Global Journal of Mathematics and Statistics
ISSN: 3078-8927
DOI: 10.61424/gjms
Journal Homepage: www.bluemarkpublishers.com/index.php/GJMS

GJMS
BLUEMARK PUBLISHERS

| RESEARCH ARTICLE

Using a Generic Decision Support Tool for Solving Job-shop Scheduling Problem

Mustafa Talal Kadhim
Ministry of Education\ Babil Education Directorate
Corresponding Author: Mustafa Talal Kadhim, **E-mail**: mostafat2000k@gmail.com

| ABSTRACT

This paper presents a broad and modular decision assistance tool designed to address a range of scheduling, assignment, and planning issues. Solving a real-world multi-period workshop scheduling challenge put forth by a case study firm that manufactures refrigerated food service equipment serves as an example of how to use this tool. The issue facing the case study firm is discussed, along with a listing algorithm that was created to incorporate the suggested solution for this particular issue. According to preliminary findings, the company's issue can be successfully resolved with the help of the suggested technology. The suggested tool has already been used to address two more issues in addition to the one this study describes. Therefore, it has been shown that the suggested tool may be readily modified to accommodate a variety of planning or scheduling issues, so circumventing the lack of adaptability typically linked to the more specialized approaches recommended in the literature.

| KEYWORDS

Meta-heuristic approaches, Batch size, Performance context

| ARTICLE INFORMATION

ACCEPTED: 19 May 2025 **PUBLISHED:** 27 July 2025 **DOI:** 10.61424/gjms.v2.i1.373

1. Introduction

Over the past forty years, scheduling theory research has developed and been the focus of a large body of significant literature. Techniques used in this research range from crude dispatching rules to sophisticated branch-and-bound algorithms, and bottleneck point-based foundations of wisdom. In business, a lot of scheduling issues come up. Although they are quite easy to formulate[1], usually only requiring figuring out the sequence and resource limitations[2], solving them is nonetheless very difficult. Although the different scheduling factors have been thoroughly examined in the literature, the majority of writers still suggest approaches that are unique to the problem and inapplicable to other scheduling variables. No solution strategy with assured performance has been established to date, as mentioned in [1]. According to the same authors, there are situations in which the majority of approximation approaches do not work, and it is not totally clear what circumstances make a certain approach more or less likely to work. A broad decision support tool designed to address a range of planning issues is presented in this study. The suggested tool is a cross between a list-based algorithm and a metaheuristic algorithm. No matter what problem is being addressed, the metaheuristic method may be applied without modification. The list-based method has to be adjusted based on the issue under investigation. A description of the decision assistance tool has been provided. Two distinct planning and scheduling problems have already been used to test the described decision support tool: (1) the problem of allocating resources and planning activities in the context of a multi-site hospital, and (2) the problem of scheduling and booking containers with setups and deadlines for a plastic injection company. The suggested method had positive outcomes in both situations. This paper's primary

goal is to make clear how the suggested tool adjusts to a novel problem that arises in a case study business that manufactures refrigerated food service equipment. This adaptation entails creating a general algorithm that is suited to the particular situation at hand by using the tool's basis, which was created for earlier issues that were resolved. The objective is to show that the suggested method is broad in the sense that it requires little development work to help the decision-making process for a wide range of planning and scheduling issues. The structure of the paper is as follows: A description of the case study firm and the issue that has to be resolved are given at the start of Section 2. After that, a succinct summary of the literature on scheduling shops—the kind of issue the case study firm was facing—is given. With an emphasis on the built list algorithm for the particular production planning example covered in this article, Section 3 outlines the suggested tool to address the case study company's issue. The capacity of the suggested tool to handle the planning issue of the case study firm is examined in Section 4 based on the outcomes of a test case. Lastly, some findings are presented in Section 5.

1.1 Problem description

i- The case study company

The firm under investigation produces refrigerated food service equipment, including freezers, cold rooms, cabinets, countertops, and air coolers. Doors, shelves, a cooling unit, and an external framework make up the majority of the items. Most clients will ask for a fully customized item, while others may buy the company's conventional goods. As a result, the equipment will either be made individually or in limited quantities. The two primary divisions of the company's plant are assembly and components manufacture. In order to create the finished product, the assembly section consists of two assembly lines that combine internal components like shelves, drawers, and baskets that are obtained from vendors with the external body, doors, and cooling unit—all of which are created internally. The construction of the parts and components of the doors are made from stainless steel sheets in the parts manufacturing department using cutting equipment and drilling systems. The company's planner creates the assembly line production schedule based on verified client orders. The components manufacturing department creates a weekly order file with a list of the metal parts needed to feed the assembly lines (see Table 1 for an example). Therefore, a list of the tasks to be created, comprising the following details, is represented by the weekly demand for the parts needed to supply the assembly lines:

The amount to be produced each day of the week;

• The part to be produced;

• Their priorities:

(a) The part must be produced on the necessary day,

(b) The part may be delayed by a maximum of one day, and

(c) The part may be delayed by a maximum of two days; The order in which each function is performed, accounting for the machine and tool utilized in each operation, as well as the processing time in seconds for each unit.

Table 1. Examples of DD assignment rules.

							Operation 1			Operation 2			Operation 3		
Part	Priority	Mon.	Tue.	Wed.	Thu.	Fri.	Mach.	Tool	PT	Mach.	Tool	PT	Mach.	Tool	PT
xxx	a		6				M1	T11	46						
yyy	b	2		7	5		M2	T22	3	M3	T32	2	M4	T43	43
zzz	c		18	20	16	32	M1	T11	9	M4	T42	7			
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

The following is a description of the scheduling issue in the production department of the firm under study. To be created, each component must adhere to a processing sequence, and each step in this sequence calls for a certain set of resources (tool and machine). Five intervals, each lasting up to eight hours, make up the planning horizon,

which spans a week. Throughout the planning horizon, a variety of component batches must be generated daily to satisfy the assembly line's needs. Determining how to carry out a set of N jobs that must be completed on a set of M machines is the challenge. A series of actions connected to a particular machine/tool combination defines each task. Every operation has a processing time, and depending on the order, there may be a setup time in between processing two consecutive operations. Every work needs to be completed in the allotted time (day). Two components of a penalty function are taken into consideration: (1) storage cost (early start) in the event that the work is completed before the deadline, and (2) delay cost in the event that the task is completed after the deadline. Finding the order in which each machine operates in order to reduce the overall penalty is the aim. Six cutting/drilling machines make up the components manufacturing division. Each function might have anything from one to three operations. To create any particular process, a collection of equipment and tools is needed. Each machine has its own set of tools since they are not shared by other machines. An example of the setup times (in seconds) needed for a particular machine to produce two consecutive functions is shown in Table 2.

Table 2. ANN based DD assignment rule input data set.

Machine M1	T11	T12	T13	T14
T11	72	360	300	300
T12	360	72	300	300
T13	300	300	30	240
T14	300	300	240	30

Two significant aspects of the topic under study are evident from Table 2. (1) Even if two processes use the same tool, there is always a setup time when switching between them, and (2) setup periods are significantly longer than processing times.

2. Literature review

One may categorize the issue discussed in the preceding section as a multi-stage workshop scheduling issue. T workshop scheduling challenges that must be resolved one after the other make up a multi-stage workshop scheduling problem.

The endeavor to complete each period of the specified planning horizon in less time than the capability that is available. Batch size and scheduling challenges developed in [5] are strongly connected to the multi-period job shop scheduling problem. Among scheduling optimization problems, the job shop scheduling problem (JSSP) is regarded as one of the most difficult ones [6]. Because of its many real-world uses, JSSP has been the subject of much research in recent decades. As far as we are aware, however, the only multi-period job shop scheduling issue that shares the same features (goals and constraints) as the one discussed in this study is the one shown in [5]. [1] presents a general taxonomy of JSSP approaches. The authors' taxonomy separates JSSP approaches into two primary categories: optimization techniques and approximation techniques. Branch-and-bound algorithms, as shown in [6, 7], or mathematical optimization approaches, such linear or mixed-integer programming, as seen in [8], are the mainstays of optimization techniques. The application of optimization techniques has been restricted to small-sized situations because of the algebraic task scheduling problem's difficulty [9]. The algebraic task scheduling problem has also been solved through the use of approximation techniques. These methods fall into two categories: general algorithms like genetic algorithms [11], local search methods [10], or, more recently, artificial neural networks [12], and distribution rules and quick strategies. The issue of scheduling concurrent jobs has also been resolved through the use of approximation techniques. These methods fall into two categories: general algorithms like local search and meta-heuristic approaches, evolutionary algorithms, or, more recently, artificial neural networks, and dispatching rules and heuristics. Modern technologies include algorithms [11] and, more recently, artificial neural networks [12]. The multi-period JSSP, which involves solving a sequence of JSSPs

consecutively, seems to be even more difficult than the multi-objective resource scheduling problem (JSSP), which is considered a difficult optimization issue. In [1], an intriguing overview of methods for solving JSSP is provided. The following succinctly describes the primary findings of this paper: The majority of researchers have turned their attention to approximation strategies since optimum processes often seem inappropriate for JSSP. Priority rules are widely used in distribution because they are simple to use and require little processing power. They are typically employed as a preliminary solution method rather than a comprehensive solution strategy for job scheduling problems, nevertheless, because they are outperformed by other strategies. Neural networks and other artificial intelligence approaches are only appropriate in small-scale situations due to their high calculation time requirements. The best approximation techniques have been shown to be local search strategies and meta-heuristics, particularly as the problem's dimensions increase. There is a need for more adaptable/general ways because no solution strategy that ensures performance has been discovered thus far.

2.1 The proposed tool

Concept

This study's suggested tool combines a list algorithm with a metaheuristic algorithm. Figure 1 illustrates the method's basic idea. The metaheuristic uses a list Y of tasks as its coding. Since random descent, a single solution metaheuristic, was applied in this project, the neighborhood system is a task arrangement. The list algorithm schedules the tasks in the order specified in the list and assigns them to the necessary resources based on a number of restrictions. This creates solution X. The evaluation of solution X determines the objective function H. The metaheuristic either chooses or rejects the answer based on this assessment. The optimal list of tasks, which uses the list method to maximize the objective function, is the solution that the combination offers at the conclusion of execution.

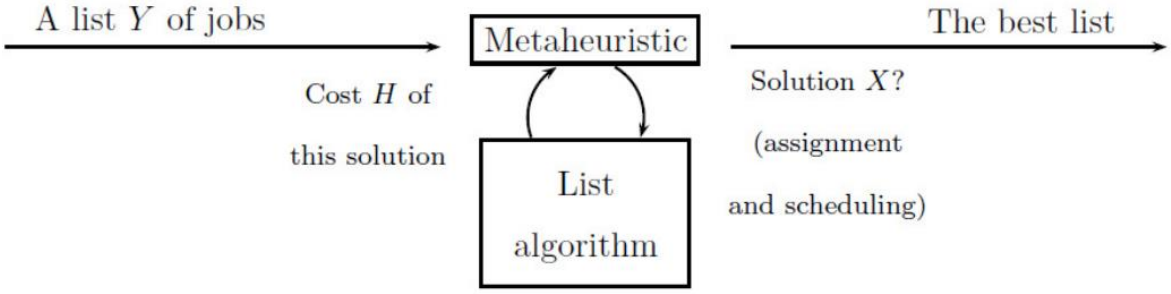

Fig. 1. Hybridization metaheuristic – list algorithm

The proposed tool: Metaheuristic

Random regression, a single-solution exploratory technique, was selected for this project. Among the first exploratory algorithms is random regression. Its basic idea is to use the neighborhood system V to determine the solution Y' that is next to the present solution Y. If the value of the objective function H(X') is less than or equal to H(X) after using the list algorithm L, X = L(Y), then the solution X' and therefore the list Y' are acceptable. Convergence of random regression to the local minimum occurs. This method was selected since the authors have already used it in other projects and because it is simple to use, which expedites the tool development process. Rapid findings were needed to evaluate the suggested tool's capacity to address the firm under study's scheduling issue. The current project development described in this work is intended to include the coding of more effective exploratory algorithms.

The proposed tool: List algorithm

List scheduling techniques are frequently used to define schedules and work in a single cycle. By allocating each task in the designated sequence to the first machine to become idle, the conventional list scheduling method generates a schedule. Working with a list algorithm is crucial because the metaheuristic uses a neighborhood

system—an arrangement between two list elements—to explore a collection of solutions. In order to assign work to machines and dates, the algorithm must take into account the list's order. The two phases of the list algorithm created for the case study problem are described in pseudocode below. Notably, the list of jobs/parts that the suggested algorithm takes into account is comparable to the one shown in Table 1.

First step:

For all jobs in the list

Assign the given operation (i) to the necessary machine (m) and the production day (d)

For each operation of the specified job (j), provided that operation i is the first operation in the job.

The start time of task j's operation 1 is equal to machine m's release date on day d. Job j's operation 1 end time equals the operation's start time. 1 of job j plus operation processing time Job J + setup time = 1. Machine m's release date is equal to the moment when task j's first operation ends.

Else

Operation I's start time for task j = max[Machine m's release date for day d; operation i-1's completion time for task j]

Process I's finish time for job j is equal to its start time plus its processing time plus its preparation time.

The end time of process I for task j is equal to the machine release date M.

It is possible to observe that the algorithm disregards the capacity-related limitations in this initial stage. Every job is allocated to the machine that is needed on the day it is supposed to be finished. Following the first phase, the algorithm carries out a second step that is explained below in order to resolve this issue and modify the schedule in accordance with capacity requirements.

Second step:

For all days

On day D, the machine's capability is being breached.

Determine which of function j's processes (i) result in a capacity violation.

For the process from 1 to i

For day d-1 to day 1

If the chosen method is feasible given the available capacity on the designated day.

Transfer the operation to the chosen day.

Update the chosen day's and day D's schedules.

From operation I to job J's final operation

Day D+1 to Day 6

If there is enough capacity on the chosen day to complete the chosen operation

Transfer the operation to the chosen day.

Update the chosen day's schedule

Update the day's schedule.

Please be aware that the algorithm takes priority constraints into account when determining whether the necessary capacity is available in a given machine on a given day for a given operation transfer. It also looks at whether the capacity can be used by the transferred operation. An illustration of the second phase of the suggested method is shown in Figure 2. Following the algorithm's initial phase, the schedule displayed in Figure 2 (a) was produced. This schedule shows a number of functions (shown in gray) where the maximum amount of processing capacity can be achieved. The capacity restriction for day 3 is broken by operations 2 and 3 of function j, which are indicated in white.

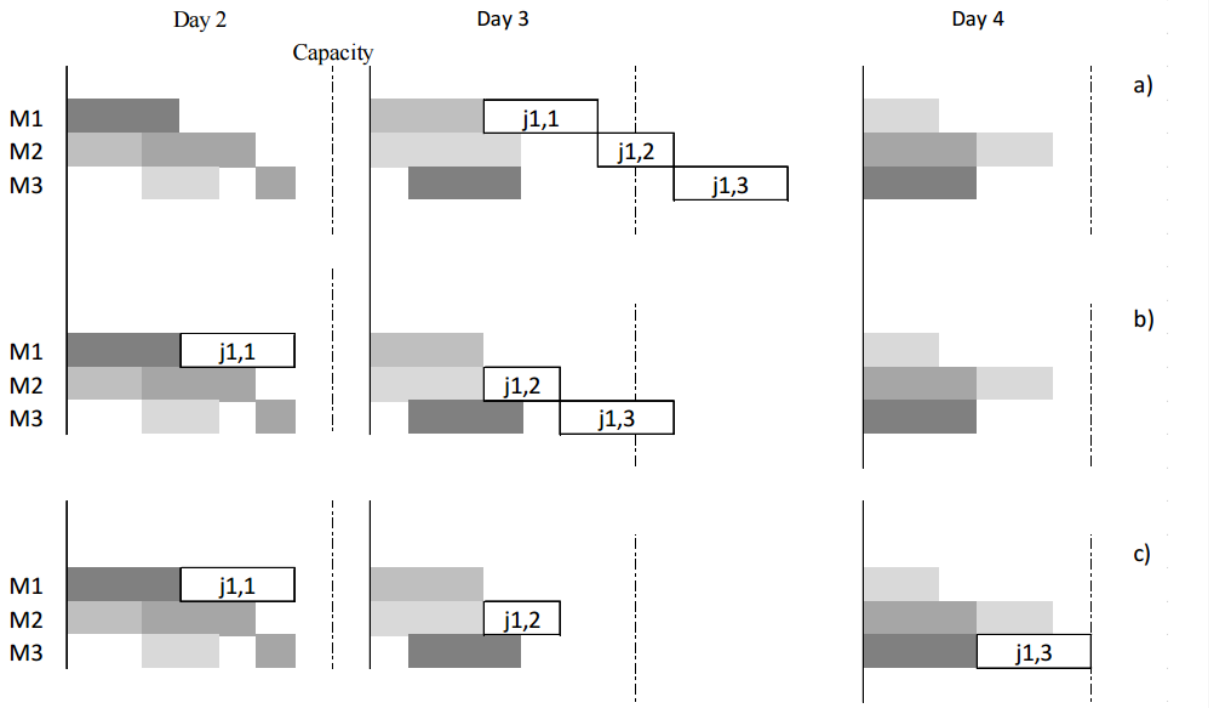

Fig. 2. Example of the application of step 2 from the list algorithm

When machine 1 has capacity, operation 1 from task C, which is processed on machine 1, can be pushed to day 2. The schedule shown in Figure 2 (b) represents the outcome of this transfer. The problem with operation 2 from job C has been fixed as a result of this transfer, however operation 3 is still above the capacity limit. As a result, operation 3 from job C is rescheduled for day 4, when machine 3 has the capability to process it. It is crucial to remember that although the algorithm takes into account a six-period planning horizon, the real planning horizon only has five periods. Operations allocated to period 6 jobs will be deemed ineffective and will result in severe penalties.

• Early completion: 0.5 × Number of work units × Number of projected days;

• Delay: Priority indicator × Number of work units × Number of delayed days;

• Tasks that are not productive: 15 × Work units

Priority jobs (A) have a priority index of 3, priority jobs (B) have a priority index of 2, and priority jobs (C) have a priority index of 1. A priority (A) work, for instance, will contribute to the objective function with a penalty of 30 units since it reflects the manufacturing of 5 units of a certain part that has been delayed for 2 days.

3. Results

Modifying and testing a generic decision support tool for a novel topic—the scheduling challenge in a multi-period workshop suggested by a study company—was the goal of this effort, as was previously stated. According to the company's historical production statistics, the weekly production scheduling issue it faces—described in section 2—considers, on average, 450 procedures and 300 requests. It was determined to generate a smaller instance in order to prevent needless complexity during the tool's development and testing. One of the six machines that comprise the manufacturing floor can handle the 49 jobs that make up the created case, each of which has one to three procedures. There are 73 operations in all that need to be processed. The roles were created using actual data from the organization under study, along with the operational features that go along with them. For each time, the capacity restriction was modified because the created case was smaller than the actual corporate issue. The capacity restriction was set at 5760 seconds and was regarded as fixed for the five recognized periods. The tool's output is shown as a list of jobs and processes that need to be completed each day of the week. It includes the following information: job number, process number, machine where processing will occur, tool utilized, and start and end times for each job process. The procedures that are highlighted in this list are either expected (produced prior to the requisite working day) or delayed (produced after the required day). We use 1000 iterations of the suggested tool to resolve the test case problem. In less than ten minutes, results are obtained. Five iterations of the test were conducted. The findings for every test are shown in Table 3. Every task for every solution is completed within a five-day planning horizon. The projected number of operations (5 in the worst case) or delayed operations (3 in the worst case) is little, and the average total penalty obtained is 210 units. Every time an iteration was completed, the solution was noted. The development of the answers found in the first 500 iterations of test 1 is depicted in Figure 3. It is evident that over 97% of the solutions—all of which carry a penalty greater than 250—are impractical, delaying activities until period 6. This demonstrates that there are few workable choices and that the capacity constraint is strict. It can be inferred that the tool can be utilized to solve the problem put forth by the case study company because it consistently finds a workable answer.

Table 3. Results obtained for the considered instance.

Test	Penalty	Nº of anticipated operations	Nº of delayed operations
1	207	4	2
2	214	2	1
3	211	5	1
4	203,5	3	3
5	209,5	3	2

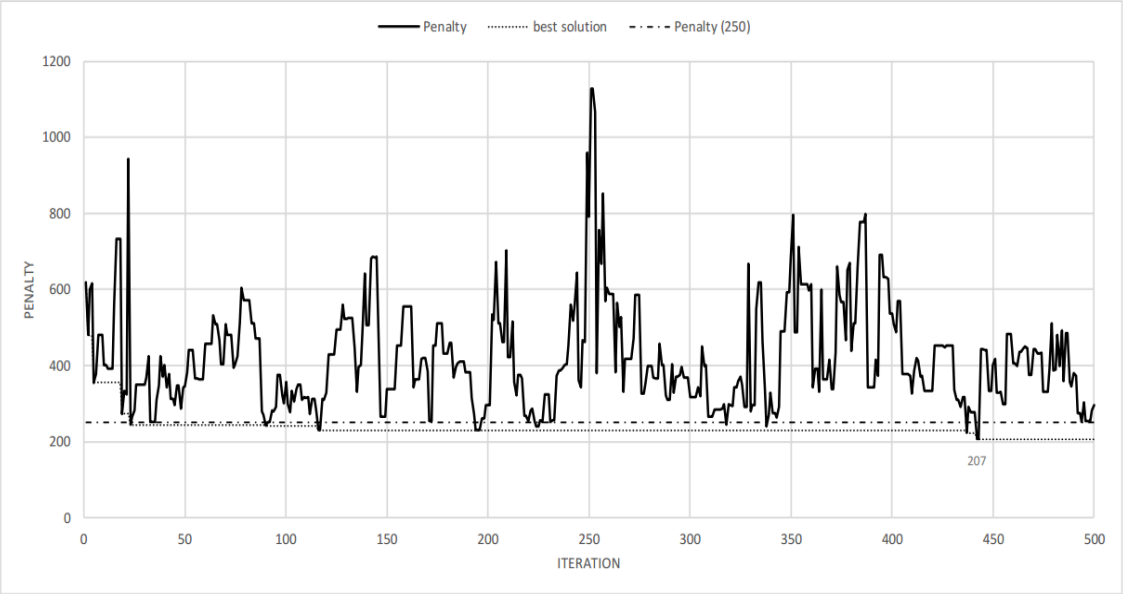

Fig. 3. Solutions obtained during the iterative process.

4. Conclusion

A broad decision support tool that may be applied to a wide range of scenarios is presented in this paper. A general reusable module for a wide range of problems and a specialized module with an existing algorithm that needs to be defined for each problem make up the suggested tool. This paper uses a real-world multi-period workshop scheduling challenge that was presented by a current study firm to demonstrate the use of this tool.

The suggested method has recently been used to address two distinct issues: the batch size and scheduling issue with settings and due dates for a plastic injection case [2], and the planning activities and resource allocation in a multi-site hospital [1]. The suggested tool's ability to handle many challenges is demonstrated by the new problem it solved in this study. Since the tool's general components can be reused, little development work is required to adapt it to new issues. Although highly specialized approaches for a given problem could be ideal for attaining high performance, they are challenging to modify for different scheduling and planning issues. The tool described in this study has the advantage of being more broad in its approach, which makes it easy to adjust to a variety of variables in planning/scheduling situations without sacrificing performance context.

References

[1] Jaiin, A.S., Memeran, S. (1998). Technical Report. University of Dundee, United Kingdom.
[2] Grimses, D., Hebrrard, E., Informs J. Comput., 27 (2015) 268-284.
[3] Klemdent, N., Godurgand, M., Gravngeon, N. (2017). 10th International Conference on Health Informatics.
[4] Silva, C., Kilement, N., Gibharu, O. (2016). in: International Joint Conference - CIO-ICIEOM-IIE-AIM, San Sebastian, Spain, ICIEOM.
[5] Daduzère-Pérès, S., Lassedrre, J.B. Eur. J. Opfer. Res. 75 (1994) 413-426.
[6] Nowwicki, E., Smuntsnicki, C. Manage. Sci. 42 (1996) 797-813.
[7] Perrdegaard, M., Clausen, J. Ann. Odper. Res. 83 (1998) 137-160.
[8] Asanio, M., Ohtda, H. Comput. Ind. Eng. 42 (2002) 137-147.
[9] Harjunfgkoski, I., Jain, V., Grosdfsmann, I.E. Comfgput. Chem. Eng. 24 (2000) 337-343.
[10] Noior, S. (2007). PhD Thesis. University of Bradford. UK
[11] Vain Laarfhoven, P., Aarsts, E., Lenestra, J. Oper. Res. 40 (1992) 113-125.
[12] Kumar, P., Kumar, P., Bhool, R., Upneja, V. (2016). Int. J. Sci. Eng. Tech. Res. 5 1439-1447.
[13] Meieran, S. (2003). in: Proceedings of 19th international conference on CAD/CAM robotics and factory of the future. Kuala Lumpur, Malaysia. 2003.
[14] Zhiu, X., Wilheilm, W.E. IIE Trians. 38 (2006) 987–1007.