in the search space has the higher competency. Moreover, each particle has a velocity directing its migration. Particles continue their motion in the search space by following the current optimum particles, as a group of PSO particles is randomly generated at the start and seek the optimum solution by renewing generations. Particles are generated with random positions and velocities at the beginning of the algorithm while in the subsequent steps, the placement and velocity of the particles are assigned considering the data from the previous step.

### 4.2. Velocity updating

To move particles, there is a velocity operator. Particles are improved at each step based on two of the best values. One is the best position that the particle has taken, which recorded as $P_{\text{best}}$, and the other the best

overall position that has been taken by a particle in the population, represented as $G_{\text{best}}$. After determining the best values, the velocity of each particle is updated using equation (4.1).

$$V_{ij}(t+1) = W \times V_{ij}(t) + c_1 r_{1j}(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[g_j(t) - x_{ij}(t)] \tag{4.1}$$

where:

**$W$:** Inertial weight, a control factor for exploration and exploitation.
For $W \geq 1$, velocity gradually increases, the swarm scatters and the particles become unable to get back to favorable regions (greater exploration for larger $W$).
For $W < 1$, the velocity of the particles gradually decreases (greater exploitation for smaller $W$).
**$V_{ij}(t)$:** Velocity of particle $i$in dimension $j$ which prevents sudden changes in the motion of population.
**Acceleration coefficients $c_1$ and $c_2$:** determine how close the particle is to $P_{\text{best}}$ and $G_{\text{best}}$.

```
Upper Level
    Generate Initial Variables Vectors (X, Y, d)
    Do
            Check the Generated Vectors Feasibility
            If Not Feasible
                Correct Vectors
    While the Feasibility Conditions Satisfied
    Call Lower Level Function for calculating Z and F2
    For each particle
    Compute F1 using Z, X, Y, d
    %%updating particle's best fitness value%%
    If F1 is better than PBest
    set current value as the new PBest
    End For
    %%updating population's best fitness value%%
    Set GBest to the best fitness value of all particles
    While the stopping criterion will be satisfied will be satisfied
        Calculate the Particle Velocity according to equation (16)
        Updating Particle Position according to equation (17)
        Do
            Check the Generated Vectors Feasibility
            If Not Feasible
                Correct Vectors
        While the Feasibility Conditions Satisfied
        Call Lower Level Function for calculating Z and F2
        Compute F1 using Z, X, Y, d
        update particle's best fitness value (PBest)
        Sort Ascending the population based on F1
        update population's best fitness value (GBest)
    End While;
    Print The best F1*, F2*, CPU Time
End.
```

FIGURE 5. Pseudo code of the NBL-PSO algorithm.

```
.....
Lower Level
    Generate Initializing Variable Vector Z
    Do
        Check the Generated Z Vector Feasibility
        If Not Feasible
            Correct Z Vector
    While the Feasibility Conditions Satisfied
    F2 Computation using (Z, d)
    %%updating particle's best fitness value%%
    If F2 is better than PBest
    set current value as the new PBest
    End For
    %%updating population's best fitness value%%
    Set GBest to the best fitness value of all particles
    While the stopping criterion will be satisfied will be satisfied
        Calculate the Particle Velocity
        Updating Particle Position
        Do
         Check the Generated Z Vector Feasibility
         If Not Feasible
            Correct Z Vector
        While the Feasibility Conditions Satisfied
        F2 Computation using (Z, d)
        update particle's best fitness value (PBest)
        Sort Descending the population based on F2
        update population's best fitness value (GBest)
    End While;
    Provide F2*, Z for Upper Level
End Lower Level;
```

FIGURE 5. Continued.

**Accidental coefficients $r_{1j}$ and $r_{2j}$:** with uniform distribution in [0, 1].
**$P_{ij}(t)$:** The best position experienced by particle $i$ in dimension $j$ up to the time $t$ ($P_{\text{best}}$).
**$g_j(t)$:** The best overall position up to the moment $t$ ($G_{\text{best}}$).
**$x_{ij}(t)$:** The current position of particle $i$ in dimension $j$ at the time $t$.

The right side of equation (4.1) is composed of three parts, the first of which is the current velocity of the particle, the second and third parts are responsible for changing the velocity and directing of the particle. If we do not consider the first part of the equation, velocities of particles are determined only considering the current situation and the best conditions experienced by the particle and the population. Therefore, the best particle is maintained in its place while the others run toward it. The convection of particles without the first part of equation (4.1) is a process that gradually shrinks the search space, resulting in a local search around the best particle. On the contrary, if only the first segment of equation (4.1) is considered, particles go on their way until they reach the region boundary performing a somewhat overall search. The particles cooperate and exchange information about their positions. The basic PSO cooperation is as follows:

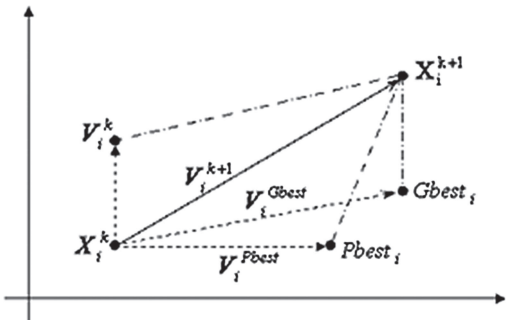


FIGURE 6. A sample step of each particle in the PSO algorithm.

– A particle has its own attributed neighborhoods.
– Particles are aware of their neighboring particles and utilize the position of the particle with the best match.
– The positions can be simply made use of adjusting velocity and consequently the location of the particle.

### 4.3. Position update

After updating the velocity of particles, the new location of each particle can be computed by the formula (4.2).

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{4.2}$$

where $x_{ij}(t)$ shows the current position of particle $i$ in dimension $j$, $x_{ij}(t+1)$ the next position of particle $i$ in dimension $j$, and $v_{ij}(t+1)$ shows the next velocity of the position of pth article $i$ in dimension $j$. Each particle in search assumes a new velocity based on its previous velocity, its own best position, the overall best position, and extends to the new position (Fig. 6).

The velocity and location of the particles are modified in such manner for each iteration until the optimum solution is attained.

## 5. Individual solution representation

The primary construction of proposed chromosome for $m = M$, $c = C$, $w = W$ and $p = P$ is presented in Figure 7, where the length of the upper level chromosome is equal to $m + p$ and the length of the lower level chromosome is equal to $w$ and also each gene on the chromosome is filled by a random integer number between [1, c]. For example, a related gene of $m_4$ and $c_j = 3$ show that the fourth machine could be allocated to the third cell.

With the assumption that $m = 5, k = 3, w = 9$ and $i = 4$ the numerical chromosome is presented as Figure 8.

Tables 2–4 have been shown the NBL-GA algorithm details in a numeric sample respectively.

However, for some algorithms such as PSO that use the chromosome with continuous numbers, the composition of this array is done in two steps. In the beginning, each gene is filled by a random continuous number between [1, k]. Then, after performing the various operators of the algorithm on this continuous chromosome, the final version of the chromosome is obtained by rounding these continuous numbers. With the assumption $m = 5, k = 3$ and $i = 4$ the numerical chromosomes are presented as Figure 9.

## 6. Proving proposed model using Taguchi's experiment design

Taguchi method is applied to select the best value of each required parameter in metaheuristic algorithms. This method was developed by Taguchi to select the best value of each parameter, instead of taking all possible

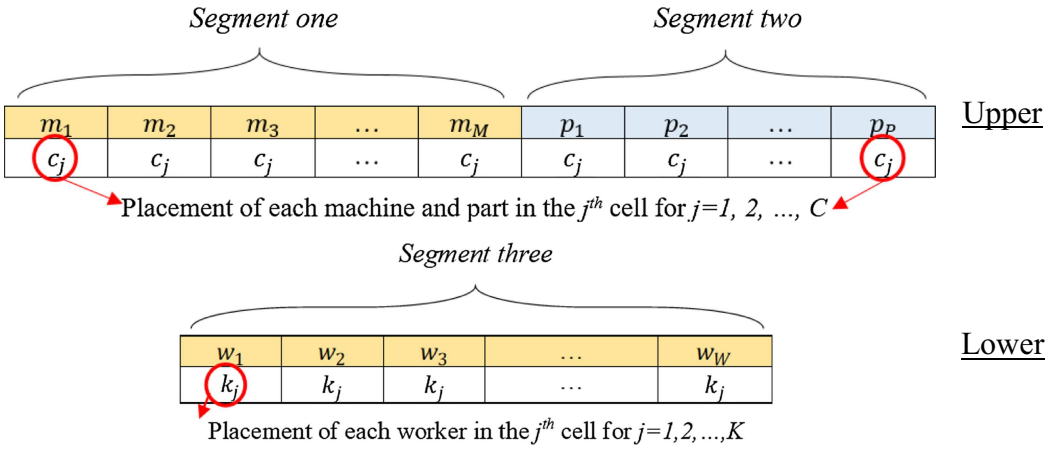


FIGURE 7. The main structure of the proposed chromosome.

| $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 1 | 3 | 2 | 1 | 3 | 1 | 2 |

| $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | $w_6$ | $w_7$ | $w_8$ | $w_9$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 3 | 1 | 3 | 2 | 1 |

FIGURE 8. The numerical example of chromosome.

TABLE 2. Feasibility check in the upper and lower levels ($m = 3$, $p = 2$, $w = 4$, $c = 3$).

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Infeasible Chromosome** | **0** | **0** | **0** | **0** | **0** | 1 | 2 | 3 | 1 | 2 | 1 | **0** | 2 | 3 | 1 |
| | 1 | 2 | 1 | 3 | **0** | **0** | **0** | **0** | 3 | 1 | 1 | 2 | | | |
| **Check and repair** | **Cells have no machines and parts.** | | | | | **Workers are not in any cell.** | | | | | **Machine 2 is not in any cell.** | | | | |
| **Repaired chromosome** | **3** | **1** | **2** | **1** | **3** | 1 | 2 | 3 | 1 | 2 | 1 | **3** | 2 | 3 | 1 |
| | 1 | 2 | 1 | 3 | **2** | **1** | **3** | **1** | 3 | 1 | 1 | 2 | | | |

TABLE 3. Cross-over of two chromosomes.

| | | | | | |
|---|---|---|---|---|---|
| **Parent 1** | 3 | 1 | 2 | 1 | 3 |
| | 1 | 2 | 1 | 3 | |
| **Parent 2** | 1 | 3 | 2 | 3 | 1 |
| | 3 | 1 | 1 | 2 | |
| **Child 1** | 1 | 3 | 2 | 3 | 1 |
| | 1 | 2 | 1 | 2 | |
| **Child 2** | 3 | 1 | 2 | 1 | 3 |
| | 3 | 1 | 1 | 3 | |

TABLE 4. Ranked chromosomes of the initial population.

| Rank | Chromosomes | | | | | Fitness value |
|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 2 | 3 | 1 | 56 |
| | 1 | 2 | 1 | 3 | | |
| 2 | 2 | 3 | 1 | 1 | 2 | 73 |
| | 3 | 3 | 2 | 1 | | |
| 3 | 2 | 1 | 3 | 2 | 3 | 81 |
| | 1 | 1 | 3 | 2 | | |
| 4 | 1 | 2 | 1 | 1 | 2 | 98 |
| | 1 | 2 | 1 | 1 | | |
| … | … | | | | | … |

| $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|---|---|---|---|
| 2.8 | 2.5 | 1.3 | 2.6 | 2.4 | 1.25 | 2.8 | 1.4 | 2.3 |

| $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 1 | 3 | 2 | 1 | 3 | 1 | 2 |

FIGURE 9. The numerical example of continuous chromosome.

experiments [31]. The problem in this study was a bi-level mathematical model, in which only the first-level objective function is used for regulating parameters, based on the Kuo *et al.* [20]. Equation (6.1) calculates the signal to noise metric and indicates how "Smaller-the-Better" select tuned parameters:

$$\mathrm{SN} = -10 \log \left( \frac{\sum_{i=1}^{n} Y^2}{n} \right) \tag{6.1}$$

where $n$ is the number of observations and $Y$ is the observed data. In this paper, nine different test problems designed for proving the proposed mathematical model which $w$, $p$, $m$, and $c$ are defined as follows (Tab. 5).

The value of $A_{pm}$, $B_{pmw}$ and $R_{ww\prime}$ are calculated randomly. The value of $\mathrm{LM}_c$, $\mathrm{LP}_c$, $\mathrm{LW}_c$, and $\mathrm{UW}_c$ for all cells are the same and assigned according to the Table 6.

Parameters of each algorithm and proposed values of Taguchi's method are shown in Table 7.

For each test problem, a set of Taguchi experiments is performed. All the experiments for the NBL-GA and the NBL-PSO algorithms on the first test problem are shown in Tables 8 and 9, respectively.

The best values of each parameter on each problem are obtained according to the SN diagram for the NBL-GA (Fig. 10). The best value of each parameter in each problem is the parameter level with the highest SN value. For example, 0.7 and 0.1 are the best values for *Pc* and *Pm*, on the first test problem respectively.

Likewise, the SN diagram for the nested particle swarm optimization algorithm is shown in Figure 11.

Because of the random nature of metaheuristics, each problem for 20 trials is run and the average of them is used. After 6480 times running, $9 \times 20$ times for each problem with NBL-GA and $27 \times 20$ times for each problem with the NBL-PSO algorithm the best value of parameters in each problem computed as follows (Tab. 10).

TABLE 5. Problems specification.

| Problem # | Part type | Machine type | Cell number | Workforce |
|---|---|---|---|---|
| 1 | 5 | 5 | 2 | 9 |
| 2 | 8 | 10 | 3 | 12 |
| 3 | 9 | 7 | 3 | 10 |
| 4 | 10 | 15 | 4 | 18 |
| 5 | 15 | 12 | 3 | 14 |
| 6 | 18 | 11 | 3 | 15 |
| 7 | 20 | 12 | 4 | 15 |
| 8 | 25 | 15 | 4 | 20 |
| 9 | 30 | 20 | 4 | 20 |

TABLE 6. The value of $LM_c$, $LP_c$, $LW_c$, and $UW_c$.

| Parameter | Value |
|---|---|
| $LM_c$ | 2 |
| $LP_c$ | 2 |
| $LW_c$ | 3 |
| $UW_c$ | 6 |

TABLE 7. Algorithm parameters range and defined levels.

| Nested bi-level algorithms | Parameters | Parameter level | | |
|---|---|---|---|---|
| | | Level 1 | Level 2 | Level 3 |
| NBL-GA | $Pc$ | 0.7 | 0.8 | 0.9 |
| | $Pm$ | 0.05 | 0.1 | 0.15 |
| | $N$-pop | 100 | 150 | 200 |
| | Max iteration | 100 | 200 | 300 |
| NBL-PSO | $C_1$ | 0.5 | 1 | 2 |
| | $C_2$ | 0.5 | 1 | 2 |
| | $W$ | 0.5 | 0.75 | 1 |
| | $N$-pop | 100 | 150 | 200 |
| | Max iteration | 100 | 200 | 300 |

TABLE 8. The main solution for each Taguchi experiment for *NBL-GA*.

| Experiment # | *Pc* | *Pm* | *N*-pop | *Max iteration* | *F1* |
|---|---|---|---|---|---|
| 1 | 0.7 | 0.05 | 100 | 100 | 48 |
| 2 | 0.7 | 0.10 | 150 | 200 | 50 |
| 3 | 0.7 | 0.15 | 200 | 300 | 48 |
| 4 | 0.8 | 0.05 | 150 | 300 | 50 |
| 5 | 0.8 | 0.10 | 200 | 100 | 52 |
| 6 | 0.8 | 0.15 | 100 | 200 | 53 |
| 7 | 0.9 | 0.05 | 200 | 200 | 50 |
| 8 | 0.9 | 0.10 | 100 | 300 | 52 |
| 9 | 0.9 | 0.15 | 150 | 100 | 51 |

TABLE 9. The main solution for each Taguchi experiment for *NBL-PSO*.

| Experiment # | *C1* | *C2* | *W* | *N*-pop | Max iteration | *F1* |
|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0.50 | 100 | 100 | 54 |
| 2 | 0.5 | 0.5 | 0.50 | 100 | 200 | 53 |
| 3 | 0.5 | 0.5 | 0.50 | 100 | 300 | 51 |
| 4 | 0.5 | 1.0 | 0.75 | 150 | 100 | 52 |
| 5 | 0.5 | 1.0 | 0.75 | 150 | 200 | 53 |
| 6 | 0.5 | 1.0 | 0.75 | 150 | 300 | 52 |
| 7 | 0.5 | 2.0 | 1.00 | 200 | 100 | 52 |
| 8 | 0.5 | 2.0 | 1.00 | 200 | 200 | 50 |
| 9 | 0.5 | 2.0 | 1.00 | 200 | 300 | 52 |
| 10 | 1.0 | 0.5 | 0.75 | 200 | 100 | 51 |
| 11 | 1.0 | 0.5 | 0.75 | 200 | 200 | 51 |
| 12 | 1.0 | 0.5 | 0.75 | 200 | 300 | 48 |
| 13 | 1.0 | 1.0 | 1.00 | 100 | 100 | 48 |
| 14 | 1.0 | 1.0 | 1.00 | 100 | 200 | 50 |
| 15 | 1.0 | 1.0 | 1.00 | 100 | 300 | 51 |
| 16 | 1.0 | 2.0 | 0.50 | 150 | 100 | 51 |
| 17 | 1.0 | 2.0 | 0.50 | 150 | 200 | 51 |
| 18 | 1.0 | 2.0 | 0.50 | 150 | 300 | 51 |
| 19 | 2.0 | 0.5 | 1.00 | 150 | 100 | 52 |
| 20 | 2.0 | 0.5 | 1.00 | 150 | 200 | 52 |
| 21 | 2.0 | 0.5 | 1.00 | 150 | 300 | 51 |
| 22 | 2.0 | 1.0 | 0.50 | 200 | 100 | 51 |
| 23 | 2.0 | 1.0 | 0.50 | 200 | 200 | 50 |
| 24 | 2.0 | 1.0 | 0.50 | 200 | 300 | 51 |
| 25 | 2.0 | 2.0 | 0.75 | 100 | 100 | 50 |
| 26 | 2.0 | 2.0 | 0.75 | 100 | 200 | 51 |
| 27 | 2.0 | 2.0 | 0.75 | 100 | 300 | 51 |

## 7. COMPUTATIONAL RESULTS

To solve the proposed bi-level programming model through the two meta-heuristics, the MATLAB software is utilized. All program runs are made on a personal computer with Intel(R) Core (TM) i5-5200U CPU @2.20 GHz under Windows 10. Each test problem was solved using the NBL-GA and NBL-PSO algorithms according to the best parameters based on Taguchi experiments for 20 trials and the average of them is used. For example, the calculated response with the NBL-GA and NBL-PSO methods for $F_1$ respectively are 104 and 109. As well as the value of $F_2$ for this test problem with NBL-GA and NBL-PSO are 37 and 39. In Figures 12 and 13, the cell planning for the test problem no.2 with NBL-GA and NBL-PSO are shown respectively.

The computed values of the $F_1$ and $F_2$ for the test problem no.5 with the NBL-GA and NBL-PSO methods are 206, 58 and 208, 59 respectively. Figures 14 and 15 are shown the cell planning for the test problem no.5 with NBL-GA and NBL-PSO.

To compare the obtained solutions, the objective functions at the first level and second level, and CPU Time and the normalized deviation (ND) value were calculated as shown in Table 11. To calculate ND for each algorithm, the following equation was applied.

$$ND = \frac{\left|F_{1,i}^{\text{best}} - F_{1,i}^{*}\right|}{F_{1,i}^{\text{best}}} + \frac{\left|F_{2,i}^{\text{best}} - F_{2,i}^{*}\right|}{F_{2,i}^{\text{best}}} \tag{7.1}$$

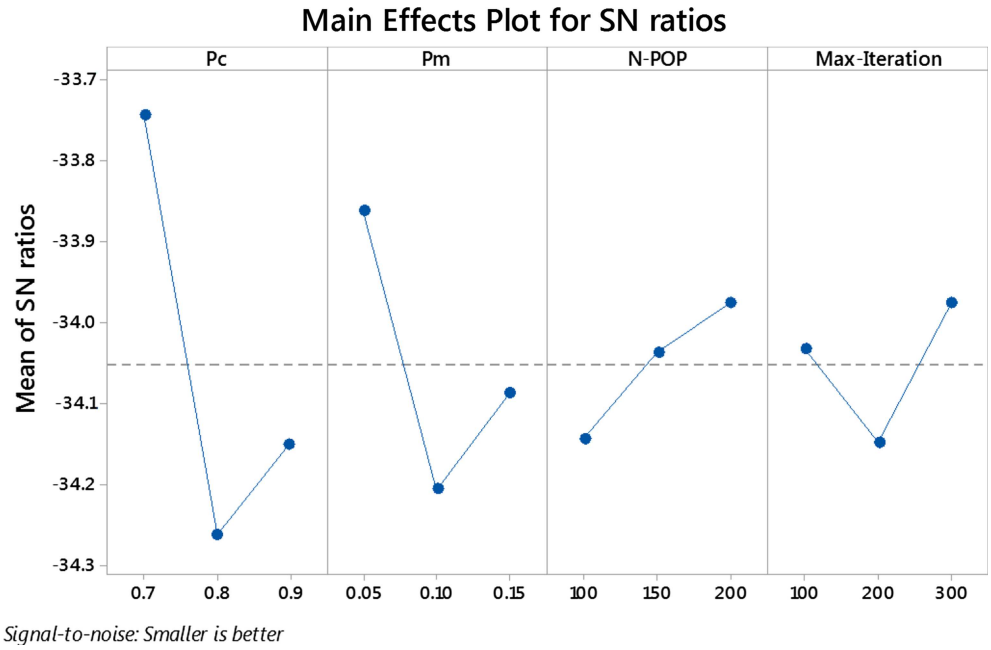


FIGURE 10. The SN diagram for the NBL-GA.

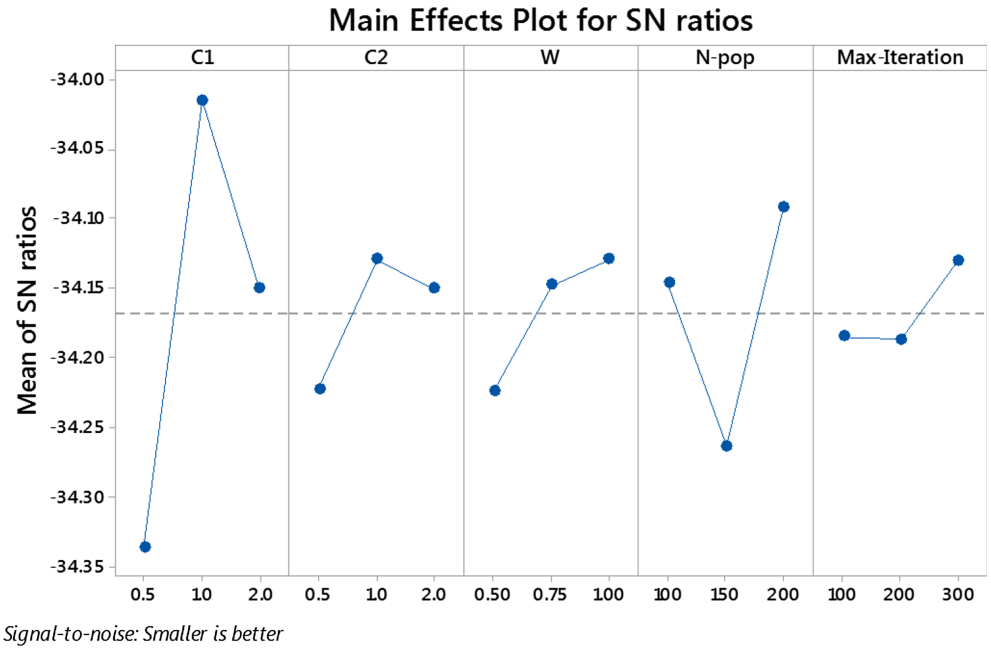


FIGURE 11. The SN diagram for the NBL-PSO.

TABLE 10. The best-tuned parameters for each problem.

| Problem # | NBL-GA | | | | NBL-PSO | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Pc* | *Pm* | *N*-pop | Max iteration | $C_1$ | $C_2$ | *W* | *N*-pop | Max iteration |
| 1 | 0.7 | 0.05 | 200 | 300 | 1 | 1 | 1 | 200 | 300 |
| 2 | 0.9 | 0.15 | 200 | 200 | 1 | 1 | 0.75 | 150 | 200 |
| 3 | 0.8 | 0.05 | 150 | 300 | 1 | 2 | 0.75 | 200 | 300 |
| 4 | 0.8 | 0.15 | 150 | 300 | 1 | 0.5 | 1 | 100 | 300 |
| 5 | 0.7 | 0.1 | 200 | 300 | 1 | 2 | 0.75 | 200 | 300 |
| 6 | 0.8 | 0.1 | 200 | 300 | 0.5 | 2 | 0.75 | 150 | 300 |
| 7 | 0.8 | 0.15 | 200 | 300 | 2 | 1 | 1 | 150 | 200 |
| 8 | 0.9 | 0.15 | 200 | 300 | 1 | 2 | 1 | 100 | 300 |
| 9 | 0.9 | 0.05 | 200 | 300 | 0.5 | 2 | 0.75 | 200 | 200 |

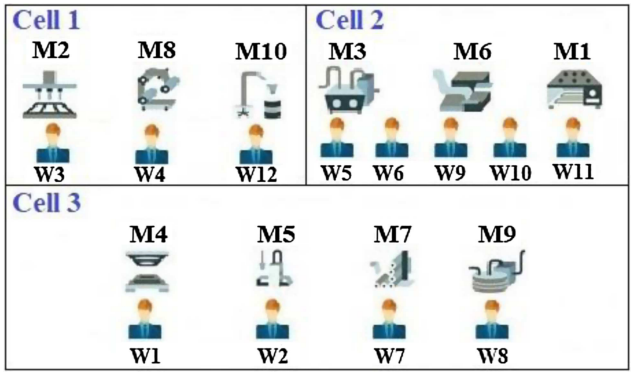


FIGURE 12. The cell planning for test problem no.2 with NBL-GA.

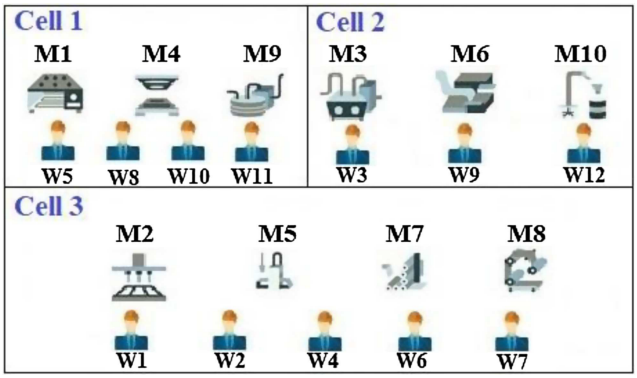


FIGURE 13. The cell planning for test problem no.2 with NBL-PSO.

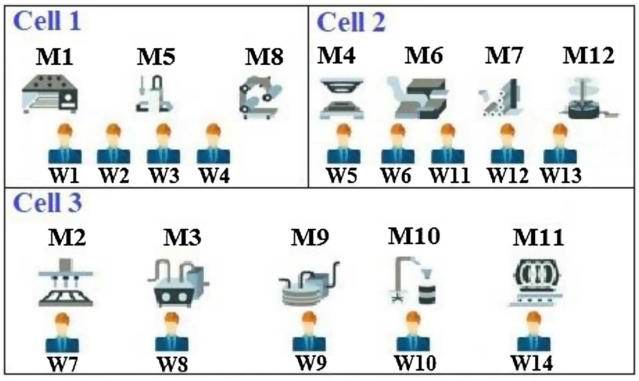


FIGURE 14. The cell planning for test problem no.5 with NBL-GA.

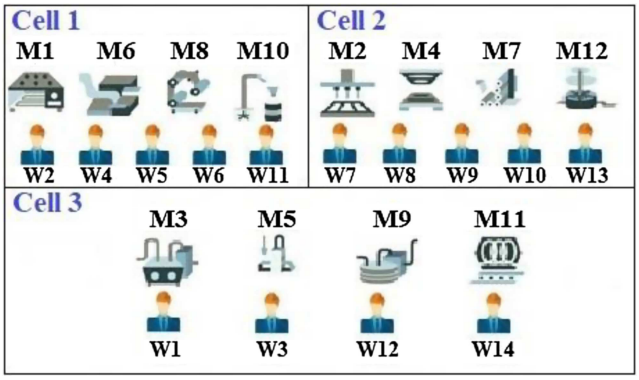


FIGURE 15. The cell planning for test problem no.5 with NBL-PSO.

TABLE 11. The result and normalized deviations of each problem and each algorithm.

| Problem number | NBL-GA | | | | NBL-PSO | | | |
|---|---|---|---|---|---|---|---|---|
| | Normalized deviation | CPU Time | $F_1$ | $F_2$ | Normalized deviation | CPU time | $F_1$ | $F_2$ |
| 1 | 0.094 | 1282.76 | 50 | 29 | 0.020 | 918.96 | 51 | 32 |
| 2 | 0.051 | 4683.89 | 104 | 37 | 0.048 | 1903.51 | 109 | 39 |
| 3 | 0.000 | 3178.04 | 76 | 27 | 0.092 | 2639.73 | 83 | 27 |
| 4 | 0.094 | 16 200.32 | 202 | 58 | 0.059 | 6445.97 | 214 | 64 |
| 5 | 0.051 | 13 102.40 | 206 | 56 | 0.010 | 4666.56 | 208 | 59 |
| 6 | 0.050 | 17 408.70 | 232 | 57 | 0.013 | 5473.56 | 235 | 60 |
| 7 | 0.100 | 28 563.38 | 214 | 45 | 0.051 | 11 602.76 | 225 | 50 |
| 8 | 0.012 | 64 989.63 | 352 | 80 | 0.139 | 17 780.23 | 401 | 81 |
| 9 | 0.080 | 97 783.29 | 434 | 81 | 0.062 | 37 506.48 | 461 | 88 |
| Ave. | 0.0591 | | | | 0.0550 | | | |

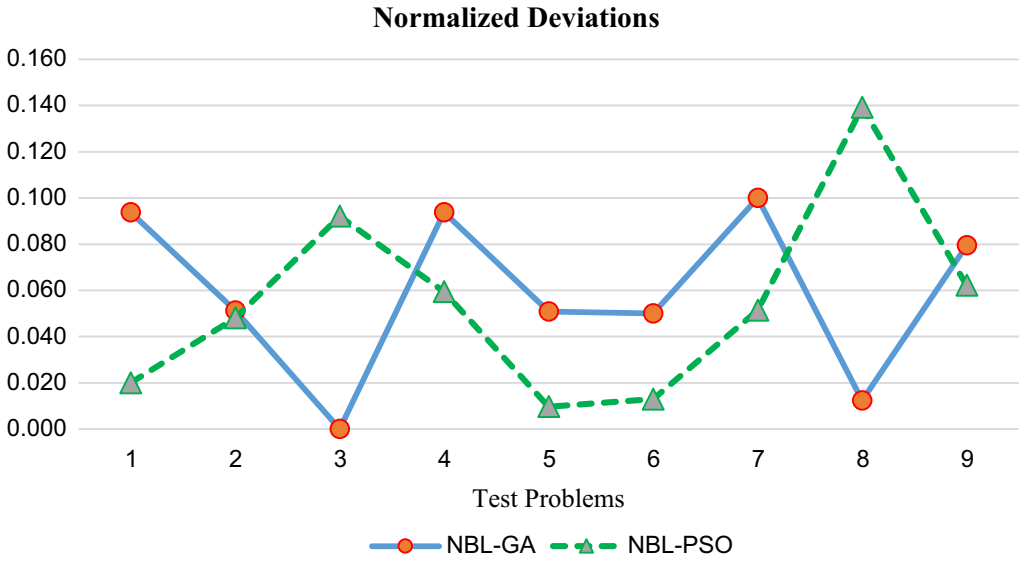


FIGURE 16. The normalized deviation for the NBL-GA and NBL-PSO in test problems.

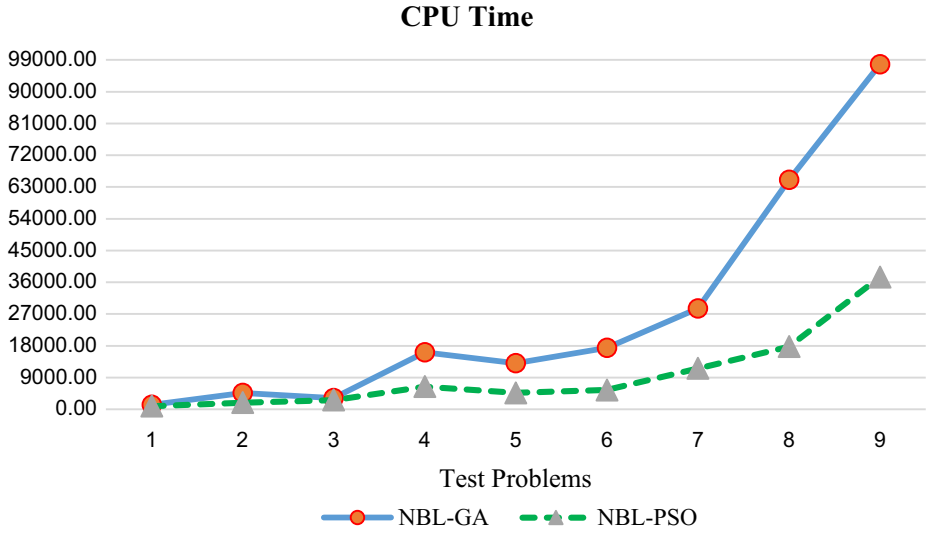


FIGURE 17. The running CPU times for the NBL-GA and NBL-PSO in test problems.

$F_{1,i}^{\text{best}}$ and $F_{1,i}^{\text{best}}$ are the best answer between NBL-GA and NBL-PSO algorithms output for the leader and the follower objective functions in $i$th problem respectively. $F_{1,i}^{*}$ and $F_{2,i}^{*}$ are the values of objective functions for each algorithm in the $i$th problem.

For better comparability, the normalized deviation values calculated for each problem and each solving method are presented in Figure 16. Based on the average normalized deviation, the results obtained from NBL-PSO have a lower deviation. However, in small-size problems, NBL-GA has less deviation than NBL-PSO.

The CPU time for each test problem is presented in Figure 17. Examining CPU time shows that when the problem size increases, the runtime ratio using NBL-GA will have more growth than NBL-PSO. Based on ND

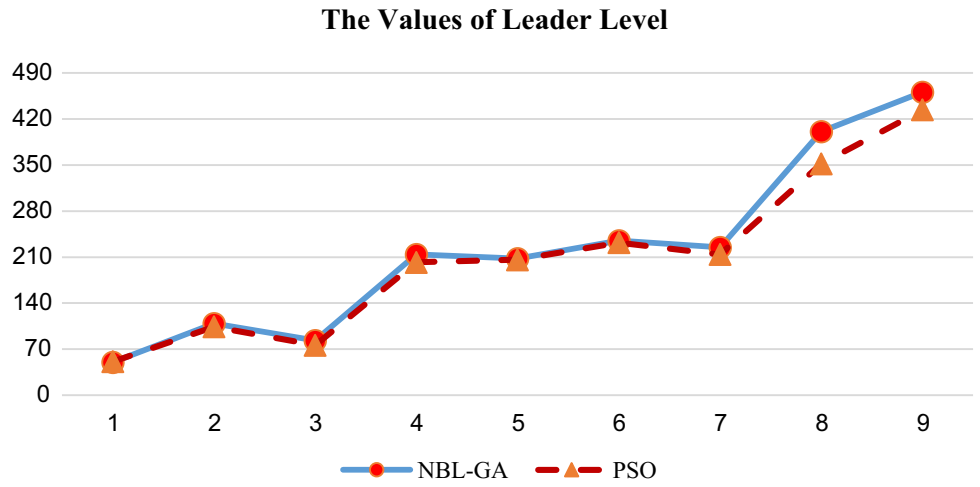


FIGURE 18. The values of the upper level in NBL-GA and NBL-PSO comparison.

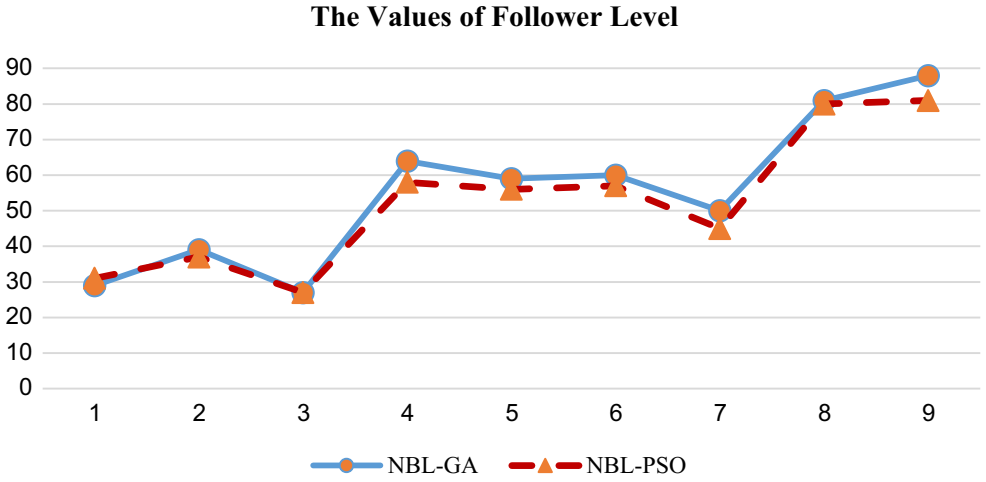


FIGURE 19. The values of the lower level in NBL-GA and NBL-PSO comparison.

and CPU time indices, in the medium to large-size problems, the NBL-PSO method is more efficient than NBL-GA.

The obtained values of the leader and follower levels in NBL-GA and NBL-PSO have been compared in Figures 18 and 19.

In the upper level, the solutions which have been obtained from NBL-GA are much better than the solutions obtained by NBL-PSO. Nevertheless, at the lower level, the results which have been obtained from NBL-PSO are better than NBL-GA. Moreover, to compare these metaheuristics more carefully, ANOVA test was utilized with a 95% confidence level. Generally, applying ANOVA to compare metaheuristics' performance is common in the literature such as [6, 10, 28]. To this end, it was used for nine tests in Table 8 based on three criteria including $F_1$, $F_2$, and CPU time. Finally, the mean and standard deviation of each algorithm for the nine tests and all criteria were reported in Table 12. Also, the other details of one-way ANOVA are illustrated in Table 13. As shown in Table 12, the significant difference was observed between the two offered meta-heuristics based on

TABLE 12. The mean and standard deviation for each algorithm in nine tests.

| Metric | ↓ $F_1$ | | ↑ $F_2$ | | ↓CPU Time | |
|---|---|---|---|---|---|---|
| | GA | PSO | GA | PSO | GA | PSO |
| Mean | **207.8** | 220.8 | 52.22 | **55.56** | 27466 | **9882** |
| StDev | **125.3** | 137.3 | **19.78** | 20.85 | 32754 | **11 637** |

TABLE 13. The results of ANOVA.

| Criteria | Source | DF | SS | MS | $F$-statistic | $P$-value | Test results |
|---|---|---|---|---|---|---|---|
| $F_1$ | Factor | 1 | 761 | 760.5 | 0.04 | 0.837 | Null hypothesis is not rejected |
| | Error | 16 | 276 585 | 17 286.6 | | | |
| | Total | 17 | 277 346 | – | | | |
| $F_2$ | Factor | 1 | 50.00 | 50.00 | 0.12 | 0.732 | Null hypothesis is not rejected |
| | Error | 16 | 6607.78 | 412.99 | | | |
| | Total | 17 | 6657.78 | – | | | |
| CPU time | Factor | 1 | 1 391 363 014 | 1 391 363 014 | 2.30 | 0.149 | Null hypothesis is not rejected |
| | Error | 16 | 9 665 957 023 | 604 122 314 | | | |
| | Total | 17 | 11 057 320 036 | – | | | |

$F_1$, $F_2$, and CPU time. For example, the GA algorithm was the better than PSO in terms of $F_1$ (a minimization criterion) in both of Mean and StDev metrics which this significant difference is shown in Figure 20a. Likewise, the PSO algorithm was the better than GA in term of $F_2$ (a maximization criterion) in Mean metric which this significant difference is shown in Figure 20b. Also, the PSO algorithm was the better than GA in term of CPU time (a minimization criterion) in both of Mean and StDev metrics which this significant difference is shown in Figure 20c. Finally, since in term of $F_2$ and CPU time, the PSO was better than GA (two criteria among three criteria), thus, the PSO as the best algorithm is reported.

### 7.1. Proposed method validation: comparison with other research

In order to proposed method validation, this paper has been used another related research (*e.g.*, reference [4]). In the mentioned research, the authors presented an appropriate method for improved dynamic cellular manufacturing by considering human factors. We have been defined the same condition as this reference too. The particular parameters settings for validating the proposed method have been shown in Table 14.

Also, other validation characteristics have been presented in Table 15.

Based on defined different problems as illustrated in Table 15, the results are shown in following Tables 16–18.

Also, the comparison of this paper and the mentioned research using computational time based on different generation number has been presented in Figure 21.

As these results show, the proposed method of this paper, has a better result in problems with large size. Hence, the computational time is in normal status than other techniques such as mentioned reference.

## 8. CONCLUSION

Improvement of product quality combined with the reduction of production costs is always considered as an important issue in cellular manufacturing systems. Due to customers' requirements and market pull, man-

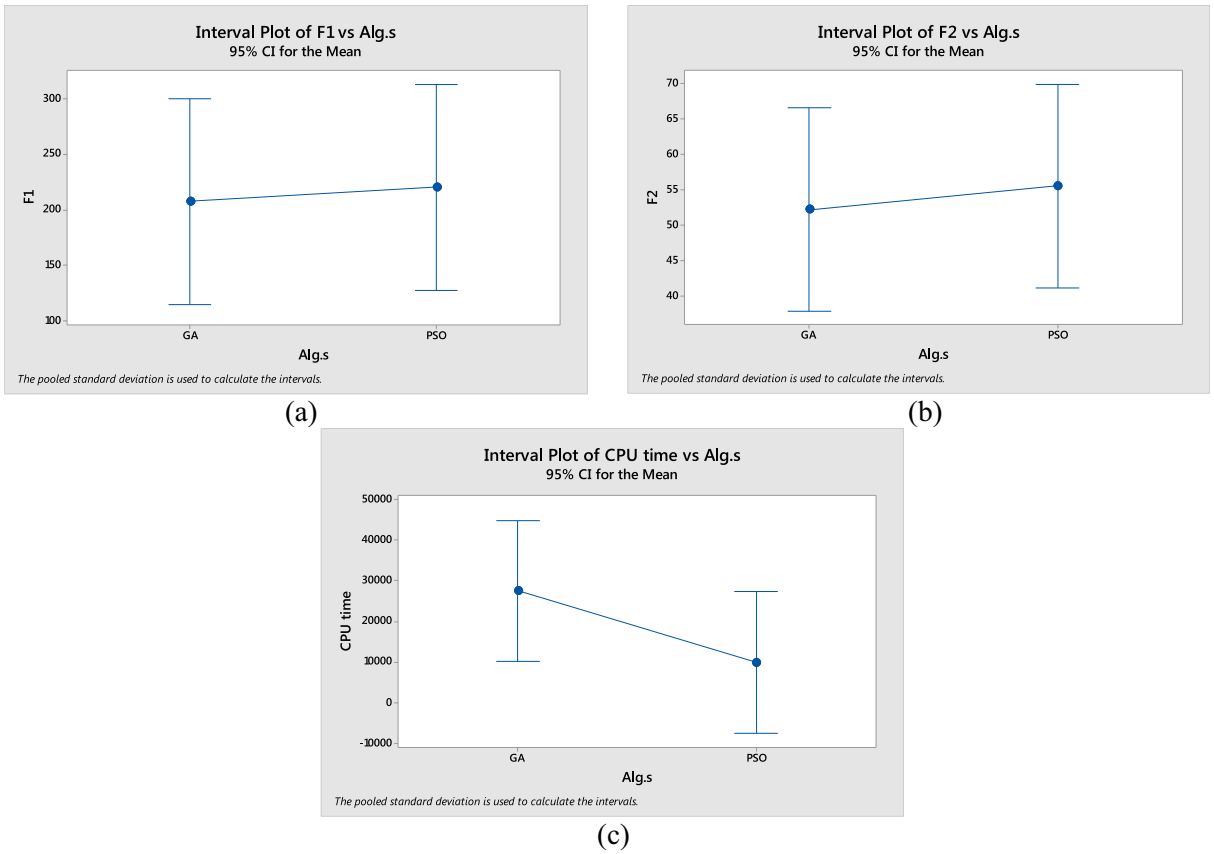


FIGURE 20. The interval plots of F1, F2, and CPU time vs two algorithms.

TABLE 14. Parameters settings.

| | | |
|---|---|---|
| Genetic algorithm | Coss over operator | Single-point |
| | Chromosome length | Equal to the number of machines, parts, and workers |
| | Maximum cell numbers | 4 |
| | Size of population | 25 |
| | Mutation probability | 0.012 |
| | Number of iterations | 2500 |
| | Number of runs | 10 |
| PSO algorithm | $C_1$ | 0.5 |
| | $C_2$ | 0.5 |
| | $W$ | 0.5 |
| | Size of population | 25 |
| | Number of iterations | 2500 |
| | Number of runs | 10 |

TABLE 15. Cases characteristics for validation.

| Problem | Approach | Size (Machine*Part) | # cells |
|---|---|---|---|
| [4] | NSG II | 5*7, 8*20 | 4 |
| [4] | MOPSO | 5*7, 8*20 | 4 |
| This Paper | NSB-GA | 5*7, 8*20 | 4 |
| This Paper | NSB-PSO | 5*7, 8*20 | 4 |

TABLE 16. Results (size (5*7)).

| Problem | Approach | Machines in each cell | | | | Number of intercellular movement | Number of intracellular movement |
|---|---|---|---|---|---|---|---|
| | | I | II | III | IV | | |
| [4] | NSG II | M1,M3 | M2 | M5 | M4 | 7 | 10 |
| [4] | MOPSO | M2,M4,M1 | – | – | M5 | 12 | 19 |
| This Paper | NSB-GA | M2,M5 | M1 | – | M3,M4 | 6 | 9 |
| This Paper | NSB-PSO | M1 | M2,M3 | M4 | M5 | 5 | 7 |

TABLE 17. Results (size (8*20)).

| Problem | Approach | Machines in each cell | | | | Number of intercellular movement | Number of intracellular movement |
|---|---|---|---|---|---|---|---|
| | | I | II | III | IV | | |
| [4] | NSG II | M2,M3 | M4,M6 | M7 | M8 | 6 | 10 |
| [4] | MOPSO | M7,M8,M1 | M2 | M3,M4,M5 | M6 | 9 | 13 |
| This Paper | NSB-GA | M1,M2 | M3 | M5 | M4,M6,M7,M8 | 5 | 9 |
| This Paper | NSB-PSO | M1,M2 | M3,M4 | M5,M6 | M7,M8 | 3 | 4 |

TABLE 18. Total results.

| Problem | Approach | Size (machine* part) | Total number of intercellular movements | Total number of intracellular movements | Maximum cell load variation (%) | Computational time in seconds |
|---|---|---|---|---|---|---|
| [4] | NSG II | 5*7 | 7 | 10 | 11% | 154 |
| [4] | NSG II | 8*20 | 6 | 10 | 10% | 200 |
| [4] | MOPSO | 5*7 | 12 | 19 | 9% | 169 |
| [4] | MOPSO | 8*20 | 9 | 13 | 7% | 170 |
| This Paper | NSB-GA | 5*7 | 6 | 9 | 8% | 160 |
| This Paper | NSB-GA | 8*20 | 5 | 9 | 9% | 180 |
| This Paper | NSB-PSO | 5*7 | 5 | 7 | 5% | 145 |
| This Paper | NSB-PSO | 8*20 | 3 | 4 | 2% | 120 |

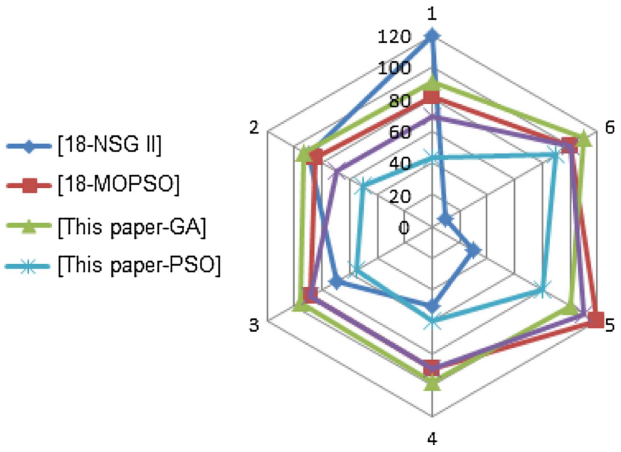


FIGURE 21. Comparison based on computational time.

ufacturers need to produce new and innovative products for their survival in the market. Issues consideration such as manufacturing technology and machinery layout can be a guarantee for improvement of the cost and quality of products. But the human resource has a decisive role in invention and creativity in the final product. The more important issue is staff planning and their interest to cooperate and collaborate with each other. Development and transfer of knowledge among employees will take place when they have the right relationships with each other. This cooperation can bring innovation and creativity in products. Most problems in the cellular manufacturing systems have considered strategic problems such as determining the optimal facility layout with the aim of reducing the voids and exceptional elements, optimal production routing, etc. In this paper, the assignment of human resources is considered to increase the level of cooperation between employees in addition to the reduction of voids and exceptional elements. Increasing the interest of staff is a tactical decision while facility layout is a strategic decision. A decentralized bi-level programming approach has been considered to solve such a problem. At the first level, priority is given to the reduction of voids and exceptional elements. At the second level, the assignment of the workforce in the cells is considered.

Since the proposed model was bi-level programming, thus the nested version of GA and PSO algorithms has been developed to solve the problem. Nine test problems were examined using these algorithms according to the best parameters achieved by Taguchi experiments. The leader level objective, follower level objective, CPU time and ND indices are considered to compare the obtained solutions. Based on the computational results, in small-size problems, NBL-GA has less deviation than NBL-PSO and in the medium to large-size problems, the NBL-PSO method is more efficient than NBL-GA. Also, to compare these metaheuristics more carefully, ANOVA test was utilized with a 95% confidence level. Finally, the mean and standard deviation of each algorithm for the nine tests and all criteria were reported, which the significant difference was observed between the two offered meta-heuristics based on $F_1$, $F_2$, and CPU time. Lastly, since in term of $F_2$ and CPU time, the PSO was better than GA (two criteria among three criteria), thus, the PSO as the best algorithm is reported.

The validation of the proposed method has been compared by different criteria such as computational time, cell load variation, total intercellular movements, and total intracellular movements too. These results have been shown that the usage of the PSO algorithm is better for large-size problems. In the design of this cell, the total cellular movements have been decreased which causes maximization in lead time and workers performance (with a decrease of workers fatigue). Also, maximum cell load variation is in the best value. In this case, an increase in product uncertainty level could be managed in better situations. Thus, the high interest level of workers for working in cellular manufacturing through the design of this cell which has significant effects in improvement of

dynamic cellular manufacturing is the main contribution of this study. To develop the model in the future, other uncertain factors could be included in the first level. Also, this framework can be performed in a real-world case study and several heuristics can be developed to compare with metaheuristics.

## References


[1] A. Aalaei, H. Fazlollahtabar, I. Mahdavi, N. Mahdavi-Amiri and M.H. Yahyanejad, A genetic algorithm for a creativity matrix cubic space clustering: a case study in Mazandaran Gas Company. *Appl. Soft Comput.* **13** (2013) 1661–1673.

[2] Z. Albadawi, H.A. Bashir and M. Chen, A mathematical approach for the formation of manufacturing cells. *Comput. Ind. Eng.* **48** (2005) 3–21.

[3] M. Anvari, M.S. Mehrabad and F. Barzinpour, Machine–part cell formation using a hybrid particle swarm optimization. *Int. J. Adv. Manuf. Technol.* **47** (2010) 745–754.

[4] A. Azadeh, M. Ravanbakhsh, M. Rezaei-Malek, M. Sheikhalishahi and A. Taheri-Moghaddam, Unique NSGA-II and MOPSO algorithms for improved dynamic cellular manufacturing systems considering human factors. *Appl. Math. Model.* **48** (2017) 655–672.

[5] M. Bashiri, M. Rezanezhad, R. Tavakkoli-Moghaddam and H. Hasanzadeh, Mathematical modeling for a p-mobile hub location problem in a dynamic environment by a genetic algorithm. *Appl. Math. Model.* **54** (2018) 151–169.

[6] B. Behnia, I. Mahdavi, B. Shirazi and M.M. Paydar, A bi-level bi-objective mathematical model for cellular manufacturing system applying evolutionary algorithms. *Sci. Iran.* **26** (2019) 2541–2560.

[7] B. Bootaki, I. Mahdavi and M.M. Paydar, New criteria for configuration of cellular manufacturing considering product mix variation. *Comput. Ind. Eng.* **98** (2016) 413–426.

[8] J.R. Brown, A capacity constrained mathematical programming model for cellular manufacturing with exceptional elements. *J. Manuf. Syst.* **37** (2015) 227–232.

[9] A. Cheraghalipour, M. Hajiaghaei-Keshteli and M.M. Paydar, Tree Growth Algorithm (TGA): a novel approach for solving optimization problems. *Eng. Appl. Artif. Intell.* **72** (2018) 393–414.

[10] A. Cheraghalipour, M.M. Paydar and M. Hajiaghaei-Keshteli, A Bi-objective optimization for citrus closed-loop supply chain using Pareto-based algorithms. *Appl. Soft Comput.* **69** (2018) 33–59.

[11] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory. *MHS'95. Proc. Sixth Int. Symp. Micro Mach. Hum. Sci.* (1995) 39–43.

[12] G. Egilmez, B. Erenay and G.A. Süer, Stochastic skill-based manpower allocation in a cellular manufacturing system. *J. Manuf. Syst.* **33** (2014) 578–588.

[13] B. Elbenani, J.A. Ferland and J. Bellemare, Genetic algorithm and large neighbourhood search to solve the cell formation problem. *Expert Syst. Appl.* **39** (2012) 2408–2414.

[14] G.M. Fazakerley, Group technology: social benefits and social problems. *Prod. Eng.* **53** (1974) 383–386.

[15] J.F. Gonçalves and M.G.C. Resende, An evolutionary algorithm for manufacturing cell formation. *Comput. Ind. Eng.* **47** (2004) 247–273.

[16] Z. Guo, D. Zhang, S.Y.S. Leung and L. Shi, A bi-level evolutionary optimization approach for integrated production and transportation scheduling. *Appl. Soft Comput.* **42** (2016) 215–228.

[17] A. Hertz, B. Jaumard, C.C. Ribeiro and W.P. Formosinho Filho, A multi-criteria tabu search approach to cell formation problems in group technology with multiple objectives. *RAIRO: OR* **28** (1994) 303–328.

[18] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press Cambridge, USA (1975).

[19] G. Jeon and H.R. Leep, Forming part families by using genetic algorithm and designing machine cells under demand changes. *Comput. Oper. Res.* **33** (2006) 263–283.

[20] R.J. Kuo, Y.H. Lee, F.E. Zulvia and F.C. Tien, Solving bi-level linear programming problem through hybrid of immune genetic algorithm and particle swarm optimization algorithm. *Appl. Math. Comput.* **266** (2015) 1013–1026.

[21] S. Ma, A nonlinear bi-level programming approach for product portfolio management. *Springerplus* **5** (2016) 727.

[22] Y. Ma, F. Yan, K. Kang and X. Wei, A novel integrated production-distribution planning model with conflict and coordination in a supply chain network. *Knowledge-Based Syst.* **105** (2016) 119–133.

[23] H. Maghsoudlou, M.R. Kahag, S.T.A. Niaki and H. Pourvaziri, Bi-objective optimization of a three-echelon multi-erver supply-chain problem in congested systems: modeling and solution. *Comput. Ind. Eng.* **99** (2016) 41–62.

[24] I. Mahdavi, M.M. Paydar, M. Solimanpur and A. Heidarzade, Genetic algorithm approach for solving a cell formation problem in cellular manufacturing. *Expert Syst. Appl.* **36** (2009) 6598–6604.

[25] M.M. Paydar, I. Mahdavi, S.V. Khonakdari and M. Solimanpur, Developing a mathematical model for cell formation in cellular manufacturing systems. *Int. J. Oper. Res.* **11** (2011) 408.

[26] M. Rabbani, M. Samavati, M.S. Ziaee and H. Rafiei, Reconfigurable dynamic cellular manufacturing system: a new bi-objective mathematical model. *RAIRO: OR* **48** (2014) 75–102.

[27] M. Sakhaii, R. Tavakkoli-Moghaddam, M. Bagheri and B. Vatani, A robust optimization approach for an integrated dynamic cellular manufacturing system and production planning with unreliable machines. *Appl. Math. Model.* **40** (2016) 169–191.

[28] K. Sarrafha, S.H.A. Rahmati, S.T.A. Niaki and A. Zaretalab, A bi-objective integrated procurement, production, and distribution problem of a multi-echelon supply chain network design: a new tuned MOEA. *Comput. Oper. Res.* **54** (2015) 35–51.

[29] A. Sbihi and M. Chemangui, A genetic algorithm for solving the steel continuous casting problem with inter-sequence dependent setups and dedicated machines. *RAIRO: OR* **52** (2018) 1351–1376.

[30] S.M. Shafer, B.J. Tepper, J.R. Meredith and R. Marsh, Comparing the effects of cellular and functional manufacturing on employees' perceptions and attitudes. *J. Oper. Manag.* **12** (1995) 63–74.
[31] G. Taguchi, Introduction to quality engineering? Designing quality into products and processes (1986).
[32] E.G. Talbi, Metaheuristics for Bi-level Optimization. Springer Berlin Heidelberg (2013).
[33] G.G. Udo and A.A. Ebiefung, Human factors affecting the success of advanced manufacturing systems. *Comput. Ind. Eng.* **37** (1999) 297–300.