RAIRO-Oper. Res. 48 (2014) 75–102
DOI: 10.1051/ro/2013054

RAIRO Operations Research
www.rairo-ro.org

RECONFIGURABLE DYNAMIC CELLULAR MANUFACTURING SYSTEM: A NEW BI-OBJECTIVE MATHEMATICAL MODEL

Masoud Rabbani[1], Mehran Samavati[1], Mohammad Sadegh Ziaee[2] and Hamed Rafiei[1]

Abstract. Dynamic Cell Formation Problem (DCFP) seeks to cope with variation in part mix and demands using machine relocation, replication, and removing; whilst from practical point of view it is too hard to move machines between cells or invest on machine replication. To cope with this deficiency, this paper addresses Reconfigurable Dynamic Cell Formation Problem (RDCFP) in which machine modification is conducted instead of their relocation or replication in order to enhance machine capabilities to process wider range of production tasks. In this regard, a mixed integer nonlinear mathematical model is proposed, which is NP-hard. To cope with the proposed model's intractability, an Imperialist Competitive Algorithm (ICA) is developed, whose obtained results are compared with those of Genetic Algorithm's (GA's), showing superiority and outperformance of the developed ICA.

Keywords. Dynamic cell formation problem, genetic algorithm, imperialist competitive algorithm, machine modification, reconfigurable cellular manufacturing system.

Mathematics Subject Classification. 90B99.

Received December 5, 2012. Accepted September 30, 2013.

[1] School of Industrial & Systems Engineering, College of Engineering, University of Tehran, North Kargar St., P.O. Box: 11155-4563, Tehran, Iran. mrabani@ut.ac.ir; mehran_samavati@ut.ac.ir; hrafiei@ut.ac.ir

[2] Faculty of Management, University of Tehran, Tehran, Iran. ziaee@ut.ac.ir

Article published by EDP Sciences

1. Introduction

Today, sophisticated competitive business environment has caused manufacturers to control their relevant production costs in any involved aspects of process. Among these aspects, facilities might be one of the fields requiring high level of investment as estimated over $250 billion annually in the United States [1]. In this regard, Group Technology (GT) is adopted by most of the companies to tackle different manufacturing issues. GT is defined as the principle of grouping parts upon their similarities in design and production process [2]. One of the concepts introduced in GT is Cellular Manufacturing System (CMS) which attempts to form manufacturing cells by grouping machines into physical cells which are able to process specific families of parts. To do so, CMS takes advantages of both flow shops and job shops with lower level of cycle time than job shops and higher level of flexibility and job satisfaction than flow shops [2]. More recently, shorter product life cycles and new product introduction have had direct influences on product mix and demand volumes, resulting in reconfiguring the cells upon the changes. To cope with this issue, Dynamic Cellular Manufacturing System (DCMS) was firstly introduced by Rheault *et al.* [3]. In DCMS, machine grouping is updated with respect to the changes of product mix and demands. To do so, one might swap existing machines between cells (machine relocation), or add new machines (machine replication), or remove existing machines [4]. Figure 1 shows a schematic view of a DCMS including three cells in two consecutive time periods. As shown in this figure, Machines 3, 4, and 5 are relocated from some cells to some others in Period 2.

Although cell changes (machine relocation, removing, and replication) are utilized in order to cope with the changes of demand mix and volumes, it seems impossible or too costly to relocate heavy machines between cells. Also, machine replication imposes purchasing costs of machines which are mostly expensive. With respect to the skilled workers, workers are also relocated with machine relocation, requiring costs of training and setup for the new tasks assigned to the workers. On the other hand, if one decides to equip all cells with all types of machines at the beginning of the planning horizon, high level of investment and property taxes are suffered. Hence, it is inevitable to conduct some sorts of actions to both cope with market changes and reduce capacity costs of system. In this regard, this paper addresses a RDCFP for the first time. To do so, concept of machine modification is adopted, which was firstly introduced by Foulds *et al.* [5] in the case of static Cell Formation Problem (CFP). Foulds and his colleagues called such a CMS as a sustainable CMS. Since sustainability is mostly dedicated to environmental issues, the authors decided to call the considered system as a reconfigurable dynamic cellular manufacturing system, because reconfigurable manufacturing system is defined as a system capable to rapidly rearrange/replace/modify its structure/software/hardware in order to respond to sudden changes in market, technology or regulatory requirements [6, 7].

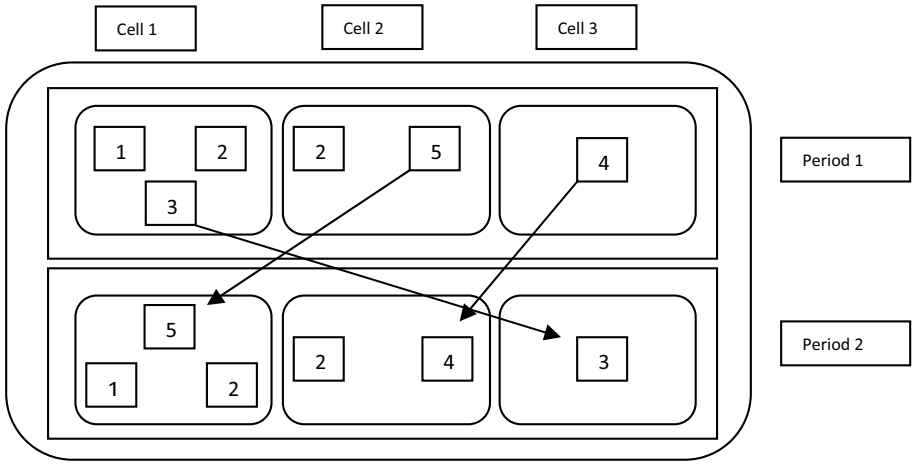


FIGURE 1. Schematic view of a three-cell DCMS in two consecutive periods.

In the considered problem, it is revealed whether machine modification or machine replication is cost effective; whilst it is not possible to add new machines to the cells whose capacity is totally utilized. Some instances of such systems are reported in [5, 8]. Grahl [8] reported a case study in ceramic industry in which machine modification could have improved firm productivity of %15; on the other hand, some manufacturing firms used machine modification instead of machine replication to perform wider range of production processes, such as Skoda (http://www.skoda.cz) [5]. To tackle the considered RDCFP, a bi-objective mixed-integer nonlinear mathematical model is proposed. In order to cope with the multi-objective structure of the proposed model, Goal Programming (GP) approach is adopted. Additionally and with respect to NP-hardness of CFP, large-size instances of the proposed model are intractable upon which an ICA is developed. ICA has been deployed in some problems so far, such as flow shop scheduling and hub covering location; while no cases are reported in the field of CMS and CFP (relevant literature review is presented in Sect. 2). Moreover, the adopted algorithm is developed to cope with discrete structure of the proposed model, since the classic ICA is suitable for the continuous problem instances. Finally, results of the developed algorithm are compared with those of GA to validate the developed algorithm. Remainder of the paper is as follows. Section 2 reviews literature body of DCFP. Section 3 elaborates objective functions of the proposed mathematical model as well as the GP approach adopted to tackle the mathematical model. An example of the considered RDCFP is described in Section 4. Sections 5 and 6 explain the improved ICA and the utilized GA, respectively, in order to validate obtained results of the developed algorithm. Section 7 reports numerical experiments conducted to validate solution quality of the improved ICA and Section 8 provides some notable conclusions and future research directions.

2. Literature review

CFP literature body might be reviewed from different points of view. For instance, mathematical programming has been used even in remote literature for modeling the CMS problems [9–12]; or, Song and Hitomi [13] developed a method by considering production planning and cellular layout for a long-run planning horizon, where flexible manufacturing cells are designed.

Moreover, previous investigations have been done in the content of modelling and methodology of the CFP in dynamic environment. For example, Safaei *et al.* [14] proposed a mixed-integer programming model of CMS under dynamic condition. The advantages of the proposed model are to consider inter/intra-cell material handling in specific batches by assuming the sequence of operations, considering alternative process routes for part types, and considering replication of machines. Their aim is to minimize the sum of the inter/intra-cell movements, reconfiguration costs, constant costs, and variable costs of the machines. Another mixed-integer programming model was developed by Ghotboddini *et al.* [15]. Their mathematical model minimizes costs of over time and human resource, and also maximizes utilization rate of human resource. Also, Wang *et al.* [16] presented a nonlinear multi-objective programming model for DCFP considering three conflicting objectives: minimizing machine relocation costs, minimizing machine utilization, and minimizing total number of inter-cell movements. Bajestani *et al.* [17] presented a multi-objective program of DCMS to deal with two objectives including minimizing total cell load variation, and minimizing miscellaneous costs (inter-cell material handling costs, machine costs, and machine relocation costs). Recently, some researches have been conducted in dynamic environment such as Saxena and Jain [4] and Mahdavi *et al.* [18], which presented a mixed-integer nonlinear programming model to design a CMS. Saxena and Jain [4] integrated different production aspects among which machines' breakdown effect and part inventory holding are the primary ones differentiating their research from the others. The aim of the proposed model by Mahdavi *et al.* [18] is to minimize hiring, firing and salary costs, holding and backorder costs, inter-cell material handling costs, and machine and reconfiguration costs.

Satoglu and Suresh [19] proposed a multi-objective mathematical programming model to design hybrid CMSs in dual resource constrained environments for real world problems. They used GP approach to deal with the conflicting objective functions. Among the considered assumptions of DMS, some assumptions, specifically, machine purchasing costs and machine relocation costs have not been taken into account in some papers such as [14, 20–26].

Due to the fact that CFP is a NP-hard problem, solving the relevant model using classical optimization approaches needs a long computational time. There are many researches discussing solving CFP using some classical metaheuristics, such as GA, Simulated Annealing (SA), Neural Networks (NNs), Tabu Search (TS), and Scatter Search (SS). To solve their DCMS models, Wang *et al.* [16] developed an SS approach and checked the efficiency of proposed SS by comparing it with GA.

Bajestani *et al.* [17] also designed a multi-objective scatter search (MOSS) for finding Pareto-optimal frontier and compared the developed MOSS with two salient multi-objective GAs (SPEA-II and NSGA-II) based on some comparison metrics to show the efficiency of their proposed algorithm. Tavakkoli–Moghaddam *et al.* [21] discussed solving DCFP using three metaheuristics GA, SA and TS. They concluded that the SA algorithm mainly found better solutions than GA and TS in less computational times. Arkat *et al.* [27] presented an integrated methodology based on a new concept of similarity coefficient to design a CMS. They used SA method to solve the problem, and compared the proposed SA with GA. Aljaber *et al.* [28] modelled the CFP based on graph theory, and proposed a TS to cope with the problem complexity. Spiliopoulos and Sofianopoulou [29], Logendran and Karim [30], Wu *et al.* [31], Lei and Wu [32] have also applied TS algorithm to the field of CFP. Additionally, NN is another solution method which has been applied in numerous instances due to its robust and adaptive nature. Papers [22, 33–35] are some examples of recent researches addressing CFP using NNs.

GAs have been widely used in the literature of the CFP, among which the first instance backs to the one applied by Venugopal and Narendran [36]. Pierreval *et al.* [37] reviewed the application of GA in CMS. Defersha and Chen [38] proposed a mathematical programming model with the objective function of minimizing several different kinds of costs. Their model involved many different elements such as alternative routings, sequence of operations, workload balancing and machine separation requirements. For large-size problems, the authors developed an efficient heuristic method based on GA. They evaluated the proposed heuristic using some sort of computational results comparing optimal solutions of the heuristic with the optimal solutions of small and medium sized problems. They also obtained the optimal solution of a large-size problem under certain assumptions. Solimanpur *et al.* [35], Wu *et al.* [39], Yasuda *et al.* [40] are other research samples using GA for CMS.

In this paper, an ICA is adopted in order to tackle problem complexity. ICA is an evolutionary algorithm firstly introduced by Atashpas–Gargari and Lucas [41]. This new method has been applied to a number of different problems such as flow shop scheduling problem and hub covering location problem. For solving various NP-hard problems, there are several researches indicating the superiority of ICA over other metaheuristics including SA, GA, and particle swarm optimization (PSO). Despite this superiority, ICA has not been considered significantly in the literature of CMS. The only research has been conducted by Sarayloo and Tavakkoli–Moghaddam [42] in which the authors considered DCFP with production planning where the objective function was to minimize constant machine costs, variable machine costs, production planning costs, reconfiguration costs, and intra/inter-cell movement costs. To expand the application of this algorithm in CFP, we adopt ICA to cope with intractability of the proposed mathematical model.

3. Proposed model

In this section, a bi-objective mixed-integer programming model is formulated for the proposed RDCFP under the following assumptions.

3.1. Assumptions

(1) All the machines are purchased at the beginning of the first period.
(2) All the machines can be modified, and each modified machine can be changed to its primary situation.
(3) Capacity of each machine type is calculated upon its time capacity.
(4) Machine Modification and returning to its primary situation are not performed in the same period.
(5) Purchased machines are allocated to cells in the first period, and subsequent periods do not involve adding, removing, or relocating machines.
(6) Each part type has a specific number of operations.
(7) The processing time of all operations on different machine types are known and deterministic.
(8) Demands are known and deterministic.
(9) Parts are moved between cells as batch.
(10) A lower bound is not considered for the cell size, because smaller cells are more suitable. Also, maximal cell size is known in advance.
(11) All machine types are single purpose.
(12) All demands must be satisfied in the given period, that is, backorders are not allowed.
(13) Setup times are not considered.
(14) No queuing in production is considered.
(15) Machine modification cost is less than their purchasing cost.
(16) Batch size for every part type is constant.
(17) The maximum number of cells that can be formed in each period is specified in advance.
(18) A modified machine cannot perform as its primary state.
(19) Machines cannot be modified more than once in each period.
(20) If a particular machine needs to be used as two distinct machine types in different periods, it is changed to its primary situation before the second modification. For example, assume that a particular machine of type m, in cell c, has been modified in period $t-1$ to be used as machine type h, and we also need to modify this machine in period t to perform as machine type d, so, in period t, we first have to change the machine to its primary situation, which is included a cost, and then modify that to perform as machine type d.

3.2. Notation

Indexes

c index for manufacturing cells ($c = 1; \ldots; C$)
m index for machine types ($m = 1; \ldots; M$)
p index for part types ($p = 1; \ldots; P$)
t index for time periods ($t = 1; \ldots; T$)
j index for operations of part p ($j = 1; \ldots; O_p$)
h index for modified machine types ($h = 1; \ldots; M$)
k index for the number of each machine type in each cell ($k = 1; \ldots; Nkc$)

Input parameters

P number of part types
O_p number of operations for part p
M maximum number of machine types
C maximum possible number of cells
D_{pt} demand for part p in period t
γ inter-cell movement cost per batch
T_m maximal time-capacity of machine type m
S maximal cell size
t_{jpm} processing time required to perform operation j of part type p on machine type m
a_{jpm} 1 if operation j of part p can be done on machine type m; 0 otherwise
G modification cost
U cost of changing each machine to its primary situation
p_m purchasing price of machine type m

Decision variables

N_{mc} number of machine type m allocated to cell c
X_{jpmct} 1 if operation j of part type p is done on machine type m in cell c in period t; 0 otherwise
z_{mkhct} 1 if kth machine of type m in cell c, is modified to perform as machine type h in period t; 0 otherwise
l_{mkct} 1 if kth machine of type m, in cell c, is changed to its primary situation in period t; 0 otherwise
y_{jpct} 1 if operation j of part type p is done in cell c in period t; 0 otherwise
C_{mct} Total capacity of set of machines of type m in cell c in period t (total capacity of set of machines of type m in cell c in period t equals the sum of the capacity of each of the machines minus the amount of time that these machines are used as the other machine types in cell c in period t, and plus the amount of time that the other machine types of cell c perform as machine type m in the same period).
B batch size

b_{mkhct} 1 if kth machine of type m which has been modified to perform as machine type h in period $t-1$ is reused as machine type h in period t; 0 otherwise

t_{mkhct} the amount of time that kth machine of type m is used as machine type h in cell c in period t

t^{+}_{mkhct} the amount of extra time that kth machine of type m, which has been modified in the previous period, perform as machine type h in period t, in comparison with the previous period. For instance, if a particular machine of type m is used as a machine of type h for 3 h in a particular period, and is used again as machine type h for 5 h in the next period, t^{+}_{mkhct} will equal 2 h

t^{-}_{mkhct} the amount of time that kth machine of type m performs for lesser time as machine h in period t, in comparison with the previous period

A_{mkhct} Last uninterrupted using time of kth machine of type m, in cell c as machine type h in periods $1, 2, \ldots, t-1$.

3.3. Mathematical model

The multiple conflicting objectives are as follows: 1) minimizing the total sum of miscellaneous costs such as constant costs, reconfiguration costs, inter-cell movement costs, and machine modification costs; and 2) Maximizing utilization rate of machines.

3.3.1. Minimizing the total sum of miscellaneous costs

$$\text{Min } 1/2\sum_{t=1}^{T}\sum_{p=1}^{P}\left\lceil\frac{D_{pt}}{B}\right\rceil\sum_{j=1}^{o_p-1}\sum_{c=1}^{C}\gamma|y_{(j+1)pct}-y_{jpct}|+\sum_{h=1}^{M}\sum_{m=1}^{M}\sum_{c=1}^{C}\sum_{t=1}^{T}\sum_{k}^{N_{kc}}gz_{hkmct}$$
$$+\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{c=1}^{C}\sum_{k=1}^{N_{kc}}Ul_{mkct}+\sum_{m=1}^{M}\sum_{c=1}^{C}p_mN_{mc} \quad (3.1)$$

$$C_{mc(t-1)}-\sum_{k=1}^{N_{mc}}\sum_{h\neq m}^{M}t_{mkhct}z_{mkhct}+\sum_{r\neq m}^{M}\sum_{k=1}^{N_{rc}}z_{rkmct}t_{rkmct}$$
$$+\left(\sum_{k=1}^{N_{mc}}\sum_{h\neq m}^{M}t_{mkhct-1}z_{mkhct-1}\right)l_{mkct}-\sum_{k=1}^{N_{rc}}\sum_{r\neq m}^{M}t_{rkmct-1}z_{rkmct-1}l_{rkct}$$
$$-\sum_{k=1}^{N_{mc}}\sum_{h\neq m}^{M}t^{+}_{mkhct}b_{mkhct}+\sum_{r\neq m}^{M}\sum_{k}^{N_{rc}}t^{+}_{rkmct}b_{rkhct}+\sum_{k=1}^{N_{mc}}\sum_{h\neq m}^{M}t^{-}_{mkhct}b_{mkhct}$$
$$-\sum_{r\neq m}^{M}\sum_{k}^{N_{rc}}t^{-}_{rkmct}b_{rkhct}=C_{mct} \qquad \forall t\geqslant 2, m, c \quad (3.2)$$

$$\sum_{h\neq m} z_{mkhct} \leqslant 1 - \left(\sum_{\alpha=1}^{t-1} \sum_{r\neq m}^{M} z_{mkrc\alpha} - \sum_{\beta=1}^{t} l_{mkc\beta} \right) \qquad \forall m, c, k, t \geqslant 2 \tag{3.3}$$

$$l_{mkct} \leqslant \sum_{\alpha=1}^{t-1} \sum_{h\neq m} z_{mkhc\alpha} - \sum_{\beta=2}^{t-1} l_{mkc\beta} \qquad \forall t \geqslant 2, m, c, t \tag{3.4}$$

$$l_{mkc1} = 0 \quad \forall m, c, k \tag{3.5}$$

$$C_{mc1} = T_m N_{mc} - \sum_{k}^{N_{mc}} \sum_{h\neq m} t_{mkhc1} z_{mkhc1} + \sum_{k}^{N_{rc}} \sum_{h\neq m} t_{rkmc1} z_{rkmc1} \quad \forall m, c \tag{3.6}$$

$$t_{mkhct} \leqslant T_m \qquad \forall m, c, k, h, t \tag{3.7}$$

$$\sum_{p} \sum_{j} D_{pt} D_{jpm} x_{jpmct} \leqslant C_{mct} \qquad \forall m, c, t \tag{3.8}$$

$$\sum_{c=1}^{C} \sum_{m} \alpha_{jpm} x_{jpmct} = 1 \qquad \forall j, p, t \tag{3.9}$$

$$\sum_{m} N_{mc} \leqslant s \qquad \forall c \tag{3.10}$$

$$\sum_{m} z_{hkmct} \leqslant 1 \qquad \forall c, h, t, k \tag{3.11}$$

$$b_{mkhct} \leqslant \sum_{t=1}^{\beta-1} \left(z_{mkhct} \times \max \left(0, \left(1 - \sum_{\alpha=t+1}^{\beta} l_{mc\alpha} \right) \right) \right) \quad \forall \beta = 2, 3, \ldots, T \tag{3.12}$$

$$A_{mkhct-1}(1 - l_{mkct}) + t^{+}_{mkhct-1} b_{mkhct-1} - t^{-}_{mkhct-1} b_{mkhct-1} = A_{mkhct} \qquad \forall m, c, k, h, t \tag{3.13}$$

$$t^{+}_{mkhct} \leqslant T_m - A_{mkhct} \qquad \forall m, c, k, h, t \tag{3.14}$$

$$t^{-}_{mkhct} \leqslant A_{mkhct} \qquad \forall m, c, k, h, t \tag{3.15}$$

$$\sum_{m\neq r}^{M} z_{rkmct} \leqslant 1 - b_{rkhct} \qquad \forall c, k, h, t, r = 1, \ldots, M \tag{3.16}$$

$$A_{mkhc1} = 0, A_{mkhc2} = t_{mkhc1} z_{mkhc1}, t^{+}_{mkhc1} = 0, t^{-}_{mkhc1} = 0 \quad \forall m, c, k, h \tag{3.17}$$

$$N_{mc}, C_{mct}, t_{hkmct}, A_{mkhct} \geqslant 0, l_{mkct}, x_{jpmct}, z_{hkmct}, b_{mkhct}, y_{jpct} \in \{0, 1\} \qquad \forall m, k, h, c, t. \tag{3.18}$$

The objective function given in equation (3.1) seeks to minimize the total sum of the miscellaneous costs. The first term is related to the inter-cell part handling costs. The second term represents the machine modification costs. The third term represents the total cost of changing the machines to their primary situations. The last term represents the purchasing cost of all machines, which are required throughout the planning horizon. Constraint (3.2) is called balance constraint that plays the role of the memory for available capacity in each period. The second term of this constraint deducts the amount of time that machine type m will perform

as the other machine types in period t from the capacity of the previous period if machine type m has not been modified in period $t-1$. The third term adds the amount of time that some of the other machine types will be used as machine type m in period t to the capacity of the previous period if these machines have not been used as machine type m in period $t-1$. Also, if machine type m has been used as the other machine types in period $t-1$, and is changed to its primary situation in period t, the available capacity of machine type m will be increased in this period. The forth term of constraint (3.2) is related to this fact. The fifth term deducts the amount of time that some of the other machine types have been used as machine type m in period $t-1$ from the capacity of this period provided that these machines do not perform as machine type m in period t. If a particular machine needs to perform as another machine for two successive periods, and also the amount of using time in the second period is more, the extra time will be added to the first period using time in the form of t^+. Therefore, the sixth term deducts the amount of extra time that machine type m will be used as another machine type from the available capacity of period $t-1$. The seventh term is related to the extra using time during which the other machine types perform as machine type m in period t Moreover, if a particular machine needs to perform as another machine for two successive periods, and the amount of using time in the second period is less, the difference will be deducted from the first period using time in the form of t^-. Therefore, the eighth term adds this difference to the available capacity of period $t-1$. In other words, machine type m will perform as a certain machine in period t for less time compared to period $t-1$ and therefore, the capacity will increase. Similarly, the ninth term ensures that the capacity will decrease in period t. Constraint (3.3) is related to the assumption (20). Constraint (3.4) ensures that every machine can be changed to its primary situation only when it is modified. Constraint (3.5) guarantees that the machines cannot be changed to their primary situations in the first period. Constraint (3.6) specifies the available capacity of the first period. Equation (3.7) states that the machines cannot perform as the other machines more than their maximal capacities. Constraint (3.8) guarantees that machine capacity is not exceeded. Equation (3.9) ensures that each operation is assigned to one type of machine and to one cell. Constraint (3.10) specifies the upper bound for cell size. Equation (3.11) is related to the assumption (19). Decision variable b_{mkhct} is applied when the kth machine of type m needs to be used as machine type h for two or more successive periods. Thus, constraint (3.12) ensures that b_{mkhct} can be 1 only if this machine is already ready to perform as machine type h, that is, it does not need to be modified at the beginning of period t. Each machine can be modified and used as a particular machine for several successive periods. Thus, variable A_{mkhct} is to indicate the last uninterrupted using time of machine m as machine h by the beginning of period t. In other words, if machine m is modified into machine h in period 2, for example, and reused in periods 3 to 6 A_{mkhc7} will be equal to the total using time of periods 2, 3, 4, 5 and 6. However, if this machine is changed to its primary situation in period 4 and is not modified to machine h again, A_{mkhc7} will be equal to zero. Therefore, equation (3.13) is to

ensure this balance for all machines. Variable A_{mkhct} is used to obtain an upper bound for variables t^{+}_{mkhct} and t^{-}_{mkhct} in constraints (3.14) and (3.15) respectively. Constraint (16) ensures that if a machine needs to be reused as a particular machine type in a particular period, it cannot be modified into another machine type in the same period. Constraint (3.17) represents some of the variable values in the first period. Equation (3.18) is the nonnegativity and integrality constraint.

3.3.2. Maximizing utilization rate of machines

The objective function given in equation (3.19) minimizes the maximum deviation between the workload assigned to each type of machine and its available capacity in order to increase machine utilization. In the case that modification cost exceeds that of machine purchasing, the last term of objective function (3.1) seeks to buy as many as required machines of each type at the beginning of the first period. This results in a high amount of investment as well as some issues, such as extra property tax and machine idle time. To cope with this deficiency, objective function (3.19) is proposed so as to maximize overall utilization rate of machines. Having the proposed objective function adopted, the idle time of machines are challenged to be minimized. Hence, the two proposed objective functions (3.1) and (3.19) are conflicting in nature.

$$\operatorname{Min}\max_{m}\left|\sum_{c} T_m N_{mc} - \sum_{t}\sum_{c}\sum_{p}\sum_{j} D_{pt} t_{jpm} x_{jpmct} - \sum_{t}\sum_{c}\sum_{h\neq m}\sum_{k}^{N_{mc}} A_{mkhct}\right|. \tag{3.19}$$

3.4. Goal programming

To find the optimal solution of multi-objective problems, several methods have been introduced, such as weighted metric, weighted sum, ε-constraint, interactive approaches, and GP. In this paper, we use GP method to combine multiple objectives into a single objective. To do so, assume that Z_1 and Z_2 are the first and the second objective functions in the proposed model. Hence, the single objective and the added constraints are as follows:

$$S = \min \sum \delta_i$$

$$\delta_1 = 1 - \frac{f_1}{z_1}$$

$$\delta_2 = 1 - \frac{(f_2 + \varepsilon)}{z_2}$$

where f_i is the ith objective function's optimal value. In other words, for each objective function, the model run individually, resulting in values of f_1 and f_2. Unlike the first objective function, it is possible for f_2 to obtain the value of zero. If so, in the second constraint, we will have $\delta_2 = 1$. As a result, δ_2 will be a constant variable and does not change when variable Z_2 varies. Hence, parameter ε is added

TABLE 1. Test problem generation.

Parameter	Value	Parameter	Value	Parameter	Value
T_m	$U(10,50)$	C	Equal to M	p_m	$U(100,200)$
S	$\lceil\sqrt{p}\rceil+1$	Op	Regard to problem size	γ	$\cong 2/3\frac{\sum p_m}{M}$
T	Regard to problem size	D_{pt}	$U(1,10)$	g	$\cong 1/5\frac{\sum p_m}{M}$
P	Regard to problem size	t_{jpm}	$U(1,10)$	U	$1/2g$
M	$[p/2]+2$	$\sum_m a_{jpm}$	$1\ \forall j,p$		

to prevent this situation If $f_2 > 0$, parameter ε will be assumed to be equal to zero, and also if $f_2 = 0$, ε will be considered as a number with small value. S, $(0 \leqslant S < 2)$, represents the value of the obtained single objective function.

4. A NUMERICAL EXAMPLE

To verify the applicability of the proposed model, a hypothetical example with randomly generated data is presumed. This example is solved by GAMS 22.1\ DICOPT. The data are generated according to Table 1.

In this table, $\sum_m a_{jpm}$ shows the number of the alternative machines for processing operation j of part p, and it is assumed that $\sum_m a_{jpm} \leqslant 2$. Term U stands for uniform distribution. Maximal cell size (S) and the number of machine types (M) are obtained according to $\lceil\sqrt{p}\rceil+1$ and $\lceil p/2\rceil+2$ respectively [14]. Modification cost (g) is assumed to be less than both inter-cell movement cost (γ) and the purchasing price of the machines. The inter-cell movement cost is also assumed to be less than the purchasing price of the machines. It is noted that values are rounded in the case of sign $\cong$. Also, it is assumed that $\varepsilon = 0$ if $f_2 > 0$, and $\varepsilon = 0.001$ if $f_2 = 0$. When $f_2 = 0$, we only need to consider ε as a number which is smaller than 1 as it has been assumed that $1 \leqslant t_{jpm} \leqslant 10$ and therefore the value of variable Z_2 will not be smaller than 1.

Table 2 consists of the data set related to the considered example. In this example, we have assumed three machine types, two part types, and two periods. During each period, every part has two operations being processed. The programming model has been solved three times using GAMS 22.1\ DICOPT. Initially, the first objective function and constrains (3.1) to (3.18) have been run, and the optimal value of the objective function has been considered as f_1. Subsequently, the same constraints have been run with the second objective function, and the optimal value has been considered as f_2. Finally, constraints (3.1) to (3.18) and the constraints proposed in Section 3.3 have been run with the single objective function which was obtained according to Section 3.3.

The best solution for this example was found after twelve minutes. The solution is illustrated in Figure 2. According to this figure, in order to satisfy the first period

TABLE 2. Typical test problem.

	Part 1		Part 2		Cost	Capacity of machines
	Op. 1	Op. 2	Op. 1	Op. 2		
Machine 1	6 min				$P1 = 100\$$	$T1 = 50$ min
Machine 2	6 min	8 min		10 min	$P2 = 120\$$	$T2 = 25$ min
Machine 3			5 min		$P3 = 110\$$	$T3 = 35$ min
Demand of period 1	5		2		$\gamma = 80\$$	
Demand of period 2	4		3		$g = 20\$$	
					$U = 10\$$	

demands, both operations of part 2 are processed on machine 3. Also, the second operation of three parts of type 1 is processed on machine 2. Machine 1, initially, performs the first operation of all the parts of type 1, and then is modified to be used as machine 2 to process the second operation of the two remaining parts of type 1. In period 2, both operations of two parts of type 2 are processed on machine 3, and the operations of the remaining parts of type 2 are processed on machine 2. This machine performs the second operation of one part of type 1 as well. Since machine 1 has been modified in period 1, we can reuse machine 1 as machine 2, without modification cost, to process both operations of part 1. In period 1, the amount of time during which machine 1 is used as machine 2 is 16 min, rising by 8 min in period 2. This difference is indicated in the form of $t^{+}_{11212} = 8$. The optimal value obtained for this example is $Z_1 = 390\$$.

5. Improved imperialist competitive algorithm

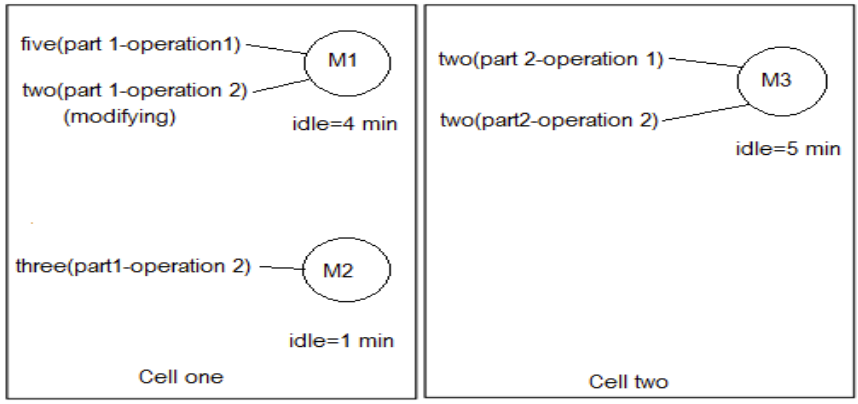
The Imperialist Competitive Algorithm (ICA) is an evolutionary algorithm introduced by Atashpaz and Lucas [41]. This algorithm uses socio-political evolution of human as a source of inspiration to develop a strong optimization method. Since the basic ICA is only suitable for problems with continuous variables, we improve the algorithm in order to be proper for discrete problems. Moreover, crossover and mutation operators of the genetic algorithm are applied to ICA. Following, we explain the steps of the proposed ICA, while its pseudocode is presented in Figure 3.

5.1. Solution coding (Country structure)

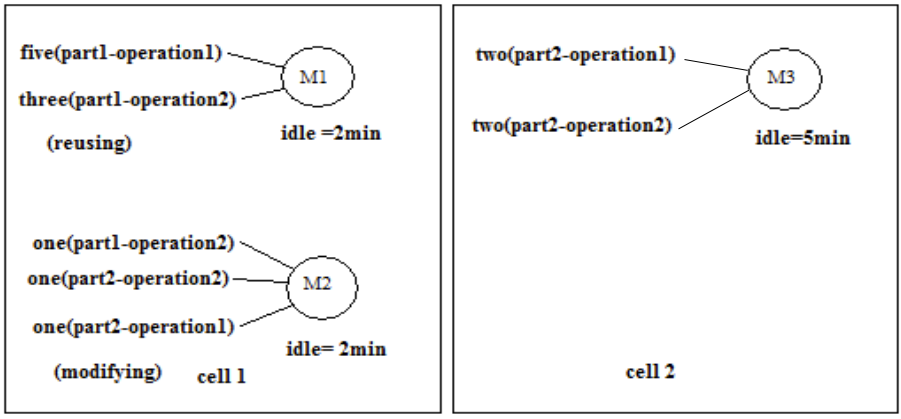
A country- or a feasible solution- developed for the described model has a macroscopic structure as follows:

$$\left[[N]_{m\times c} \,|\, [t]_{m_k\times h\times t} \,|\, \left[t^{+}\right]_{m_k\times h\times t} \,|\, [x]_{p\times j\times t}\right].$$

Matrix $[N]_{m\times c}$ is related to the assignment of machines to cells. The members of this matrix are limited to 0, 1, 2, ..., S (maximal cell size). Term $N_{12} = 3$, for example, means that there are three machines of type 1 assigned to cell 2. Matrix



period 1



period 2

FIGURE 2. Best obtained cell configurations for typical test problem presented in Table 2.

$t_{m_k \times h \times t}$ is a three-dimensional matrix associated with machine modification. We assume that matrix $t_{m_k \times h \times t}$ is encoded for every couple (c, m), and $[t]_{m_k \times h \times t}$ indicates the set of these three-dimensional matrices. The members of these matrices have a structure of $t_{kht}^{(m)(c)}$, and their values are between $-Tm$ and $+Tm$ where Tm is the time capacity of machine type m. Also, m_k is the number of machine type m in each cell, t indicates the period and h is a notation for the type of machine. For instance, $t_{235}^{(1)(4)} = 6$ means that the second machine of type 1 in cell 4 is modified to be used for six minutes as machine type 3 in period 5. It is obvious that $[t]_{m_k \times h \times t}$ consists of $C \times M$ matrices $t_{m_k \times h \times t}$ at most. In other words, matrix $t_{m_k \times h \times t}$ is encoded for couple (c, m) if machine type m is placed in cell c. It is noted that the existence of machine m in cell c can be recognized by matrix $[N]_{m \times c}$. Figure 4

1. *Create the initial population randomly.*
2. *Select* N_{imp} *countries from the best ones as the imperialist, and construct the empires.*
3. *Divide the rest of the countries among the empires based on their normalized power.*
4. *Select any unselected colony.*
   - 4.1. *Apply the Crossover operator to the colony and its relevant imperialist.*
   - 4.2. *Name the best produced countries 'the best offspring'.*
   - 4.3. *Apply the Mutation operator to the colony and name the result 'Mutated offspring'.*
   - 4.4. *Select the best solution among Mutated offspring, best offspring, and related colony.*
   - 4.5. *If the selected solution is the colony, eliminate that; otherwise, replace the selected solution as the colony.*
   - 4.6. *If there is unselected colony, go to step 4.*
5. *If there is a colony in an empire which has lower cost than it's imperialist, exchange the position of that colony and the imperialist.*
6. *Imperialistic competition*
   - 6.1. *Find the weakest empire based on the total normalized power.*
   - 6.2. *Separate the worst country of the selected empire and assign it to the empire which has the most likelihood to possess it.*
7. *If there is an empire with no colonies, eliminate that.*
8. *If more than one empire remain, go to step 4; otherwise, go to step 9.*
9. *Display the only imperialist as the optimal solution.*

FIGURE 3. Pseudo-code of the developed ICA.

illustrates the matrix structure of $t_{m_k \times h \times t}$ for each couple (c, m). Similarly, matrix $[t^+]_{m_k \times h \times t}$ consists of three-dimensional matrices $t^+_{m_k \times h \times t}$ which are encoded for every couple (c, m). This matrix is associated with variables t^-_{mkhct} and t^+_{mkhct}, and its members have a structure of $t^{+(m)(c)}_{kht}$. For example, $t^{+(4)(1)}_{235} = -6$ means that the second machine of type 4 in cell 1 is used for 6 min less than the amount of time for which it has performed as machine 3 in the previous period. Matrix $[X]_{p \times j \times t}$ consists of c matrices $X_{p \times j \times t}$ indicating the assignment of the parts to the machines, and its members are defined as $x^{(C)}_{pjt} x^{(2)}_{143} = 5$, for instance, means that the forth operation of part type 1 is processed on machine type 5 in cell 2. Obviously, the number of three-dimensional matrices in $[X]_{p \times j \times t}$ equals the number of cells.

5.2. Initial population

A sequential strategy is used for obtaining a feasible solution. In this strategy, machines are first assigned to the each cell, randomly. This assignment makes matrix $[N]_{m \times c}$, and the only limitation for assignment is the upper bound of the cell size. Then the values of the members of matrix $[t]_{m_k \times h \times t}$ are produced randomly with respect to the number of each machine type in each cell. Also, another limitation is that all the members of layer $h = m$ in each three-dimensional matrix $t_{m_k \times h \times t}$ are zero. For instance, in Figure 4, $m = 2$ and so the entire layer $h = 2$ (the blue layer) is zero. Since the feasibility control of matrix $[t^+]_{m_k \times h \times t}$ is difficult, we consider a penalty for the infeasible solutions resulted from matrix $[t^+]_{m_k \times h \times t}$. Finally, parts are assigned to machines and to cells randomly with

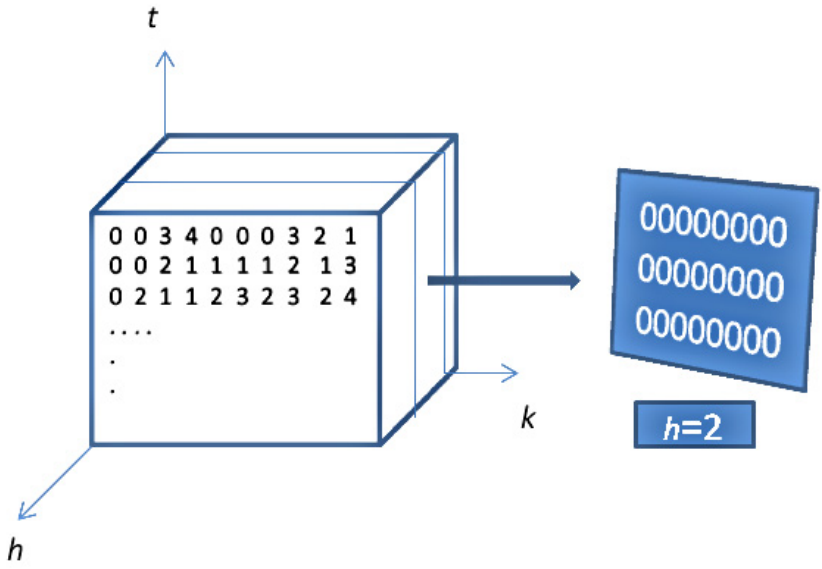


FIGURE 4. An illustration of three-dimensional matrix $t_{m_k \times h \times t}$ for the couple ($c = 1$, $m = 2$).

respect to the capacity of each machine in each cell, the number of machines in cells and from the situation of the modified machines. According to the proposed strategy, matrices $[N]_{m \times c}$ and $[t]_{m_k \times h \times t}$ are always feasible and therefore, the infeasibility of matrix $[t^+]_{m_k \times h \times t}$ affects only matrix $[X]_{p \times j \times t}$.

According to the proposed strategy, we create initial population of size of N_{country}. Subsequently, N_{imp} countries are selected from the best members- the ones with the lowest costs- of this group to be considered as the imperialists. Now there are N_{col} countries left as colonies. Then, proportional to the imperialists' power, calculated as below, the colonies are allocated to them.

$$A = \max\{c_n\} \quad \forall n \in N_{\text{imp}}$$
$$C_n = A - c_n$$

where c_n is the cost of the nth imperialist and C_n is its normalized cost. The cost of each country is appointed by its fitness function. In this paper, the fitness function is as the same as the single objective function, obtained using goal programming approach. Also, the imperialists' proportional power is computed as following:

$$p_n = \left| \frac{C_n}{\sum_{n=1}^{N_{\text{imp}}} C_n} \right|.$$

Consequently, the number of each imperialist's primary colonies equals $NC_n = \text{round}\,[p_n \cdot N_{\text{col}}]$. In this formula, NC_n represents an imperial's initial number of

colonies and 'round' is a function which shows the closest integer to a decimal. So, we select some primary colonies randomly for the nth imperialist and appoint them to it.

5.3. Solution improving

5.3.1. Crossover operator

After specifying the imperialists and their colonies, each colony has to move toward its imperialist. To approve it, we apply the Crossover operation of genetic algorithm on all the colonies by selecting a colony and applying the crossover operation between it and the imperialist. There are two countries as the product of each crossover operation. The lower cost produced country is compared to the colony. If this country has lower cost than the colony, it is substituted for the colony; otherwise, both countries are omitted. To apply the crossover operator, the Arithmetic Crossover operator is used. Two offspring of this operator are produced by their parents' linear combinations. These linear combinations are:

$$p_1 = \alpha x_1 + (1-\alpha)\, x_2 \quad p_2 = \alpha x_2 + (1-\alpha)\, x_1.$$

If $0 \leqslant \alpha \leqslant 1$, the values of p_1 and p_2 are between their parents' value. For this reason, it is assumed that $\alpha = U(-\gamma, 1+\gamma)$ where γ is free parameter which is considered $\gamma = 1$ in this paper. The crossover operator is applied to all the members of matrices $[N], [t]$, $[t^+]$, and concerning the mentioned strategy in Section 6.2 the feasibility of the solution is controlled. To apply the Crossover operator to matrix $[X]$ the same Arithmetic Crossover is used. However, the only difference is that α can be 0 or 1 at random.

5.3.2. Mutation operator

To search more space around imperialists, we use Mutation operator, which is another GA operator, for each colony. To fulfill it, we apply the Mutation operator to each colony and in the case of accessing a better result than Crossover result, we substitute the new country. Whereas, the Crossover operator led to omit the colony, the mutation function is applied as well and in the case of better result, it is substituted for the colony. To apply Mutation operator to a particular colony, as seen in Figure 5, the members of a block of matrix $[N]$ of the colony are replaced by those of the same block of matrix $[N]$ of the related imperialist. Considering the cell size restriction, the rest of the members of matrix $[N]$ are produced randomly. According to the mentioned strategy, we produce matrices $[t]$, $[t^+]$, $[X]$ considering matrix $[N]$. The produced solution is known as mutated country.

5.3.3. Colony and imperialist substitution

While colonies are moving toward their imperialists, they are possible to reach a better situation compared to the imperialist. In this case, the colony will be substituted for the imperialist, and the algorithm will continue with the new situation.

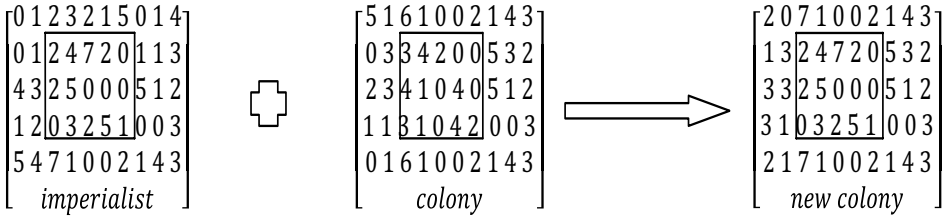


FIGURE 5. A schematic of the mutation operation.

5.4. Empire power

According to Atashpaz and Lucas [41], the power of an empire equals the imperialist's power plus a fraction of its total colonies' power. Thus, the power of an empire is computed as follows:

$$TC_n = \cos t\,(\text{imperialist}_n) + \alpha \cdot \text{mean}\,(\cos t\,(\text{colonies of empire}_n))$$

where TC_n indicates the total cost of an empire, and α represents a number between 0 and 1. In general, $\alpha = 0.3$ is a proper measure and has had a proper result for the proposed problem.

5.5. Imperial competition

According to the basic ICA, in this section β numbers of the weakest colonies of the weakest empire are considered, and then a competition among all the empires is arranged to take possession of these colonies. In this paper, we have considered $\beta = 3$. The colonies will not be possessed necessarily by the most powerful empire, but the more power, the more possibility. To model the empire competitions to possess these colonies, we first calculate the probability of each empire possession considering the total cost of an empire:

$$\begin{aligned} B &= \max\{TC_n\} \quad \forall n \in N_{\text{imp}} \\ NTC_n &= B - TC_n. \end{aligned}$$

In this equation, TC_n equals the total cost of the nth empire, and NTC_n represents the normalized cost by which the possession probability of each empire is computed as following:

$$p_{p_n} = \frac{NTC_n}{\sum_{n=1}^{N_{\text{imp}}} NTC_n}.$$

Using the above probability, we make vector p to randomly divide the mentioned colonies among the empires.

$$P = \left[p_{p_1}, p_{p_2}, \ldots, p_{p_{N_{\text{imp}}}}\right].$$

Then we create a vector with the same size as P whose elements are uniformly distributed random numbers.

$$R = \left[r_1, r_2, \ldots, r_{N_{\text{imp}}}\right].$$

Finally, vector H is formed as follows:

$$H = P - R = \left[p_{p_1} - r_1, p_{p_2} - r_2, \ldots, p_{p_{N_{\text{imp}}}} - r_{N_{\text{imp}}}\right].$$

Given vector H, we will hand the mentioned colonies to an empire whose relevant index in H is maximum. This level of the algorithm ends through possessing mentioned colonies.

5.6. Eliminating an empire

An empire is eliminated when it has lost all of its colonies. After a while, all the empires except the most powerful one will be collapsed. In such a condition we stop the algorithm and display the only imperialist as the optimal solution.

6. Genetic algorithm

Since GA is a classical metaheuristic algorithm yielding acceptable results, it has been used a lot in the literature (for example, in papers [38, 42, 43]). Thus, GA can be a valid measure to compare the proposed heuristic algorithm. To apply the classical GA, a few successive steps are implemented: 1. *Solution coding*: to code the solution of the problem, a special structure for the chromosome of GA is constructed. In this regard, we have employed the country structure used in the proposed ICA. 2. *Initial population*: the first step to start the algorithm is to produce the initial population. The number of the produced solutions (N_i) depends on the problem size, and in this paper, after analyzing the variety of sizes, we concluded the proper values. These values will be introduced in the experimental results section. 3. *Fitness value*: in order to assess the quality measurement of a solution or chromosome, fitness value is used as a criterion. In this paper, the fitness value is as the same as the single objective function obtained using goal programming approach. 4. *GA Operators*: the three well-known genetic algorithm operators are Inversion operator, Crossover operator, and Mutation operator. In this paper, Crossover and Mutation operators are considered to be used. To do the operations, firstly, we use Rolette wheel selection [44] to select the parents. It is assumed that the Crossover operator is assigned to the 80% of the initial population, and Mutation operator is applied to the rest of that. Both Mutation and Crossover operators are as the same as those of the improved ICA. 5. *Stopping criterion*: the new population, produced by Crossover and Mutation operators, has more individuals than the considered population size. Thus, the next step is to choose the best N_i individuals and consider them as the new generation. The number of generations N_g is considered as a stopping criterion. This number varies

1. *Create N_i chromosomes as the initial population randomly.*
2. *Calculate the fitness value of each solution (chromosome).*
3. *Select parents using Rolette wheel selection method.*
4. *Implement Crossover and Mutation operators.*
5. *Choose the best N_i individuals and consider them as the new generation.*
6. *If the number of generations is less than N_g, go to step 3; otherwise, go to step 7.*
7. *Display the best individual in the last generation.*

FIGURE 6. Genetic algorithm pseudo code.

depending on the problem size, and will be discussed more in the experimental results section.

7. Experimental results

In order to evaluate the effectiveness of the improved ICA for solving the proposed problem, we have solved 25 random instances (15 small/medium-sized + 10 large-sized). These problems have been generated randomly according to Table 1 based on similar data in the literature. All test problems have been solved by both GA and the improved ICA. These algorithms have been coded in MATLAB 7.0 and implemented on an Intel_Celeron_Mobile 2.5 GHz (core 2Duo) personal computer with 3 GB RAM. In order to verify the performance of the considered metaheuristics, for the small/medium-sized problems, the results of both GA and ICA are compared to the solutions obtained by GAMS 22.1\ DICOPT. As a quality criterion for these comparisons, we use the percentage of gap which is the deviation of the first objective function's values obtained by the metaheuristics ($ICAz$ and GAz) from those obtained by GAMS (Lz). The percentage of the gap is calculated as follows:

$$\%\text{Gap} = \frac{ICAz - Lz}{Lz} \times 100.$$

Since the second objective function adds neither new decision variables nor new constraints, so its effect on the value of the first objective function is more important than its value. Thus, it is enough to take into account only the value of z_1. For large-size problems the results of the improved ICA are compared to the GA-related results. Table 4 contains the results of ICA and GAMS for the small/medium-size problems. Table 5 compares the results of GA to those of GAMS for small/medium-size problems. The comparison of ICA to GA for the large-size problems is reported in Table 6. Also, the ICA parameter setting is shown in Table 3.

The metaheuristic algorithms have been run four times for each test problem and the 'Average' columns show the mean of the values obtained in the individual

TABLE 3. ICA parameter settings.

ICA Parameters	Small and medium size problems					Large size problems				
	N_{country}	N_{imp}	N_{col}	α	β	N_{country}	N_{imp}	N_{col}	α	β
Value	150	20	130	0.3	3	300	60	240	0.3	3

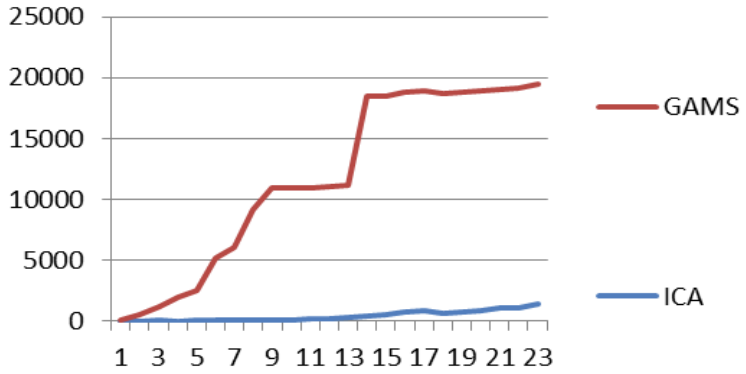


FIGURE 7. Comparison of the solving times between ICA and GAMS for the 23 test problems.

runs. In Tables 4 to 6, p, O_p, t and M indicate the number of part types, the number of operations, the number of periods and the number of machine types respectively. For simplicity, we assumed that all the part types have the same number of operations. It is noted that for all small and medium-size problems, the run time is limited to three hours (10 800 s). In Tables 5 and 6, Ni and Ng indicate the number of population and generation of GA, respectively. According to Tables 4 and 5, although both metaheuristics have reasonable solutions compared to GAMs, the average of Gap^{mean} and Gap^{best} have been improved in the proposed ICA in comparison to the classical GA. Also, both metaheuristics have more practical solving times compared to those of GAMS. Figure 7, for example, compares the solving times of ICA to GAMS. However, Tables 4 and 5 show that the average of CPU time has been improved more when using the proposed ICA. For large-size problems, Table 6 shows that the average of the optimal values obtained by the improved ICA is better than the GA's. Figure 8 indicates this superiority. Also, according to Table 6, the average CPU time for the ICA is 863 s while it is 902.8 s for the classical GA. This superiority is indicated in Figure 9. To sum up with respect to the mentioned measures, although both of the algorithms have reasonable solutions, the proposed ICA dominates the classical GA in solving the proposed problem.

TABLE 4. Comparison between B&B and ICA for small and medium-size problems.

No.	Problem info.	Improved ICA Objective Value			Improved ICA	GAMS	GAMS	%Gap	
	$P^*O_p^*t^*M$	Max	Min	Average	average time (s)	objective value	time (s)	Gap^{mean}	Gap^{best}
1	2*2*1*3	305	305	305	4	305	1	0	0
2	2*2*2*3	395	390	391.25	6	390	12	0.30	0
3	4*3*3*4	629	625	627	16	625	100	0.32	0
4	6*3*3*5	831	831	831	25	822	504	1.09	1.09
5	7*4*4*6	968	950	957	39	946	1205	1.16	0.42
6	8*5*4*6	1099	1076	1089.5	38	1069	1900	1.92	0.65
7	8*6*5*7	1223	1190	1200.75	52	1174	2502	2.28	1.36
8	10*7*6*8	1411	1390	1393.5	73	1353	5064	3.00	2.73
9	12*8*6*8	1600	1538	1543.53	82	1505	6034	2.56	2.19
10	15*10*6*10	1970	1921	1938.01	102	1878	9122	3.20	2.29
11	20*10*8*12	2298	2257	2279.6	128	2204	>10 800	3.43	2.40
12	22*13*10*15	2501	2461	24 085.85	141	2409	>10 800	3.19	2.16
13	25*15*10*17	2741	2701	2734.27	152	2639	>10 800	3.61	2.35
14	30*20*12*20	3650	3560	3567.67	204	3622	>10 800	–1.50	–1.71
15	35*20*15*25	4043	3956	3959.6	346	4033	>10 800	–1.82	–1.91
					Ave. = 93.86			Ave. = 1.52	Ave. = .93

TABLE 5. Comparison between B&B and GA for small and medium-sized problems.

No.	Problem info.	GA objective value					GA average	GAMS	GAMS	%Gap	
	$P^*O_p^*t^*M$	Max	Min	Ni	Ng	Average	Time (s)	Objective value	Time (s)	Gap^{mean}	Gap^{best}
1	2*2*1*3	305	305	200	150	305	4	305	1	0	0
2	2*2*2*3	395	390	200	150	391.18	6	390	12	0.30	0
3	4*3*3*4	633	625	200	150	628.5	16	625	100	0.56	0
4	6*3*3*5	842	837	200	150	839.3	28	822	504	12.1	2.07
5	7*4*4*6	979	959	200	150	960.3	40	946	1205	1.51	1.37
6	8*5*4*6	1106	1086	200	150	1092.5	48	1069	1900	2.2	1.59
7	8*6*5*7	1231	1191	200	150	12094	62	1174	2502	302	1.45
8	10*7*6*8	1414	1396	320	200	1401.5	89	1353	5064	3.58	3.18
9	12*8*6*8	1610	1545	320	200	1558.5	102	1505	6034	3.55	2.66
10	15*10*6*10	1981	1930	320	200	1939	127	1878	9122	3.25	2.77
11	20*10*8*12	2300	2259	320	200	2284	143	2204	>10 800	3.63	2.50
12	22*13*10*15	2520	2461	320	200	2489.8	159	2409	>10 800	3.35	2.16
13	25*15*10*17	2780	2756	320	200	2766.2	194	2639	>10 800	4.82	4.43
14	30*20*12*20	3668	3573	320	200	3601.4	252	3622	>10 800	0.57	–1.35
15	35*20*15*25	4099	3995	320	200	40515	382	4033	>10 800	0.46	–0.94
							Ave. = 110.13			Ave. = 2.12	Ave. = 1.46

TABLE 6. Comparison between ICA and GA for large-sized problems.

No.	Problem info.	Developed ICA objective value			Developed ICA average	GA objective value					
	$P^*O_p^*t^*M$	Max	Min	Average	time (s)	Max	Min	Average	Ni	Ng	GA average time (s)
16	40*20*17*25	4693	4602	4645.6	453	4780	4651	4725	600	450	478
17	45*25*20*30	5635	5460	5529.2	509	5542	5492	5536	600	450	536
18	48*25*25*30	5780	5689	5715.2	788	5729	5693	5720.5	600	450	809
19	50*28*25*35	6267	6040	6133	894	6305	6067	6254	600	450	895
20	50*28*30*35	6793	6602	6629.4	687	6814	6629	6751.2	600	450	905
21	53*30*30*38	7112	6810	6973.7	777	7056	6982	7000.6	600	450	845
22	53*30*33*38	7298	7008	7223	871	7400	7065	7321	600	450	879
23	58*33*33*45	8508	7914	8209.4	1073	8408	7995	8339.5	600	450	1086
24	58*33*40*45	7815	7709	7775	1120	8005	7709	7778	600	450	1105
25	60*40*40*50	8805	8691	8733.3	1458	8884	8732	8786.8	600	450	1490
		Ave. = 6652.5			Ave. = 863	Ave. = 6701.5					Ave. = 902.8

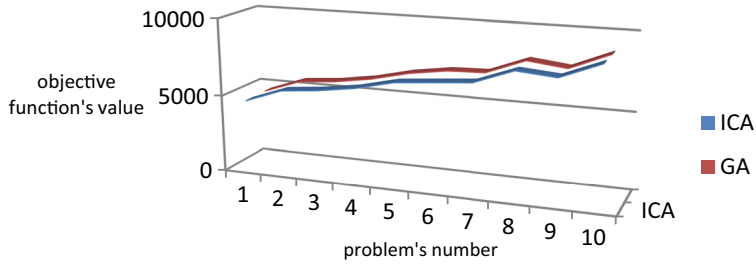


FIGURE 8. Comparison of ICA to GA in terms of their best objective function values.

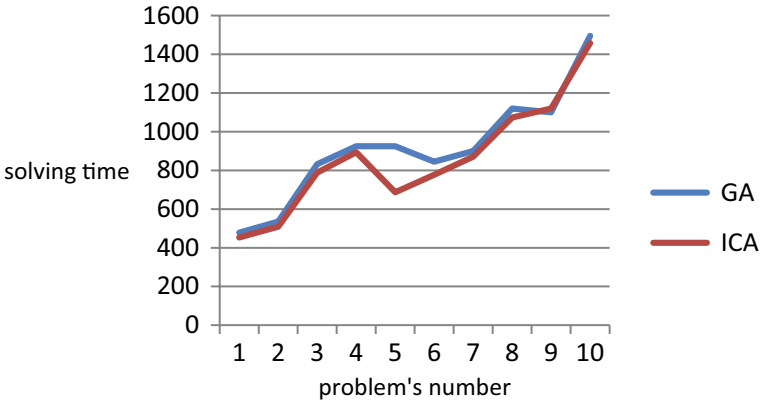


FIGURE 9. Comparison of ICA to GA in terms of their average solving times.

8. Conclusions and future research directions

Today, competitive environment and shorter product life cycles have caused the emergence of dynamic cellular manufacturing systems whose cells are reconfigured during different planning periods by means of machine relocation, replication, or removing. In most cases, machine relocation is too costly or even impossible due to reasons, such as machine sizes and skilled workers movements. To tackle this shortcoming, this paper considered RDCFP in which machine modifications were performed in order to respond the need to cell reconfiguration. In this regard, this paper proposed a mixed-integer nonlinear mathematical model which was NP-hard. Therefore, an ICA was developed in order to tackle the proposed discrete model, although it was tested on continuous-variable models. The ICA is selected because to the best of our knowledge it was the first research adopting ICA in the field of CMS and DCFP. Also, obtained results of the improved ICA were compared with those of GA, which validate outperformance of the developed ICA.

In order to continue current research direction of this paper, recommendations are threefold. First, modeling and analysis of RDCFP using the approach upon adjacency matrices might be interesting. Also, considering setups might yield more realistic results in different production settings. Mathematical models of such manufacturing environments have more binary variables intensifying computational complexity of the developed model. At last, it is highly suggested to make effort to linearize CFP mathematical model's binary variables. It might help practitioners put in practice the concept of cell reconfigurability.

Acknowledgements. The authors would like to acknowledge the financial support of University of Tehran for this research under grant number 8109002/1/08. The authors are also grateful to the editor of *RAIRO-Operations Research* as well as the anonymous referees for their valuable and constructive comments to enhance quality of the paper.

References

[1] J.A. Tompkins, J.A. White, Y.A. Bozer and J.M.A. Tanchoco, Facilities Planning. Wiley, USA (2003).
[2] J. Balakrishnan and C.H. Cheng, Multi-period planning and uncertainty issues in cellular manufacturing: A review and future research directions. *Eur. J. Oper. Res.* **177** (2007) 281–309.
[3] M. Rheault, J.R. Drolet and G. Abdulnour, Physically reconfigurable virtual cells: A dynamic model for a highly dynamic environment. *Comput. Ind. Eng.* **29** (1995) 221–225.
[4] L.K. Saxena and P.K. Jain, Dynamic cellular manufacturing systems design-a comprehensive model. *Int. J. Adv. Manuf. Technol.* **53** (2011) 11–34.
[5] L.R. Foulds, A.P. French and J.M. Wilson, The sustainable cell formation problem: manufacturing cell creation with machine modification costs. *Comput. Oper. Res.* **33** (2006) 1010–1032.
[6] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy and H. Van Brussel, Reconfigurable Manufacturing Systems. *Ann. CIRP* **48** (1999) 1–14.
[7] M.G. Mehrabi, A.G. Ulsoy and Y. Koren, Reconfigurable manufacturing systems: key to future manufacturing. *J. Intelligent Manuf.* **11** (2000) 403–419.
[8] C.L. Grahl, Increasing efficiencies with synthetic dies. *Ceramic Ind.* **151** (2001) 31–32.
[9] G.F.K. Purcheck, A mathematical classification as a basis for the design of group technology production cells. *Prod. Eng.* **54** (1974) 35–48.
[10] A. Kusiak, The generalized group technology concept. *Int. J. Prod. Res.* **25** (1987) 561–569.
[11] A. Shtub, Modeling group technology cell formation as a generalized assignment problem. *Int. J. Prod. Res.* **27** (1989) 775–782.
[12] J.C. Wei and N. Gaither, A capacity constrained multi objective cell formation method. *J. Manuf. Syst.* **9** (1990) 222–232.
[13] S. Song and K. Hitomi, Integrating the production planning and cellular layout for flexible cell formation. *Prod. Plan. Control.* **7** (1996) 585–593.
[14] N. Safaei, M. Saidi–Mehrabad and M.S. Jabal–Ameli, A hybrid simulated annealing for solving an extended model of dynamic cellular manufacturing system. *Eur. J. Oper. Res.* **185** (2008) 563–592.
[15] M.M. Ghotboddini, M. Rabbani and H. Rahimian, A comprehensive dynamic cell formation design: Benders' decomposition approach. *Exp. Syst. Appl.* **38** (2011) 2478–2488.
[16] A. Kusiak, The generalized group technology concept. *Int. J. Prod. Res.* **25** (1987) 561–569.
[17] A. Shtub, Modeling group technology cell formation as a generalized assignment problem. *Int. J. Prod. Res.* **27** (1989) 775–782.

[18] I. Mahdavia, A. Aalaei, M.M. Paydar and M. Solimanpur, Designing a mathematical model for dynamic cellular manufacturing systems considering production planning and worker assignment. *Comput. Math. Appl.* **60** (2010) 1014–1025.
[19] S.I. Satoglu and N.C. Suresh, A goal-programming approach for design of hybrid. *Comput. Ind. Eng.* **56** (2009) 560–575.
[20] A. Mungwattana, Design of cellular manufacturing systems for dynamic and uncertain production requirement with presence of routing flexibility. Ph.D. dissertation, Blacksburg State, University of Virginia (2000).
[21] R. Tavakkoli–Moghaddam, M.B. Aryanezhad, N. Safaei and A. Azaron, Solving a dynamic cell formation problem using metaheuristics. *Appl. Math. Comput.* **170** (2005) 761–780.
[22] M. Saidi–Mehrabad and N. Safaei, A new model of dynamic cell formation by a neural approach. *Int. J. Adv. Manuf. Technol.* **33** (2007) 1001–1009.
[23] R. Tavakkoli–Moghaddam, N. Safaei and A. Sassani, A new solution for a dynamic cell formation problem with alternative routing and machine costs using simulated annealing. *J. Oper. Res. Soc.* **59** (2008) 443–454.
[24] F. Defersha and M. Chen, A comprehensive mathematical model for the design of cellular manufacturing systems. *Int. J. Prod. Econ.* **103** (2006) 767–783.
[25] S. Ahkioon, A.A. Bulgak and T. Bektas, Cellular manufacturing systems design with routing flexibility, machine procurement, production planning and dynamic system reconfiguration. *Int. J. Prod. Res.* **47** (2009) 1573–1600.
[26] M.B. Aryanezhad, V. Deljoo and S.M.J. Mirzapour Al-e-hashem, Dynamic cell formation and the worker assignment problem: a new model. *Int. J. Adv. Manuf. Technol.* **41** (2009) 329–342.
[27] SMJ. Arkat and B. Abbasi, Applying simulated annealing to cellular manufacturing system design. *Int. J. Adv. Manuf. Technol.* **32** (2007) 531–536.
[28] N. Aljaber, W. Baek and C.L. Chen, A tabu search approach to the cell formation problem. *Comput. Int. Eng.* **32** (1997) 169–185.
[29] K. Spiliopoulos and S. Sofianopoulou, Designing manufacturing cells: a staged approach and a tabu search algorithm. *Int. J. Prod. Res.* **41** (2003) 2531–2546.
[30] R. Logendran and Y. Karim, Design of manufacturing cells in the presence of alternative cell locations and material transporters. *J. Oper. Res. Soc.* **54** (2003) 1059–1075.
[31] T.H. Wu, C. Low and W.T. Wu, A tabu search approach to the cell formation problem. *Int. J. Adv. Manuf. Technol.* **23** (2004) 916–924.
[32] D. Lei and Z. Wu, Tabu search for multiple-criteria manufacturing cell design. *Int. J. Adv. Manuf. Technol.* **28** (2006) 950–956.
[33] M.S. Yang and J.H. Yang, Machine-part cell formation in group technology using a modified ART1 method. *Eur. J. Oper. Res.* **188** (2008) 140–152.
[34] P. Venkumar and A.N. Haq, Manufacturing cell formation using modified ART1 networks. *Int. J. Adv. Manuf. Technol.* **26** (2005) 909–916.
[35] M. Solimanpur, P. Vrat and R. Shankar, A multi-objective genetic algorithm approach to the design of cellular manufacturing systems. *Int. J. Prod. Res.* **42** (2004) 1419–1441.
[36] V. Venugopal and T.T. Narendran, A genetic algorithm approach to the machine component grouping problem with multiple objectives. *Comput. Ind. Eng.* **22** (1992) 469–480.
[37] H. Pierreval, C. Caux, J.L. Pairs and F. Viguier, Evolutionary approaches to the design and organization of manufacturing systems. *Comput. Ind. Eng.* **44** (2003) 339–364.
[38] F.M. Defersha and M. Chen, Machine cell formation using a mathematical model and a genetic-algorithm-based heuristic. *Int. J. Prod. Res.* **44** (2006) 2421–2444.
[39] X. Wu, C. Chao–Hsien, Y. Wang, W. Yan, A genetic algorithm for cellular manufacturing design and layout. *Eur. J. Oper. Res.* **181** (2007) 156–167.
[40] K. Yasuda, L. Hu, Y. Yinza, Grouping genetic algorithm for the multi-objective cell formation problem. *Int. J. Prod. Res.* **43** (2005) 829–853.
[41] E. Atashpaz–Gargari and C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *IEEE Cong. Evolut. Comput. Singapore* (2007) 4661–4667.

[42] F. Sarayloo and R. Tavakkoli–Moghaddam, Imperialistic Competitive Algorithm for Solving a Dynamic Cell Formation Problem with Production Planning. *Adv. Intell. Comput. Theor. Appl.* **6215** (2010) 266–276.
[43] G.C. Onwubolu and M. Mutingi, a genetic algorithm approach to cellular manufacturing systems. *Comput. Ind. Eng.* **39** (2001) 125–144.
[44] E.G. Talbi, Metaheuristic from design to implementation. John Wiley & Sons Publisher: USA (2009).