International Journal of Industrial and Manufacturing Systems Engineering
2016; 1(3): 52-58
http://www.sciencepublishinggroup.com/j/ijimse
doi: 10.11648/j.ijimse.20160103.12

Optimization of a New Tool Switching Problem in Flexible Manufacturing Systems with a Tool Life by a Genetic Algorithm

Hamid Dadashi, Shiva Moslemi, Abolfazl Mirzazadeh

Department of Industrial Engineering, Kharazmi University, Tehran, Iran

Email address:
shivamoslemi1991@yahoo.com (S. Moslemi)

To cite this article:
Hamid Dadashi, Shiva Moslemi, Abolfazl Mirzazadeh. Optimization of a New Tool Switching Problem in Flexible Manufacturing Systems with a Tool Life by a Genetic Algorithm. *International Journal of Industrial and Manufacturing Systems Engineering.* Vol. 1, No. 3, 2016, pp. 52-58. doi: 10.11648/j.ijimse.20160103.12

Received: October 31, 2016; **Accepted**: November 11, 2016; **Published**: December 23, 2016

Abstract: This paper considers a tool switching problem (ToSP) in flexible manufacturing systems (FMSs). Indeed, a new version of the ToSP that may be faced in practice is proposed. In this paper, a tool life is considered for each tool and a new formulation of the ToSP is presented, however, because of the complexity of such an NP-hard problem, the exact method can't be used to solve large-sized problems. Therefore, a well-known meta-heuristic method, genetic algorithm (GA), is proposed to solve the problem. Furthermore, the computational results obtained by the proposed GA and the B&B methods are compared by performing on several non-large dimension instances. Finally, it is shown that the GA results are promising.

Keywords: Tool Switching, Flexible Manufacturing System, Total Part Tardiness, Tool Purchasing Cost, Genetic Algorithm

1. Introduction

Current trends in manufacturing systems are to move towards highly flexible production systems that can respond quickly to demand changes and to the processing of a variety of products. Flexible manufacturing systems (FMSs) are considered as one of the most important production technologies to efficiently handle today's changes in market requirements. An FMS consists of a computer-controlled, integrated configuration of numerically controlled machine tools with automated material handling systems.

Problems of a flexible manufacturing technology are relatively complex compared to traditional manufacturing systems, this difficulty originates primarily from the fundamental objective behind the FMS concept that said to be as efficient as a mass production facility and yet as flexible as a job shop facility. Since each machine in an FMS is quite versatile and capable of performing many different operations and each part type may have alternate routes through the system, it becomes very complex to solve FMS planning, scheduling, and operational problems.

Management tool, which is one of the FMS problem sections, is about to get the right tool, to the right place at the right time. The need for tool management stems from the high variety of tools that are typically found in automated manufacturing systems.

The experiments show significant costs can be avoided by performing appropriate tool management strategies. In this study, we deal with a single machine that has several slots in, which different tools can be loaded. Each tool occupies only one slot and each part type executed on machine requires a particular set of operation to be done. Part types are sequentially executed and only one part type can be installed on the machine each stage and therefore, each time a part type will be processed, the corresponding tools should be loaded into the machine magazine. If the number of available slots is limited, it should be required at some points a tool switch (i.e., removing a tool from the magazine and inserting another one in its place). Although the order of tools in the magazine is often irrelevant, the need of performing a tool switching is dependent on the order, in which the part type is executed. It is assumed that the tools are versatile and considered a tool life for each tool, as we know there is not a considerable amount of the literature related to a tool life in an FMS. The objective is to minimize the total part type tardiness and tool purchasing cost.

In this study a new formulation of the given problem is presented; but because of the complexity of the NP-hard

problem, we cannot use an exact method (e.g., branch-and-bound) to solve the large-sized problem. Therefore, we propose a well-known meta-heuristic method, namely genetic algorithm (GA), to solve the given problem. We compare the proposed GA and the branch-and-bound (B&B) method by performing several non-large dimension instances. Finally, it is shown that the associated results are promising.

2. Literature Review

The previous research carried out in tool switching problems can be found in the early of 60's by Belady [1]. Then, the uniform tool switching (each tool occupies one slot) has been tackled by a number of different techniques. In the late 80's, a number of studies have been contributed specially to solve the problem by El Maraghy [2] and Kiran and Krason [3]. Tang and Denardo [4], proposed an ILP formulation of the problem, and later Bard [5], described a non-linear integer programming formulation with a dual-based relaxation heuristic method. Heuristic-based constructive methods have been also applied to the problem. For instance, Djellab *et al.* [6], tackled the ToSP by a hypergraph representation and proposed a particular heuristic method oriented towards minimizing the number of gaps in edge-projection. They used the hypergraph to represent the relation among products and the needed tools. Also, Hertz *et al.* [7], described three constructive methods, namely FI, GENI and GENIUS, in their model at each step first a job is selected to be inserted in current tour and then best position in the tour are selected. In addition, the nearest neighbor (NN) and 2-opt search methods were also considered.

The exact methods have been also applied to the problem. For instance, Laporte *et al.* [8], proposed two exact algorithms, namely a branch-and-bound approach and linear programming-based branch-and-cut algorithm. Precisely, the last one is based on a new ILP formulation having a better linear relaxation than that proposed previously by Tang and Denardo [4]. It should be noted that these exact methods are inherently limited, since Oerlemans [9] and Crama *et al.* [10], proved formally that the tool switching problem (ToSP) is NP-hard for $c>2$, where c is the number of the slots in the machine's magazine. This limitation has been already highlighted, where Laporte *et al.* [8], reported that their algorithm was able to manage instances with 9 products; but, it presented a very low success ratio for instances over 10 products.

Sarin and Chen [11], also formulated the machine loading and tool allocation as 0–1 linear programming. Part assignments and tool allocations are determined simultaneously considering a tool life, tool size (i.e. number of slots a tool occupies on tool magazine) and magazine capacity. Buyurgan et al. [12], introduced a heuristic approach for tool selection and allocation. Their proposed approach utilizes the ratio of a tool life over a tool size for tool selection and allocation they also consider a tool life.

The clustering/grouping methods have also been attended. For instance, Salonen *et al.* [13], attacked the uniform ToSP of the printed circuit boards (PCBs) and described an algorithm that iterated the process of first determining a good (or even optimal) grouping of the PCBs for further sequencing them. A hierarchical product grouping technique, based on the Jaccard's similarity coefficient as clustering criterion, is also employed to avoid identical groupings. A different and very interesting approach has been described by Zhou *et al.* [14], who proposed a beam search algorithm. This method was especially efficient and practical in comparison to the previous techniques. The reason for this result is that the performance of the algorithm can be adjusted by changing the search and evaluation functions.

The meta-heuristic methods have been also used recently. So, several tabu search (TS) methods have been used in the literature by Salonen *et al.* [13], Herts and Widmer [15], Al Fawzan and Al Sultan [16], Amaya et al. [17], Konak et al. [18].

3. Problem Statement

In this section, the assumptions, mathematical formulation and meta-heuristic method of the presented model are elaborated.

3.1. Assumptions

Except the other usual assumptions of tool switching problem (that are said in introduction section) there are other new assumptions in this model that are listed below:

(1) Operating times for all part type operations on different tool types are known.
(2) The purchase cost of each tool type is known.
(3) Each tool has a tool life that specified in advance if its remaining life is over and we need more then it will be substituted with the new one.
(4) The number of the magazine slot (i.e., magazine capacity) should be specified in advance.
(5) Each tool type can perform one or more operations (i.e., versatile tool).
(6) Required time for switching tools is constant and known.
(7) The release time is the time that all of part types are done (i.e., it is considered makespan for the presented model).
(8) The due date should be specified in advance.
(9) The tardiness penalty is considered when the release time is greater than the due time.
(10) The insert of a new tool type is not allowed except when the last tool type remaining capacity is zero or there is no remaining capacity to assign.

3.2. Input Parameters

Input parameters are as follow:

d_k	tool life for tool k (k=1,…, K)
tsw	required time for tool switching
t_{lk}	required time for doing operation l (l=1,…,L) by using tool k
duedate	due date of part types
penalty	rate of penalty of tardiness
c_k	purchasing cost of tool k
C	magazine capacity

$$a_{l,k} = \begin{cases} 1 \text{ if tool } k \text{ can do operation } l \\ 0 \text{ otherwise} \end{cases}$$

$$b_{i,l} = \begin{cases} 1 \text{ if part type } i\ (i = 1, \dots, n) \text{ needs operation } l \\ 0 \text{ otherwise} \end{cases}$$

3.3. Decision Variables

Decision variables are as follow:

$$x_{ij} = \begin{cases} 1 \text{ if part type } i \text{ is scheduled to stage } j\ (j = 1, \dots, n) \\ 0 \text{ otherwise} \end{cases}$$

$$z_{ikl} = \begin{cases} 1 \text{ if operation } l \text{ on part type } i \text{ is done by tool } k \\ 0 \text{ otherwise} \end{cases}$$

y_{kj}	Number of tool type k on stage j
yy_{kj}	Number of new tool type k that inserted on stage j
ru_{kj}	The remaining capacity of last tool type k that is inserted into magazine at the beginning of stage j
pp_{kj}	Number of non-new tool type k that are inserted or removed on stage j (with consideration of previous stage)
nu_{kj}	Number of using tool k on stage j
ntu_k	Total number of tools using of tool type k
duepenalty	Total cost of tardiness
purcost	Total cost of purchasing of tools
sw	Total number of the tool switching
ft	Releasing time of total part type

3.4. Mathematical Formulation

The objective of the mathematical model is minimizing the total cost of tool purchasing and tardiness of part types. The mathematical formulation is as follows:

$$\min U = \text{duepenalty} + \text{purcost} \quad (1)$$

s.t.

$$\sum_i x_{ij} = 1\ \forall j \quad (2)$$

$$\sum_j x_{ij} = 1\ \forall i \quad (3)$$

$$\sum_k y_{kj} = C\ \forall j \quad (4)$$

$$\sum_i \sum_l \frac{z_{ikl}}{d_k} \le ntu_k\ \forall k \quad (5)$$

$$\sum_k ntu_k \times c_k = purcost \quad (6)$$

$$\sum_i \sum_l x_{ij} \times z_{ikl} = nu_{kj}, \forall k, j \quad (7)$$

$$ru_{k,j-1} + \left(yy_{k,j-1} \times d_k\right) - nu_{k,j-1} = ru_{kj}, \forall k, j \quad (8)$$

$$nu_{k,j} - ru_{k,j} \le d_k \times yy_{kj}, \forall k, j \quad (9)$$

$$nu_{k,j} \le d_k \times y_{kj}, \forall k, j \quad (10)$$

$$yy_{k,j} \le y_{k,j}, \forall k, j \quad (11)$$

$$yy_{k,1} = y_{k,1}, \forall k \quad (12)$$

$$ru_{k,j} \le d_k, \forall k, j \quad (13)$$

$$ru_{k,1} = 0, \forall k \quad (14)$$

$$\sum_k \sum_{j=2}^{n} \frac{yy_{kj} + pp_{kj}}{2} = sw \quad (15)$$

$$y_{k,j-1} - \left(y_{kj} - yy_{kj}\right) \le pp_{kj}, \forall k, j \quad (16)$$

$$-\left(y_{k,j-1} - \left(y_{kj} - yy_{kj}\right)\right) \le pp_{kj}, \forall k, j \quad (17)$$

$$\left(\sum_i \sum_k \sum_l z_{ikl} \times t_{lk}\right) + (sw \times tsw) = ft \quad (18)$$

$$(ft - duedate) \times penalty \le duepenalty \quad (19)$$

$$\sum_k z_{ikl} = b_{il}, \forall i, l \quad (20)$$

$$z_{ikl} \le a_{lk} \times b_{il}, \forall i, l, k \quad (21)$$

$$x_{ij}, z_{ikl} = 0 \text{ or } 1, y_{kj}, nu_{kj}, yy_{k,j}, ntu_k, pp_{k,j} \ge 0 \text{ and integer}, duepenalty \ge 0$$

Constraint 2 ensures that only one part type assigns to each stage. Constraint 3 ensures each part type assigns to one stage. Constraint 4 ensures that the number of assigned tools to each stage is equal to the magazine capacity. Constraint 5 counts the number of tools using of each tool type, constraint 6 specifies the total purchasing cost of tools. Constraints 7 specifies Number of using tool k on stage j (for all k and j), constraint 8 specifies The remaining capacity of last tool type k that is inserted into magazine at the beginning of stage *j* (for all k and j), constraint 9 specifies Number of new tool type k that inserted on stage j (for all k and j), constraint 10 specifies Number of tool type k on stage j (for all k and j), constraint 11 ensures that the number of new tool does not exceed the number of total tools, constraint 12 assigns the value of number of new tools to number of total tool, for the first stage, constraint 13 ensures that the remaining tool life does not exceed capacity of tools in other word Constraint 13 ensures Assumption 10 given in Section 3.1, constraint 14 assigns the value zero for remaining tool life at the first stage,

Constraints 15 counts the number of total tool switching, Constraint 16 and 17 counts the number of non-new tool type k that is inserted or removed on stage j (pp_{kj}), Constraint 18 specifies the finishing time of all part types. Constraint 19 specifies the penalty cost of tardiness. Constraints 20 and 21 ensure that the value of z_{ikl} corresponding to specify the tool, operation and part type. Constraint 20 also ensures that if part type *i* needs operation *l*, it should be done only by one tool.

3.5. Proposed Genetic Algorithm

Because of the complexity of the given problem known as NP-hard one, it can't be used any exact method (e.g., branch-

and-bound) for solving large-sized problems. Therefore, in this section a meta-heuristic method based on a genetic algorithm (GA) in order to solve such a hard problem is proposed. The chromosome structure and coding system are inspired by a coding method that introduced by Tavakkoli-Moghaddam et al. [19]. The solution coding and method of our proposed GA are described below.

3.5.1. Chromosome Structure

A feasible solution or chromosome consists of the following genes.

(1) The gene related to the assignment of a tool to the operations of the part type is named as matrix $[X_{i,j}]$. For example, $X_{2,3} = 5$ means that tool 5 is used for doing operation 3 on part type 2 (if $a_{35}\times b_{23}$ =1), note that if b_{ij}=0, then the corresponding allele will be zero.

(2) The gene related to the assignment of the part type to the stage is named as matrix $[Y_i]$ that alleles are limited to 1, 2,…, n (n is a number of the part type). For example, $Y_2 = 5$ means that part type 2 assign to stage 5.

(3) The gene related to the remaining tool life at the end of specific part types is named $[F_{i,j}]$. For example, $F_{3,6}$=4 means the remaining life of tool 6 after processing part type 3 is 4 (it means that it can be used 4 times more for the next part types until its failure and after that if it is needed this tool more, A new tool type 6 should be inserted). During the process of part types, matrix $[F_{i,j}]$ will be continually up to date. For example, if a specific tool type that is selected before (for the previous operation of the current part type), is selected and the capacity of a tool is not zero then it should be used the same tool that is selected before (without using another tool) otherwise it should be inserted a new tool type and up to date matrix F. It means more than one tool with the same type are exist on the current stage.

(4) The gene related to counting the failed tool of each tool type on the magazine when a specific part type is installed on a machine is named $[G_{i,j}]$. For example, $G_{1,5}$=2 means when part type 1 is on machine, a number of failed tool type 5 that exist on the magazine is 2.

(5) The gene related to counting the total tool number of each tool type on the magazine when a specific part type is on a machine is named $[H_{i,j}]$. For example, $H_{1,5}$=2 means when part type 1 is on a machine, a number of total tool type 5 that exist on magazine is 2.

(6) The gene indicates which tools exist on a magazine is named $[MAG_{i,j}]$. For example, $MAG_{2,3}$ =3 means when part type 2 is on a machine, tool 3 exists on slot 3. Note that the order and layout of tools in a magazine are not considered.

The feasible solution chromosome structure will be as follows:

$$\lfloor [X]_{n\times L}, [Y]_{n\times 1}, [F]_{n\times K}, [G]_{n\times K}, [H]_{n\times K}, [MAG]_{n\times C} \rfloor$$

where n, L, K and C denote a number of the part type, total operation tool and magazine capacity, respectively. To generate a new solution, only matrices X and Y are considered and arrays of these two matrices are changed, then the other matrices are been up to date by means of them.

3.5.2. Generation of an Initial Population

To obtain the initial population, first a random feasible solution of X and Y are created separately, then F, G, H and MAG are obtained and been up to date base on X and Y. To create a random feasible solution X, for each array of X, tool k is selected to be used for operation l on part type i randomly if $a_{lk}\times b_{il}$ =1. In other words, A tool among the tools that can be used for doing the corresponding operation is selected randomly. Note that values of arrays with b_{il} =0 are zero constantly.

To Create a random feasible solution Y, a random permutation of 1 to $n\times Z$ *is generated.*

As we mention after generation of X and Y, other matrices are obtained based on data of these two matrices. Also for obtaining MAG, a tool loading method is used. This method is based on the KTNS (keep tool needed soonest) method first introduced by Belady [1], Tang and Denardo [4] and Bard [5]. The aim of the KTNS rule is to minimize a number of tool switching. Following is our proposed algorithm to load the magazine at stages:

- Step 1: According to the matrix Y for all stages inserts the tools that are selected at the corresponding row (i.e., part type that assigned to current stage) of matrix X (with considering and up to date matrix *F)*.
- Step 2: For i=1 to n, do as follows (i denotes a stage)
- Step 2-1: If i =1, then go to Step 2-3; otherwise, if one or more tools are to be inserted and there are no vacant slots in the magazine, then go to step2-2; otherwise, set i=i+1 and go to Step 2-4.
- Step 2-2: Remove the failed tools until the number of a vacant slot is sufficient for the remaining inserted tool on Step 2-1; but, if more vacant slots are still needed and there are no more failed tool on a magazine, then use the KTNS method and keep the tools that are needed soonest and remove other tools to make a vacant slot, set i=i+1 and go to Step 2-4.
- Step 2-3: Select the new tool types that need the soonest until the magazine is became full, and then set i=i+1.
- Step 2-4: End.

Note 1: Consider Step 2-3 if there is a tool selected before at the current stage; but, its remaining capacity is zero. If it is needed soonest, then a new tool of this type can be selected and the corresponding array on matrix F is being up to date.

Note 2: Only the tools that are selected in Step 1 are used at each stage.

3.5.3. Fitness Value

The fitness value is the same objective function presented in Section 3.4. Also, an algorithm is proposed to count the number of tool switching on a magazine. This algorithm is explained bellow.

Algorithm parameters:

$\acute{G}_{ij}$ number of failed tool type j at stage i that exists on a magazine

$\acute{H}_{ij}$ number of total tool type j at stage i that exists on a magazine

$\acute{F}_{ij}$ remaining capacity of tool type j at the end of stage i

Algorithm steps:

Consider the initial value of a number of tool switching is zero, sw=0 and for all i and $j \mid \acute{H}_{ij} \neq 0 \,\&\, i \neq n$, do as follows ($i$ and j denote the stage and tool type, respectively):

I If $\acute{H}_{(i+1),j} \neq 0$, then go to Step II; otherwise, $sw = sw + (\acute{H}_{ij})$ and go to Step III.

II If $\acute{F}_{ij} = 0$, then $sw = sw + (\acute{H}_{ij} - \acute{G}_{(i+1),j})$; otherwise, $sw = sw + ((\acute{H}_{ij} - 1) - \acute{G}_{(i+1),j})$.

III End.

Note1: Consider Step II if the remaining capacity of tool type j that exists on a magazine at stage i is not zero ($\acute{F}_{ij} \neq 0$). Then, according to assumption 10 given in Section 3.1, then number of failed tool type j at the end of stage i *will be* $(\acute{H}_{ij} - 1)$*;* but, if the remaining capacity of tool type j is zero $\acute{F}_{ij} = 0$, then it will be $\acute{H}_{ij}$.

3.5.4. Mating Pool Selection Strategy

To create the new generation, it is necessary to select some chromosomes (mating pool) based on their fitness function values in the current generation for recombining or creating chromosomes related to the new generation. In this case, a normalized fitness strategy is used, in which the fitness of the current generation chromosomes is first normalized according to Eq. (22), then the chromosomes, which their normalized fitness is less or equal to zero, are selected as a mating pool.

$$Z_i = \frac{f_i - \mu}{\delta}, i = 1, 2, \dots, k, \tag{22}$$

where Z_i is the normalized fitness of chromosome i and f_i is the fitness of chromosome i μ and δ are the mean and the standard deviation of the chromosomes fitness values in the current generation, respectively.

3.5.5. Operators of the Proposed GA

As mentioned before, to generate a new solution only arrays of matrices X and Y are changed and other matrices are changed by them, thus the GA operator is exercised only over matrices X and Y. Matrix Y is a linear one so operators (i.e., mutation or crossover) are performed on it by using a traditional method. However, the structure of matrix X is formed as a non-linear one. Thus, the GA linear operators cannot be used in a non-linear matrix type as the traditional forms and these operators should be improved proportionally. So the improved operators that proposed by Tavakkoli-Moghaddam et al. [19], are used to exercise on matrix X. In each run of the GA, one of the operations is exercised over one of the matrices X and/or Y related to the current chromosome randomly in order to generate new solutions.

i. The crossover and mutation on matrix X

For the mutation operator, one X chromosome is selected from the mating pool, randomly and then a portion of the matrix is selected by a block or diagonal method, randomly. Then, randomly change the array of the selected portion while considering the feasibility. Figs. 1 and 2 denote the mutation operator over matrix X.

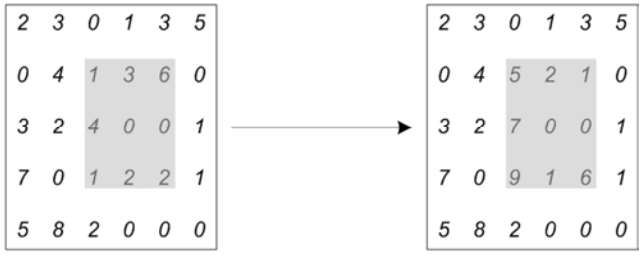

Fig. 1. Mutation operator over matrix X, block method.

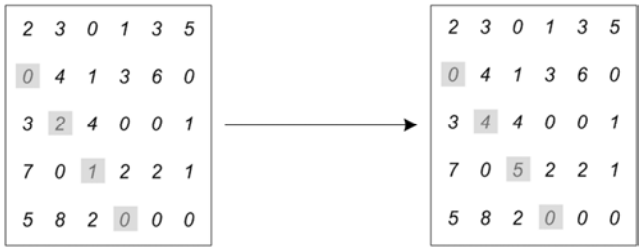

Fig. 2. *Mutation operator over matrix X, diagonal method.*

For the crossover operator, two X chromosomes, which called as parents, are selected, randomly, from the mating pool and A portion of the matrix (i.e., block or diagonal method) is selected, randomly, Then in order to create a new chromosome the two portions are swapped. Fig. 3 denotes the crossover operator over matrix X.

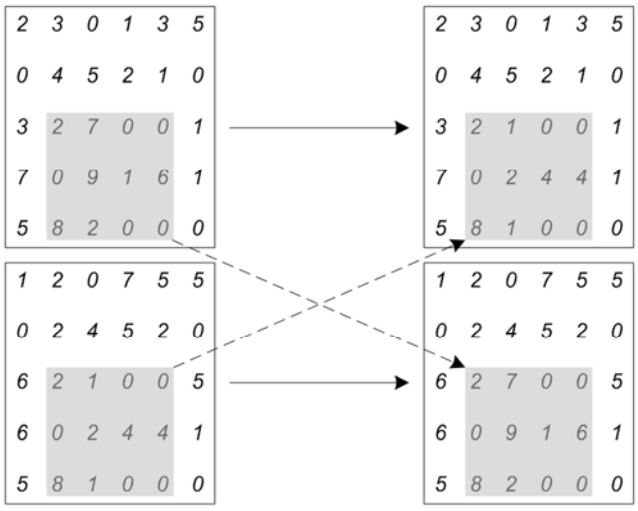

Fig. 3. *Crossover operator over matrix X, block method.*

ii. The crossover and mutation on matrix Y

For crossover on Y, the well-known order crossover, named as OX is used. For the purpose of mutation on Y, the block neighborhood for the tool switching problem proposed by Al Fawzan and Al Sultan [16], is considered. It is based on swapping the whole segments of contiguous positions. The

resulting mutation operator is called the random block insertion (RBI) and works as follows:

1. A block length $b_l \in N_{n/2}$ is uniformly selected at random.
2. The starting point of the block $b_s \in N_{n-2b_l}$ is subsequently selected at random.
3. Finally, an insertion point b_i is selected, such that $b_s + b_l \leq b_i \leq n - b_l$ and the segments

(b_s, b_s+b_l) and (b_i, b_i+b_l) are swapped.

3.5.6. Stopping Criteria

This paper, the following stopping criteria are considered.

1. Number of generations: In this case, the algorithm terminates if the number of generations exceeds the specific number.
2. Time interval: In this case, the algorithm terminates if the difference between the current time and the achievement time to the best solution exceeds the specific time interval.

4. Computational Results

In this section, an application of the proposed model is presented and also the performance of the proposed GA method is evaluated by using several examples. In this section 16 numerical examples in different small and large sizes are considered. Small-sized examples are optimally solved by a branch-and-bound (B&B) method under the Lingo 8.0 software. However, because of the complexity of such an NP-hard problem, exact methods (e.g., B&B) cannot optimally solve large-sized problems in a reasonable CPU time. Furthermore, the proposed GA algorithm is coded in order to solve both small and large-sized examples.

The parameter tuning for the GA (e.g., number of populations and number of generations) is carried out after an extensive phase of experimentation with different values. The best combinations of the parameter values are finally selected. Both the Lingo and Matlab software run on PC with two Intel® CoreTM2 T9300@ 2.5 GHz processors and 2 GB RAM. All the parameter values of the given problem described in Section 3.2 are chosen randomly. Instances main parameters are described by $C\partial_n^L|K$ that C, L, n and K denote the capacity of the magazines, the total number of operations, the number of part types and the total number of tools, respectively.

Tables 1 and 2 list the examples that show the minimum and maximum number of operations required for all part types (Rows 1 and 2) and the minimum and maximum number of substitute tools, which can be used for a specific operation, for all operations (Rows 3 and 4).

Table 1. *Explanations of small-sized examples.*

	Example definitions	$3\partial_5^8\|8$	$4\partial_5^{10}\|10$	$5\partial_5^{12}\|12$	$6\partial_5^{15}\|15$	$3\partial_6^8\|8$	$4\partial_6^{10}\|10$	$5\partial_6^{12}\|12$	$4\partial_7^8\|8$
1	Min	2	2	2	3	2	2	2	2
2	Max	3	4	5	5	3	4	4	4
3	Min	2	2	2	3	2	2	3	2
4	Max	4	5	4	6	4	6	5	4

Table 2. *Information of large-sized examples.*

		$4\partial_8^8\|8$	$5\partial_8^{10}\|10$	$5\partial_{10}^{12}\|12$	$7\partial_{12}^{15}\|15$	$9\partial_{15}^{15}\|15$	$10\partial_{20}^{18}\|18$	$10\partial_{30}^{15}\|15$	$12\partial_{40}^{30}\|30$
1	Min	2	2	2	3	3	3	3	3
2	Max	4	5	4	6	8	9	9	10
3	Min	2	2	3	2	2	2	2	2
4	Max	4	5	6	8	8	10	10	10

The examples are solved by the proposed GA and the B&B method. Tables 3 and 4 illustrate the comparative results of small and large-sized examples, respectively. In these tables, columns CPU, OFV, NOG, NOP, MCPUT and BOFV denote the CPU run time of Lingo, the objective function value of Lingo, the number of generations of GA, the number of populations of GA, the mean CPU run time of GA and the best objective function value of GA in all iterations, respectively. Data in the GAP column are obtained as follows:

$$GAP = \left(\frac{BOFV - OFV}{OFV}\right) \times 100 \quad (23)$$

In Table 4, only in three out of eight instances, the B&B method is able to access to the feasible space within six hours, and the related results of these instances are not as good as the proposed GA and the average of the Gap between the GA and the B&B method is computed about 2%. By considering the CPU run time, it shows that the performance of the proposed GA is promising.

Table 3. *Comparison between B&B and GA runs for small-sized instances.*

No.	Problem information							
		CPU	OFV	NOG	NOP	MCPUT	BOFV	GAP (%)
1	$3\partial_5^8\|8$	230	115	200	70	12.5	115	0
2	$4\partial_5^{10}\|10$	637	241.5	200	70	15.2	241.5	0
3	$5\partial_5^{12}\|12$	1180	223.5	200	80	21.1	233.5	4
4	$6\partial_5^{15}\|15$	3581	271	200	80	23.4	281	3.7
5	$3\partial_6^8\|8$	6844	164.5	200	80	15.2	164.5	0
6	$4\partial_6^{10}\|10$	12990	328	300	90	29.3	357	8.8
7	$5\partial_6^{12}\|12$	12708	266	300	90	32.1	266.5	0.2
8	$4\partial_7^8\|8$	19534	395	400	90	40.8	395	0

Table 4. *Comparison between B&B and GA runs for large-sized instances.*

No.	Problem information							
		CPU	OFV	NOG	NOP	MCPUT	BOFV	GAP (%)
1	$4\,\partial_{8}^{8}\|8$	21600	415*	500	90	53.8	395	-
2	$5\,\partial_{8}^{10}\|10$	21600	430*	500	90	68.9	380	-
3	$5\,\partial_{10}^{12}\|12$	21600	680*	600	100	134	510	-
4	$7\,\partial_{12}^{15}\|15$	21600	-	700	100	240.4	810	-
5	$9\,\partial_{15}^{15}\|15$	21600	-	700	100	440.6	740	-
6	$10\,\partial_{20}^{18}\|18$	21600	-	800	110	853.6	660	-
7	$10\,\partial_{30}^{15}\|15$	21600	-	900	120	1376	830	-
8	$12\,\partial_{40}^{30}\|30$	21600	-	1000	130	2744	710	-

*Best solution found after 6 hours

5. Conclusion

This paper has considered the problem of tool switching in flexible manufacturing systems (FMS), which is a well-known problem in operations research. Indeed, A new version of tool switching is proposed. In addition, A tool life for each tool is considered. The objective of the problem was to minimize the total part tardiness and tool purchasing costs. Also the formulation of the problem is proposed; however, because of the complexity of such an NP-hard problem, it could not be used any exact method (e.g., branch-and-bound (B&B)) in order to solve large-sized problems. Therefore, an adopted meta-heuristic method (GA), is proposed, to solve the given problems. At last results that are obtained by the GA and the B&B method are compared to small-sized problems. The obtained results show a relative gap about 2% between the proposed GA and the B&B method in terms of objective function values.

References

[1] Belady, L. A study of replacement algorithms for virtual storage computers, IBM Systems Journal., 1966, 5, 78–101.

[2] El Maraghy, H. A. Automated tool management in flexible manufacturing, Journal of Manufacturing Systems., 1985, 4 (1), 1–14.

[3] Kiran, A., and Krason, R. Automated tooling in a flexible manufacturing system, Industrial Engineering., 1988, 20, 52–57.

[4] Tang, C. S., and Denardo, E. V. Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches, Oper. Res., 1988, 36 (5), 767–777.

[5] Bard, J. F. A heuristic for minimizing number of tool switches on a flexible machine. IIE Transactions., 1988, 20 (4), 382–391.

[6] Djellab, H., and Djellab, K., and Gourgand, M. A new heuristic based on a hypergraph representation for the tool switching problem, International Journal of Production Economics., 2000, 64 (1-3), 165–176.

[7] Hertz, A., and Laporte, G., and Mittaz, M., and Stecke, K. Heuristics for minimizing tool switches when scheduling part types on a flexible machine, IIE Transactions.,1988 30, 689–694.

[8] Laporte, G., and Salazar Gonzalez, J., and Semet, F. Exact algorithms for the job sequencing and tool switching problem, IIE Transactions., 2004, 36 (1), 37–45.

[9] Oerlemans, A. Production planning for flexible manufacturing systems, PhD Dissertation, University of Limburg, Maastricht, Limburg, Netherlands, 1992.

[10] Crama, Y., and Kolen, A., and Oerlemans, A., and Spieksma, F. Minimizing the number of tool switches on a flexible machine, Int. J. of Flexible Manufacturing Systems., 1994, 6, 33–54.

[11] Sarin, SC., and Chen, CS. The machine loading and tool allocation problem in a flexible manufacturing system. Int J Prod Res., 1987, 25 (7), 1081–94.

[12] Buyurgan, N., and Saygin, C., and Engin Kilic, S. Tool allocation in flexible manufacturing systems with tool alternatives, Robotics and Computer-Integrated Manufacturing., 2004, 20, 341–349.

[13] Salonen, K., and Raduly Baka, C., and Nevalainen, O. S. A note on the tool switching problem of a flexible machine, Computers & Industrial Engineering., 2006, 50(4), 458–465.

[14] Zhou, B. H., and Xi1, L. F., and Cao, Y. S. A beam-search-based algorithm for the tool switching problem on a flexible machine, Int. J. of Advanced Manufacturing Technology., 2005, 25 (9-10), 876–882.

[15] Hertz, A., and Widmer, M. An improved tabu search approach for solving the job shop scheduling problem with tooling constraints, Discrete Applied Mathematics., 1993, 65, 319–345.

[16] Al Fawzan, M. A, and Al Sultan, K. S. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine, Computers & Industrial Engineering., 2003, 44 (1), 35–47.

[17] Amaya, E., and Cotta, C., and Fern´andez, A. J. A memetic algorithm for the tool switching problem, in: Proc. Of the Int, Workshop on Hybrid Metaheuristics, Málaga, Sapin, 2008, Lecture Notes in Computer Science (LNCS), Springer-Verlag., 2008, 5, 190–202.

[18] Konak, K., and Kulturel Konak, S., and Azizoglu, M. Minimizing the number of tool switching instants in flexible manufacturing systems, Int. J. of Production Economics., 2008, 116, 298–307.

[19] Tavakkoli-Moghaddam, R., and Aryanezhad, M. B., and Safaei, N., and Azaron, A. Solving a dynamic cell formation problem using Metaheuristics, Applied Mathematics and Computation., 2005, 170, 761–780.