Abstract Number: 025-0655

A Meta-Heuristic Optimization for Scheduling Heat-Treatment Furnace in Steel Casting Industry[1,2]

[1]M. Ramasubramaniam Muthurathinasabapathy,
Centre for Logistics and Supply Chain Management,
Loyola Institute of Business Administration,
Chennai – 600 034.
E-mail: rams@liba.edu.

[2]M. Mathirajan,
Department of Management Studies,
Indian Institute of Science,
Bangalore – 560 012.
E-mail: msdmathi@mgmt.iisc.ernet.in

POMS 23rd Annual Conference
Chicago, Illinois, U.S.A.
April 20 to April 23, 2011

Abstract: This paper addresses a new research problem of scheduling, observed in steel casting industry, related to heat-treatment furnace, to minimize the maximum completion time of all castings. Due to computational intractability, a few greedy heuristic methods and Genetic Algorithm are proposed. A series of computational experiments conducted indicate that the GA implemented has excellent average performance in comparison with a lower-bound.

Keywords: Scheduling, steel casting industry, heat-treatment furnace, makespan, greedy heuristic, GA

1. Introduction:

This paper addresses a new research problem of scheduling a special BP, the application of which could be found in Heat-treatment furnace (HTF) which is used in the post casting stage of steel casting industry. A BP, in this paper, is one that can process several jobs simultaneously subject to capacity restrictions and in a way that the jobs have a common starting and ending time. This method of scheduling batches plays an important role in variety of industries. The literature related to BP addresses these problems as scheduling of batch processing machine.

Heat treatment furnace operations are used generally to improvise the structural properties of steel castings which are subjected to stressful mechanical and thermal loads. Although there are many types of heat treatment operations based on the type of castings, it is worthwhile to observe that this is the only operation in the entire steel casting manufacturing process which takes a significant part of the total processing time. The processing times can vary from a few hours to a few days depending on the type of heat treatment operations. So, from a process flow perspective, the heat treatment stage could be termed as a bottleneck. Generally, the system throughput increases if the capacity of the bottleneck is increased or utilized properly (Pinedo, 1995). In this paper we have taken the second approach. Also, since utilization is the primary objective, we have chosen the scheduling objective of makespan (Lee and Uzsoy 1999).

This research problem addressed in this paper is pertinent to medium and large scale steel casting industries which are engaged in several heat treatment operations such as normalizing, homogenizing and tempering etc. Also, the jobs which are processed using these heat treatment operations belong to different job families because technically they cannot be processed together in the same batch. Thus, these job families are called multiple incompatible job families from the point of view of processing. Also, since there are many such steel casting industries which offer such facilities, these industries have to consider the due date issues also strongly in view of the competition. Accordingly, in this paper, we address a new research problem of scheduling a BP with Incompatible Job Families (MIJF), NIJS, NIJD and NARD to minimize makespan.

The scheduling problem defined in this paper can be denoted in three-field notation: $\alpha \mid \beta \mid \gamma$ as $1 \mid$ *p-batch, multiple incompatible job family, non-identical job sizes, non-identical job-*

dimensions, non-agreeable release times and due dates | Cmax problem. We propose a set of heuristic algorithms for the research problem and a genetic algorithm with the following assumptions:

- There are n jobs (n single castings) to be processed and these jobs are from a single job-family. Each job has a size s_i (in terms of weight) with dimensions l_i - length, w_i – width and h_i - height
- All jobs must pass through the BP
- There is a single BP, which has capacity limit in terms of size S (in terms of weight) and dimensions L - Length, W - Width, and H - Height
- The BP is available continuously
- Orientation of jobs is not allowed
- All data of $S, L, W, H, s_i, l_i, w_i,$ and h_i are deterministic and known a priori
- It is assumed that $s_i \leq S$; $l_i \leq L$; $w_i \leq W$; and $h_i \leq H$ for all jobs i
- Jobs are assumed to have non-agreeable release times and due dates. i.e . $r_i \leq r_j$ does not imply $d_i \leq d_j$
- Processing time of job family is assumed to be independent of number of jobs in the batch and constant
- Preparation time, if any, is included in the processing time, and there is no setup time in switching from one batch to another
- Once processing of a batch is initiated, the BP cannot be interrupted and other jobs cannot be introduced into the BP until the current processing is completed.
- Machine breakdowns are not considered

The paper is organized as follows: In section 2, we review the related work. In section 3, we propose a Greedy Heuristic Algorithms (GHA) and a Genetic Algorithm (GA). In section 4, we propose a lower bound (LB) for the problem. The computational experiments and its analysis are discussed in Section 5. We conclude the paper with a summary and possible future research direction in Section 6.

2. Related works:

Although there is a vast amount of literature in scheduling of BP in semiconductor industries [Mathirajan and Sivakumar (2006b)], there are only few studies which are close to the problem configuration considered in our study. Also, scheduling of heat treatment furnaces in steel casting industry has been addressed in only some studies that are close to the research problem. We review only these studies in this section.

The past research on the scheduling of BP has been primarily focusing on one prevalent application namely scheduling BP in semiconductor industries, partly because of importance and the impact of the industry on country's economy. It is also worthwhile to note that because of the computational complexities of the problems studied, majority of the researchers adopt meta-heuristic approaches to solve the problems.

Scheduling a BP with incompatible job families was studied by Koh et al. (2005) with the objectives of minimizing makespan, total completion time and total weighed completion time. They propose a integer programming model, a set of simple heuristics and a genetic algorithm for the problems. The detailed comparison with lowerbounds for the problem show varied performance of the genetic algorithm for different objectives.

An ant colony framework for scheduling a BP with arbitrary job sizes and incompatible job families to minimize completion time objective have been proposed by Kashan and Karimi (2007). After a detailed computational experiments with a number of heuristic and meta-heuristic algorithms from literature, the meta-heuristic was shown to provide good quality solutions.

Nong et al.(2008) studied a bounded BP configuration with release dates and family jobs with the objective of minimizing makespan. They propose a polynomial-time approximation algorithms for the identical and non-idenical job size cases.

Recently, Damodaran et al. (2012) addressed the problem of scheduling multiple BP with unequal ready times with the objective of minimizing makespan. They propose a simulated annealing (SA) algorithm which is compared with a Modified Delay (MD) Heuristic and a Greedy Randomized Adaptive Search Procedure (GRASP). Their computational experiments reveal that the SA performs equally well in comparison to GRASP with less computational burden.

Apart from the semiconductor industry, there are only a few studies which focus on addressing the problem of scheduling heat treatment furnaces ((Mathirajan (2004a,2006a)). Mathirajan et al. (2004a, 2006a) consider the problem of scheduling heterogenous batch processors with incompatible job families, non-identical job sizes and dynamic job arrivals to maximize the utilization of the batch processors to minimize the total weighted tardiness respectively assuming all the jobs are having same dimensions.

In real world applications, particularly in steel casting industries, the assumption on job dimension does not hold. Therefore, in this paper the assumption on job dimension is relaxed and the resulting problem is studied.

Also, our study differs from all the above studies in that we consider the important job characteristics: NARD which has not been addressed so far.

3. Development of heuristic algorithms

The problem of scheduling a BP with a MIJF, NIJS, NIJD and NARD is NP-hard as this problem subsumes the well-known bin-packing problem. According to Garey and Johnson (1979), the bin packing problem is strongly NP-hard, and then the problem of scheduling a single BP with MIJF, NIJS, NIJD and NARD is strongly NP hard, too. Thus, we focus on developing simple heuristic algorithms for obtaining efficient solutions and also a genetic algorithm based on our observations from a couple of steel casting industries, located in Coimbatore, TamilNadu.

3.1 Greedy Heuristic Algorithm (GHA):

The GHA proposed in this paper broadly consists of three phases:

1. Sort the jobs using a criterion and identify the job family.
2. Construct the batches from the selected job family and sequence them.
3. Sequence the batches.

For simplicity, selection of a job family is assumed to be arbitrary. This type of rule would also allow the GHA to be used as a future benchmark when designing more complex heuristic algorithms. Also, this type of rule is also being used in practice where the operator of heat-treatment furnace selects the family arbitrarily to be loaded when the heat-treatment furnace becomes free.

The detailed explanation of the GHA is as follows. In the proposed GHA, first we sort the jobs and then construct a set of batches by picking jobs from the sorted list. Finally, the set of batches constructed will be sequenced. The general details of these steps are as follows:

Step 1: Sort the list of jobs waiting in front of the BP using a specific criterion.

Step 2: Select the first job family arbitrarily

Step 3: Select a set of feasible jobs from the job family, sequentially from the sorted list of jobs and construct batch satisfying capacity restrictions on size, dimension and release time constraint. For this step, we follow the following batch construction logic:

> *The basic logic of the batch construction phase of the GHA starts with creation of a new batch. A batch contains two attributes namely shelf and layer which are opened when a new batch is formed. Each job is assigned to the open shelf whose height is dictated by the maximum height of all the jobs that are in the shelf. Width of the shelf is the maximum width of all the jobs that are in the current shelf. The position of a newly assigned job in the shelf is current position of the shelf. The current position inside the shelf, for a new job, is given by x, y and z coordinates of the left bottom corner with respect to the coordinate axes X, Y and Z in three dimensional space. This would be 0, 0, 0 for a new shelf in a new batch. When no more jobs can be filled in the open shelf the current shelf is closed and a new shelf is created on the top of the closed shelf inside the layer which is currently open. So, a layer may contain one or more shelves. The layer height is the total height of all shelves in it and layer width is the maximum width among all shelves in it. When no more shelves can be created in the open layer, the current layer is closed and a new layer is opened and a new shelf created. When no more layers can be created in the existing batch, the current batch is closed and a new batch is opened and a new layer and a shelf are created.*

Step 4: If the job list is not empty, select the next job family arbitrarily in the job list and repeat step 2 - 4 until all the jobs in the job list are assigned to batches.

Step 5: Sequence the set of constructed batches using Earliest Batch Available Time (EBAT) rule, where, EBAT = maximum{*release time of all jobs in a batch*}}

Step 6: Calculate the makespan using the following equation:

Cmax = (*Number of batches resulted in the end of step 3*) x (*processing time of job family*)

The proposed GHA can be varied by introducing different criterion in step 1 as follows:

- Sort by length of the job
- Sort by width of the job
- Sort by height of the job
- Sort by volume of the job
- Sort by size of the job
- Sort by due date: minimum due date to maximum due date
- Sort by release time: minimum releasetime to maximum releasetime
- Sort by ratio (Volume/Duedate): high ratio to low ratio
- Sort by the ratio (Size/Duedate): high ratio to low ratio

The idea behind the first five sorting criteria is based on the practice followed by the industries for scheduling a HTF. The last four criteria are developed to address the characteristic of the problem 'NARD'. Taken together, we have proposed nine variants of the GHA which are uniquely termed in the study as defined in Table 1. Additionally, the flowchart of one of the GHA: SLB(NARD&MIJF) is given in Figure 1.

3.2 Genetic Algorithm:

One popular stochastic procedure for hard optimization problems based on natural selection is GA. GA has been used extensively for complex scheduling problems with good results. To the best of our knowledge, there are no studies which use GA for scheduling a BP in the steel casting industry. However, GA has been applied to scheduling BP related to the semiconductor industry (Wang and Uzsoy, 2002; Cheraghi et al., 2003; Malve and Uzsoy, 2007). This has motivated us to propose GA for scheduling a BP for the steel casting industry.

The development of *GA* requires proper representation. In this study, we use random key-based representation for *GA* (Norman and Bean, 1999). The success of GA lies in fixing certain *GA* parameters such as number of generations, population size, crossover percentage, crossover probability, migration percentage and mutation percentage, which are problem-specific. In order to fix these *GA* parameters, we conducted a pilot computational experiment and the appropriate values are fixed and presented in Table 2.

Table 1: Brief details of the variant of GHA with unique name for the variants

GHA Variant	Brief details of the variant of GHA	Unique name of the variant of GHA
1	**S**ort by **L**ength and construct **B**atches considering basic capacity constraints, Non-Agreeable Release Time And Due Dates And Additional Restrictions On Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SLB(NARD&MIJF)*
2	**S**ort by **W**idth and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SWB(NARD&MIJF)*
3	**S**ort by **H**eight and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SHB(NARD&MIJF)*
4	**S**ort by **V**olume and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SVB(NARD&MIJF)*
5	**S**ort by **S**ize and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SSB(NARD&MIJF)*
6	**S**ort by **D**ue date and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SDB(NARD&MIJF)*
7	**S**ort by (**V**olume/**D**ue date) and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SVDB(NARD&MIJF)*
8	**S**ort by (**S**ize/**D**ue date) and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SSDB(NARD&MIJF)*
9	**S**ort by **R**elease time and construct **B**atches considering basic capacity constraints, Non-Agreeable Release time and Due dates and additional restrictions on Incompatible Job Family- Sequence the constructed batches using EBAT rule	*SRB(NARD&MIJF)*

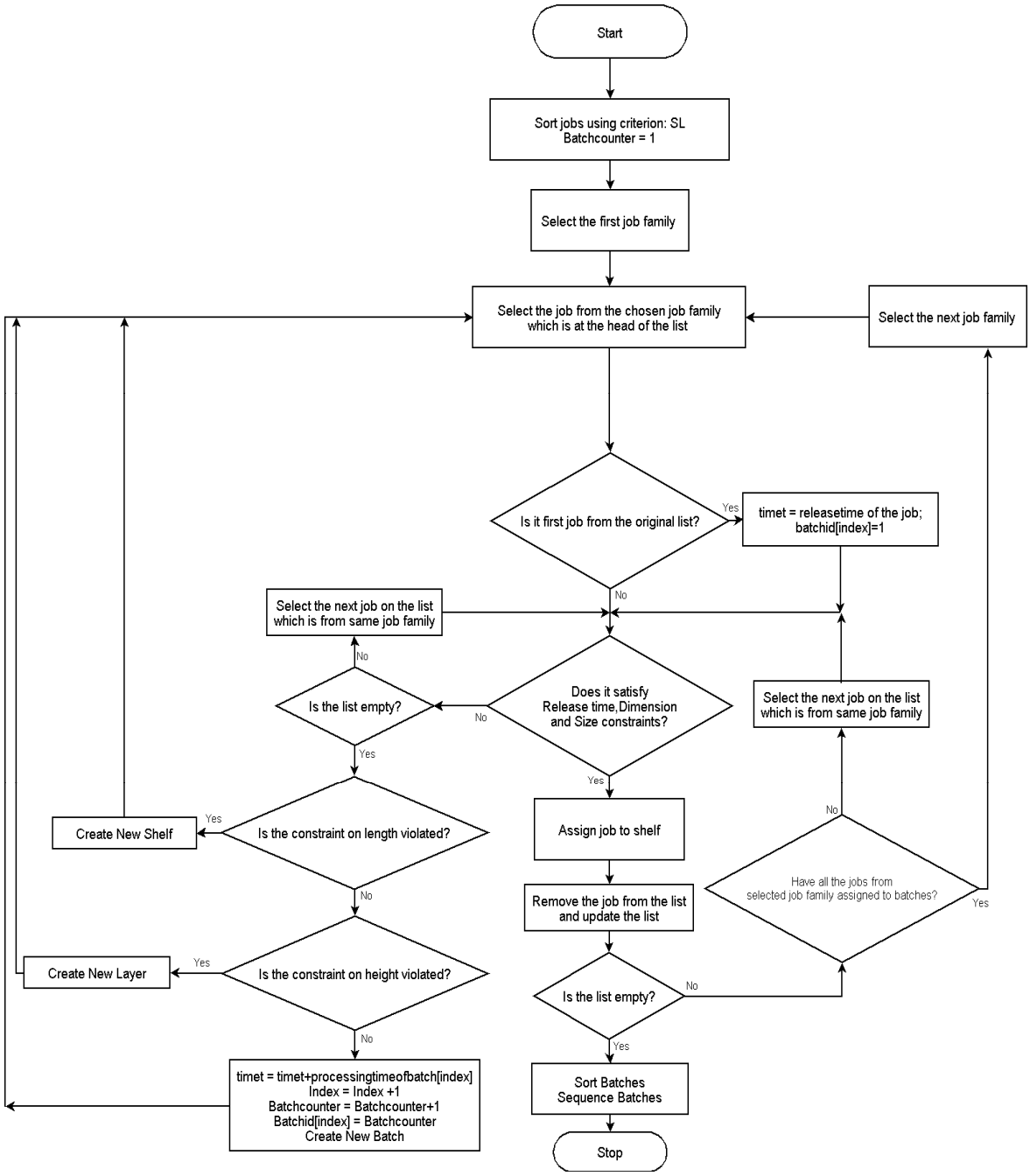

Figure 1: Flowchart for the greedy heuristic algorithm: *SLB(NARD&MIJF)*

Table 2: GA parameters and their values

***GA*-parameters**	**Values of the parameters**
Number of generations	200
Population size	25
Migration percentage	20%
Crossover percentage	50%
Mutation percentage	30%
\ Crossover probability	0.6

Using the values of the *GA* parameters as given in Table 2, the step-by-step procedures of the proposed *GA* for scheduling a BP with MIJF, NIJS, NIJD and NARD are as follows:

Step 1. Generate a population of permutation sequences with size 25. This sequence is essentially job indices numbered from 1 to n.

Step 2. Sort the first sequence based on the job length using criterion Sort by length mentioned in section 3.1 with the maximum length job being first and the minimum length job being last.

Step 3. Repeat step 2 for the next eight sequences using criteria: Sort by Width, Sort by Height, Sort by Volume, Sort by Size, Sort by Duedate, Sort bu Volume-Duedate ratio, Sort by Size-Duedate ratio, Sort by Releasetime.

Step 4. Generate random permutation sequences for the rest of the population.

Step 5. If the number of generations is less than or equal to 200, construct batches (a feasible schedule) using the GHA mentioned in section 3.1 for each of the sequence. Else, go to step 16.

Step 6. Calculate the fitness value and the makespan of the schedule for each of the sequences.

Step 7. Sort the population of sequences based on the fitness value with lowest value on the top to highest on the bottom.

Step 8. Encode the permutation sequences using random keys.

Step 9. Keep top 20% of the population for migration to next generation.

Step 10. Select two chromosomes randomly from the population and apply crossover operation.

Step 11. Repeat step 10 to generate a total of 50% of population sequences which are the new set of offsprings.

Step 12. Replace the middle 50% of the population with the new offsprings.

Step 13. Generate rest 30% of the chromosome population randomly and replace the lowest 30% of the population.

Step 14. Sort each chromosome by random keys which are generated using crossover and mutation in descending order.

Step 15. Decode the chromosomes to get a new set of permutation sequences. Go to step 5.

Step 16. Output the best makespan value from the last generation.

4. Lower Bound

The solution quality of the proposed GHA and GA can be evaluated against any benchmark procedure. Since there is no published literature for evaluation of the heuristic algorithm for the problem configuration studied in this paper, we attempt to develop LB on makespan for large scale problems.

In order to develop a lower bound for the problem under consideration, we combine the LB procedures proposed by Lee and Uzsoy(1999) (which considers NIJS and releasetime characteristic) for scheduling BP and LB procedures proposed by Martello et al. (2000) that considers NIJD characteristic. The details of LB procedure to obtain LB on makespan are as follows:

Step 1: Index the jobs in the non-decreasing order of the release times r_j

Step 2: Consider all the jobs to be available at time zero.

Step 3: Now calculate *CMAX(f, j, n)* for jobs *j, ..., n* belonging to the first job family '*f*' considering only NIJD characteristic. For this LB procedure proposed by Martello et al. (2000), taking appropriate processing time for the family. This is called $C_{max}^{NIJD}(f, j, n)$. The details of this LB procedure are given in *Appendix 1*.

Step 4: Calculate LB on *CMAX(f,j,n)* for jobs *j=1,..., n* belonging to the first job family '*f*' by considering only NIJS characteristic(called $C_{max}^{NIJS}(f, j, n)$) using equation (1) which is analogous to LB procedure given in *Appendix*.

Step 5: Calculate LB on *CMAX (f, j,n)* for the job family by considering both NIJD and NIJS and this is = max $\{C_{max}^{NIJS}(f, j, n), C_{max}^{NIJD}(f, j, n)\}$.

Step 6: Add the release time of the job at head of the list with *CMAX (f, j, n)*.

Step 7: Store the value (r_j + *CMAX(f, j, n)*) for the family in a separate list.

Step 8: Repeat step 3-7 for other job families in the list with their appropriate processing times.

Step 9: Then LB for the research problem, $LBCMAX^{NARDIMJF}$ is maximum of all the values

stored in the separate list. i.e.

$$LBCMAX^{NARDMIJF} = \max_{j=1,2,\ldots,n} \{ r_j + \sum_{f=1}^{p} CMAX(f, j, n) \}$$

Where, *p* is number of job families.

The appropriate equation for calculating $C_{max}^{NIJS}(f, j, n)$ for a selected job family 'f' is :

$$C_{max}^{NIJS}(j,n) = p^* \left| \left\{ j \in J : s_j > \frac{S}{2} \right\} \right| + p^* \max_{1 \le r \le s/2} \left\{ \left\lceil \frac{\sum_{j \in J_2(r)} s_j - \left(|J_1(r)| S - \sum_{j \in J_1(r)} s_j \right)}{S} \right\rceil, \left\lceil \frac{|J_3(r)| - \sum_{j \in J_3(r)} \left\lfloor \frac{S - s_j}{r} \right\rfloor}{\left\lfloor \frac{S}{r} \right\rfloor} \right\rceil \right\}$$

where,

$$J_1(r) = \left\{ j \in J : S - r \ge s_j > \frac{S}{2} \right\},$$

$$J_3(r) = \left\{ j \in J : \frac{S}{2} \ge s_j \ge r \right\} \qquad (1)$$

sj	-	Size of job
S	-	Size of the BP
p	-	Processing time of the job family
r	-	Integer
C_{max}^{NIJS}	-	Lower bound on makespan by considering only NIJS

The procedure to obtain LB on makespan for scheduling a BP with MIJF, NIJS, NIJD and NARD is implemented in C language on a system with 1GB RAM, 2.4Ghz processor with Windows XP operating system. Using the LB procedure, the proposed greedy heuristic algorithms (nine variants of GHA) for scheduling a BP with MIJF, NIJS, NIJD and NARD are evaluated in the next section.

In this section, we discuss the details of the computational experiments, proposed for evaluating the performance of the nine variants of FMGHA and FMGA.

5. Computational Experiments

In this section, we discuss the details of the computational experiments, proposed for evaluating the performance of the nine variants of GHA and *GA*.

5.1 Experimental Design

The experimental design is the process of planning an experiment to ensure that the appropriate test data are identified and/or collected or generated. To the best of our knowledge, the problem configuration studied in this paper has not been addressed so far in the literature and consequently no benchmark data set is available. Based on the observation from a couple of user industries located in Tamilnadu, India for a week's time, we selected the parameters: Number of jobs (n), Job size (s_i), Length (l_i), Width (w_i), Height (h_i), Release time (r_i), Due date (d_i), Number of Job families(f_i).

Based on the above parameters, an experimental design is developed to study the quality of the proposed heuristic algorithms on makespan in comparison with LB the summary of which is given in Table 3. In the proposed experimental design, the job parameters are assumed to follow uniform distribution as given in Table 3. Uniform distribution is chosen because it is a relatively high-variance distribution, which would allow the LB to be tested under conditions relatively unfavorable to them (Chandru et al., 1993a). Furthermore, the range of intervals for the NIJS and NIJD are based on the observation from the user industries and also by trial experiments. In addition, it is observed from the industries that there are 4 to 6 incompatible job families on an average in front of HTF. Thus we assume two levels for the parameter, MIJF, namely 4 job families and 6 job families for this research problem.

Also, for the NARD we set range values based on observation from the user industries. The scheduling horizon commonly followed in the user industries is one week which consists of total 168 h. Jobs arrive at various points of time within this given week to the work-in-process inventory before the BP. But, since in our study the problem parameter: number of jobs starts at 25, the release time range of 0 to 168 h will result in only few jobs competing for the BP at any given time. Thus, the release time ranges for the study are assumed to be 0 to 84 h and 0 to 42 h respectively. Also, each job requires anywhere between one week to ten days for completion of heat-treatment operation (as there are some inventory of jobs which are already waiting before the BP and the longer processing time required at BP). Based on this the due date range is fixed between 168 h and 240 h. The summary of the experimental design including the release time data and due date data for evaluating the heuristic algorithms for the problem addressed in this paper is given in Table 3.

Table 3: A summary of experimental design for research problem

Problem Parameters		No. of Levels	Values
No. of jobs (n)		6	25, 50, 75, 100, 125, 150
Job Parameters	Number of job families (f_i)	2	4,6
	Release time (r_i)	2	U[0,84], U[0,42]
	Due date (d_i)	1	r_i +U[168,240]
	Job size (s_i)	2	U[1, S/2], U[1, S/4]
	Job length (w_i)	2	U[1, W], U[1, W/2]
	Job height (h_i)	2	U[1, H], U[1, H/2]
	Job length (l_i)	2	U[1, L], U[1, L/2]
Number of configurations			6 x 2 x 1x 2 x 2 x 2 x 2 x 2 = 384
Number of instances per configuration			10
Total number of instances			3840

The experimental design shown in Table 3 for generating test problems is implemented in programming language C. For each combination of values for (*f*, *d*, *n, s, w, h, l, r*), ten problem instances were randomly generated, yielding a total of 3840 [= 10 x (6x2x1x2x2x2x2x2)] problem instances.

In addition to the input provided for the problem parameters, mentioned in Table 3, we assume that there is only one BP which has: Size: S = 2500 kg; and Dimensions: L = 2500 mm; W = 1000 mm; H = 1250 mm.

Also, we assume the processing time of the job families as follows:

No. of Job Families	Processing times of Job Families (h)
4	13, 15, 12, 10 respectively
6	13, 15, 12, 10, 22, 18 respectively

5.2 Measures of effectiveness

Since the performance of the proposed heuristic algorithms may vary over problem instances, the following performance measures are used for comparison:

Relative Percentage Deviation (RPD) of a heuristic solution from LB on makespan

Average Relative Percentage Deviation (ARPD)

Maximum Relative Percentage Deviation (MRPD)

Let C_H be the makespan given by the proposed heuristic algorithm "H", where H = 1 = SLB (NARD&MIJF), H = 2 = SWB (NARD&MIJF), H = 3 = SHB (NARD&MIJF), H = 4 = SVB (NARD&MIJF), H = 5 = SSB (NARD&MIJF), H = 6 = SDB (NARD&MIJF), H = 7 = SVDB (NARD&MIJF), H = 8 = SSDB (NARD&MIJF) and H = 9 = SRB (NARD&MIJF), H = 10 = *GA(NARD&MIJF)*. Let Copt be the Cmax given by the proposed LB. Then, the Relative Percentage Deviation (RPD) on instance 'i' for the proposed heuristic algorithm "H" is RPDH(i) and computed as follows:

$$RPD_H(i) = \left(\frac{C_H(i) - C_{opt}(i)}{C_{opt}(i)} \right) * 100 \qquad (2)$$

The average RPD (i.e , ARPD) and the maximum RPD (i.e., MRPD) for the proposed heuristic algorithm "H", for the problem parameter p are calculated as follows:

$$ARPD_H(p) = \frac{\sum_{i=1}^{N} RPD_H(i)}{N} \quad (3)$$

and

$$MRPD_H(p) = \max_{1 \le i \le N} \{RPD_H(i)\} \quad (4)$$

Where,

ARPD $_H$ (p) = ARPD of proposed heuristic algorithm 'H' for problem parameter *p* over N instances of planned configuration *p* and

$MRPD_H$ (p) = MRPD of proposed heuristic algorithm 'H' for problem parameter *p* over N instances of planned configuration *p*

5.2.1 Absolute Evaluation of the Proposed Heuristic Algorithms:

For this evaluation, the test data (3840 instances) generated according to the experimental design in Table 3 is used. First, the proposed heuristic algorithms and the GA are run through each of the 3840 instances and the makespan value is recorded for each instance. Then the LB procedure is also run through each of these 3840 instances and the LB on makespan is obtained for each instance. The details of these results are not presented due to the brevity of the report.

However, using these results, heuristic wise, the number of times the heuristic result matched with the LB on makespan is computed. Further, average loss with respect to LB on makespan (in percentage) over 1920 instances is computed. These results are shown in the Table 4.

Table 4: Performance of the proposed heuristic algorithms with respect to LB

Performance Measure	Proposed variants of GHA									Proposed *GA* (NARD &MIJF)
	SLB (NARD &MIJF)	SWB (NARD &MIJF)	SHB (NARD &MIJF)	SVB (NARD &MIJF)	SSB (NARD &MIJF)	SDB (NARD &MIJF)	SVDB (NARD &MIJF)	SSDB (NARD &MIJF)	SRB (NARD &MIJF)	
Number of times heuristic solution equal to LB on makespan	105	112	97	110	265	56	107	193	40	1370
Average deviation of a heuristic solution from LB on makespan	48%	41%	45%	43%	47%	55%	43%	48%	57%	16%

From the Table 4, it is observed that *GA(NARD&MIJF)* outperforms the other proposed heuristic algorithms for the experimental data used in this study. But, the computational time for *GA(NARD&MIJF)* is in the order of minutes, whereas the computational time for each of the GHA is in the order of milliseconds for large scale problem size (***n*** = 150). Thus, due to this computational advantage, the proposed variant of the GHA: *SWB(NARD&MIJF)* could be a good candidate for scheduling a BP with MIJF, NIJS, NIJD and NARD among the proposed greedy heuristic algorithms.

For each problem instance, the makespan value obtained by each of the proposed heuristic algorithms and the LB procedure, the RPD value is computed using equation (2). In order to check for the influence of individual problem parameters on the performance of the proposed heuristic algorithms in comparison with LB on makespan, the computed RPD score is used. For this purpose, a statistical test is undertaken. Accordingly, a multi-factor (***heuristic algorithm, n, f, r, s, w, h, l***) ANOVA is used on the score: RPD. The statistical package SIGMASTAT is used for this analysis. Since the normality and equal variance assumption failed for our data, the non-parametric tests: Mann-Whitney test for factors with two groups (***f***, ***r, s, h, w, l***) and Kruskal-Wallis test for factors with more than two groups (***n, heuristic algorithm***) are used. The results of this analysis (*Appendix 2*) show that there is an influence of all problem parameters on the performance of the heuristic algorithms.

The computed Relative Percentage Deviation (RPD) of heuristic algorithm 'H' on problem instance '*i*' ($RPD_{H(i)}$), is used for calculating the scores: ARPD and MRPD. These scores are calculated, using the equation (3) and equation (4) for each of the heuristic algorithms, as follows:

(a) For each level of the problem parameter ***n*** and the problem class (*, ***f***, ***r***, ***s***, ***w***, ***h***, ***l***) over 640 problem instances.

(b) For each level of the problem parameter ***f*** and the problem class (***n***, *, ***r***, ***s***, ***w***, ***h***, ***l***) over 1920 problem instances

(c) For each level of the problem parameter ***r*** and the problem class (***n***, ***f***, *, **s**, ***w***, ***h***, ***l***) over 1920 problem instances.

(d) For each level of the problem parameter ***s*** and the problem class (***n***, ***f***, ***r***, *, ***w***, ***h***, ***l***) over 1920 problem instances.

(e) For each level of the problem parameter ***w*** and the problem class (***n***, ***f***, ***r***, ***s***, *, ***h***, ***l***) over 1920 problem instances.

(f) For each level of the problem parameter ***h*** and the problem class (***n***, ***f***, ***r***, ***s***, ***w***, *, ***l***) over 1920 problem instances.

(g) For each level of the problem parameter ***l*** and the problem class (***n***, ***f***, ***r***, ***s***, ***w***, *, ***l***) over 1920 problem instances.

The computed ARPD and MRPD scores are presented in Table 5 and Table 6 respectively. From the Table 5 and the Table 6 we can observe that: (i) the GA: *GA(NARD&MIJF)* outperforms the other proposed heuristic algorithms and (ii) within the proposed variants of GHA: the heuristic algorithm *SWB(NARD&MIJF)* perform relatively better and the algorithm *SVDB(NARD&MIJF)* becomes relatively the second best one.

In addition, irrespective of problem parameters, the ARPD and MRPD scores are computed over 3840 instances and these are shown in the Figure 2 and Figure 3 respectively. Figure 2 and Figure 3 indicate the same observations made based on the results from Table 5 and 6.

Table 5: Performance of the proposed heuristic algorithms (ARPD based on LB)

Problem Configuration	No. of Instances	Proposed variants of GHA									Proposed GA (NARD & MIJF)
		SLB (NARD & MIJF)	SWB (NARD & MIJF)	SHB (NARD & MIJF)	SVB (NARD & MIJF	SSB (NARD & MIJF)	SDB (NARD & MIJF)	SVDB (NARD & MIJF)	SSDB (NARD & MIJF)	SRB (NARD & MIJF)	
(*n* = *1*, *, *, *, *, *, *)	640	18.9	17.5	18.4	17.5	19.5	22.9	17.8	20.6	26.3	4.4
(*n* = *2*, *, *, *, *, *, *)	640	17	15.8	16.9	16.2	16.5	20.3	16.6	17.6	21.4	3.9
(*n* = *3*, *, *, *, *, *, *)	640	24.9	20.8	22.3	21.1	28.4	29.3	21.5	29.2	29.5	9.2
(*n* = *4*, *, *, *, *, *, *)	640	15.3	13.7	14.2	13.5	14.5	17.1	13.9	15.1	17.2	4.9
(*n* = *5*, *, *, *, *, *, *)	640	12.9	11.4	12.6	11.6	12.4	14.2	11.8	12.6	14.6	4.3
(*n* = *6*, *, *, *, *, *, *)	640	18.2	13.3	15.4	14.2	23.8	20.8	14.4	23.4	20.6	7.9
(*, *f* = *1*, *, *, *, *, *)	1920	19.1	16.4	17.7	16.7	20.4	21.5	17	20.8	22.3	6.2
(*, *f* = *2*, *, *, *, *, *)	1920	16.7	14.4	15.5	14.6	18	20	15	18.7	20.9	5.3
(*, *, *r* = *1*, *, *, *, *)	1920	19.9	17.8	18.9	17.8	21.4	23.4	18.2	22.2	24.5	6.6
(*, *, *r* = *2*, *, *, *, *)	1920	15.9	13.1	14.4	13.5	17	18.1	13.8	17.3	18.7	4.9
(*, *, *, *s* = *1*, *, *, *)	1920	10.6	9.9	10.3	9.9	9	12.4	10	9.6	13.3	2.5
(*, *, *, *s* = *2*, *, *, *)	1920	25.2	20.9	23	21.5	29.4	29.2	22	29.9	29.9	9
(*, *, *, *, *h* = *1*, *, *)	1920	22.7	18	20.2	19	25.3	26	19.2	25.9	26.9	9
(*, *, *, *, *h* = *2*, *, *)	1920	13	12.8	13	12.4	13.1	15.5	12.8	13.6	16.3	2.5
(*, *, *, *, *, *w* = *1*, *)	1920	22.5	18	20.1	18.6	24.9	26.2	19	25.7	27	8.9
(*, *, *, *, *, *w* = *2*, *)	1920	13.2	12.8	13.2	12.8	13.4	15.4	13	13.8	16.2	2.6
(*, *, *, *, *, *, *, *l* = *1*)	1920	21	17.3	19.6	17.8	23.4	24.6	18	24	25.4	8.1
(*, *, *, *, *, *, *, *l* = *2*)	1920	14.7	13.5	13.7	13.6	15	16.9	14	15.5	17.8	3.4

Table 6: Performance of the proposed heuristic algorithms (MRPD based on LB)

Problem Configuration	No. of Instances	Proposed variants of GHA									Proposed GA (NARD & MIJF)
		SLB (NARD & MIJF)	SWB (NARD & MIJF)	SHB (NARD & MIJF)	SVB (NARD & MIJF	SSB (NARD & MIJF)	SDB (NARD & MIJF)	SVDB (NARD & MIJF)	SSDB (NARD & MIJF)	SRB (NARD & MIJF)	
(*n* = *1*, *, *, *, *, *, *)	640	78.4	75	66.7	78.4	87	90.5	78.4	87	79.1	53
(*n* = *2*, *, *, *, *, *, *)	640	87.6	71.9	110.6	81.7	110.6	105.7	72.4	104	99.3	67.1
(*n* = *3*, *, *, *, *, *, *)	640	102.8	85.7	89.9	91.4	100	108.8	82.1	114.3	107.2	71.4
(*n* = *4*, *, *, *, *, *, *)	640	89.6	61.5	77.6	78	99.4	109.1	74.2	108.5	104	60.8
(*n* = *5*, *, *, *, *, *, *)	640	101.7	61.8	80.9	74.7	107.7	104.6	71.4	107.5	106.7	57.8
(*n* = *6*, *, *, *, *, *, *)	640	95.8	69.1	79	73.8	110	106.5	72.9	107.6	102.3	67.3
(*, *f* = *1*, *, *, *, *, *)	1920	102.8	85.7	94	91.4	107.7	106.5	82.1	114.3	107.2	71.4
(*, *f* = *2*, *, *, *, *, *)	1920	87.6	75.6	110.6	81.7	110.6	109.1	77.3	108.5	106.7	67.3
(*, *, *r* = *1*, *, *, *, *)	1920	102.8	75.6	110.6	73.8	110.6	108.8	77.3	108.5	106.7	67.3
(*, *, *r* = *2*, *, *, *, *)	1920	101.7	85.7	94	91.4	106.3	109.1	82.1	114.3	107.2	71.4
(*, *, *, *s* = *1*, *, *, *)	1920	69.1	71.9	71.9	67.1	88	71.9	67.1	85.8	77.8	67.1
(*, *, *, *s* = *2*, *, *, *)	1920	102.8	85.7	110.6	91.4	110.6	109.1	82.1	114.3	107.2	71.4
(*, *, *, *, *h* = *1*, *, *)	1920	102.8	85.7	110.6	91.4	110.6	109.1	82.1	114.3	107.2	71.4
(*, *, *, *, *h* = *2*, *, *)	1920	67.2	62.5	59.5	59.1	67.1	65	59.1	72	75.3	44.4
(*, *, *, *, *, *w* = *1*, *)	1920	102.8	85.7	110.6	91.4	110.6	109.1	82.1	114.3	107.2	71.4
(*, *, *, *, *, *w* = *2*, *)	1920	75.2	71.9	71.9	67.1	85	90.8	67.1	89.1	73.2	67.1
(*, *, *, *, *, *, *l* = *1*)	1920	102.8	85.7	110.6	91.4	110.6	109.1	82.1	114.3	107.2	71.4
(*, *, *, *, *, *, *l* = *2*)	1920	80	75.6	71.9	78.4	86.6	88.9	78.4	87.7	91.1	67.1

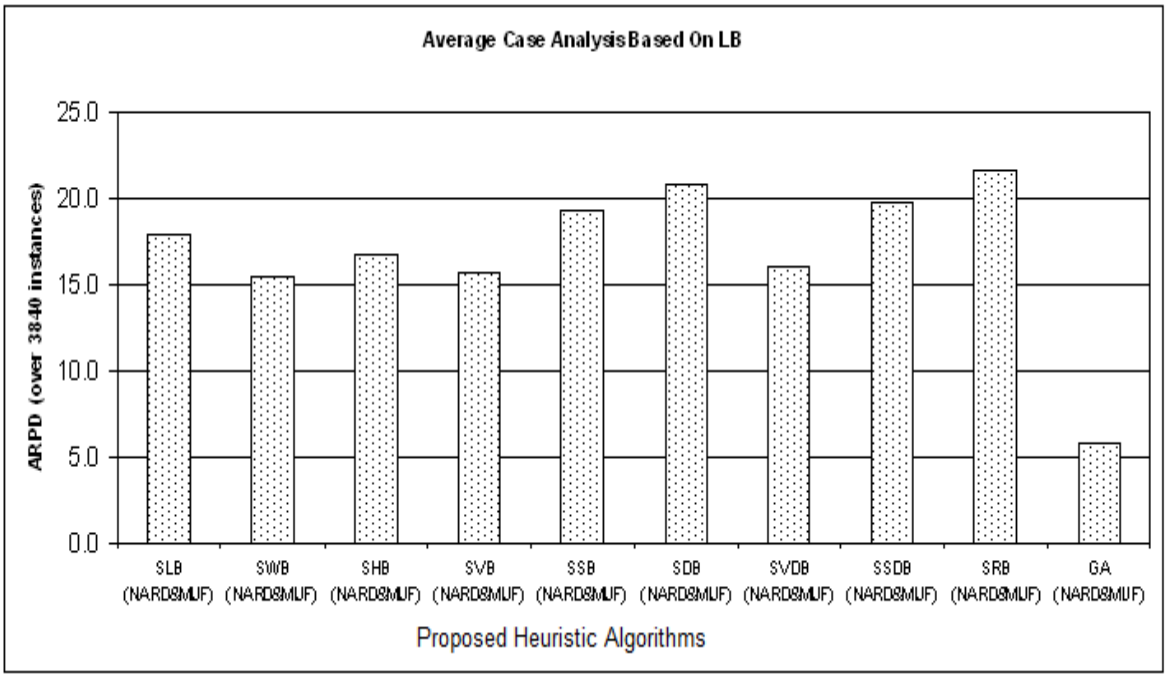

Figure 2: Performance of heuristic algorithms in comparison with LB on 3840 instances

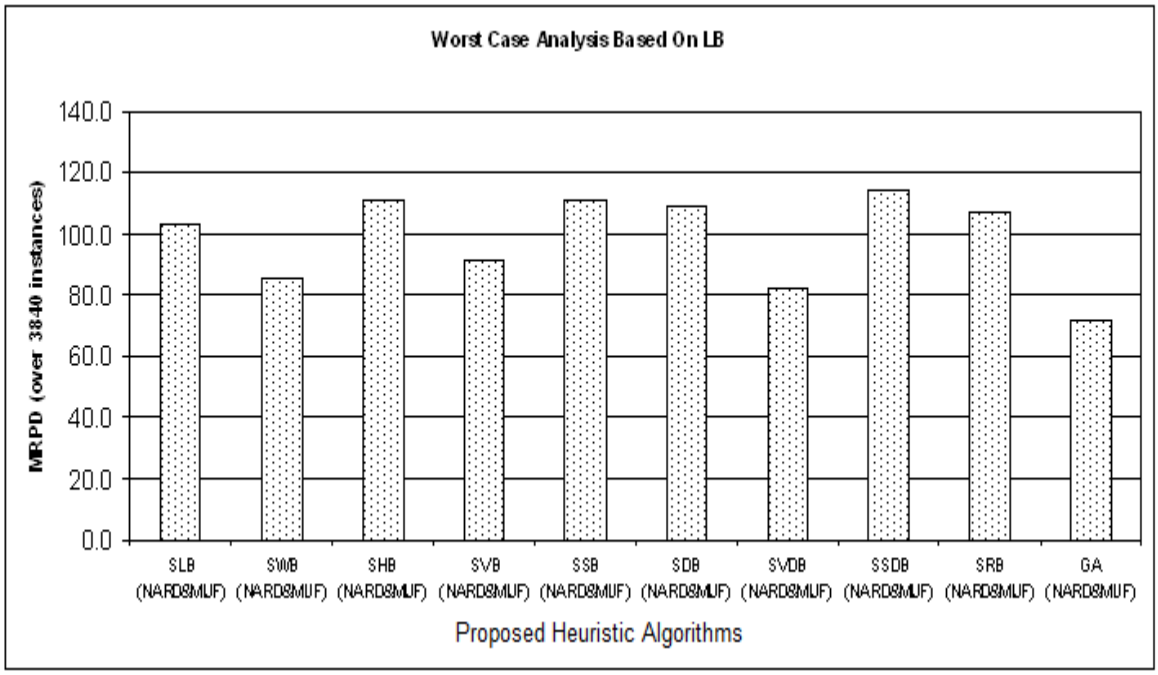

Figure 3: Performance of heuristic algorithms in comparison with LB on 3840 instances

Overall, we can conclude that the modified heuristic algorithm: *GA(NARD&MIJF)* outperforms the proposed heuristic algorithms in scheduling a BP with MIJF, NIJS, NIJD and NARD in comparison with LB. However, if decision maker wishes to choose one or two simple common sense heuristic procedures such as the proposed variants of a *GHA*, the variants of *GHA*: *SWB(NARD&MIJF) and SVDB(NARD&MIJF)* could be best choice as they perform relatively better among the variants of a *GHA* and could be good candidates for scheduling a BP with MIJF, NIJS, NIJD and NARD&MIJF.

6. Conclusion

In a recent observation of heat-treatment operations in a couple of steel casting industries and the research studies reported in the literature, we noticed that the real-life problem of dynamic scheduling of heat-treatment furnace with multiple incompatible job families, non-identical job sizes, non-identical job dimensions, non-agreeable release times and due dates to maximize the throughput, higher utilization and minimize the work-in-process inventory has not been addressed.

Due to the difference between the real-life situation on dynamic scheduling of heat-treatment furnace of the steel casting manufacturing and the research reported on the same problem, we identified a new batch processor problem, which is applicable to a real-life situation.

Due to the computational intractability of the research problem, we proposed a GHA and GA for the research problem. In order to evaluate the performance of the proposed *GHA* and *GA*, particularly for large scale problems, a lower bound procedure is proposed for the research problem. A series of computational experiments are carried out to conduct absolute evaluation in comparison with lower bound for small and large scale problems. Based on the series of computational experiments conducted for the research problem defined in this study, we observe the following:

- The problem parameters considered in this study has influence on the performance of the heuristic algorithms.
- The proposed lower bound procedure is found to be efficient.

- The proposed *GA(NARD&MIJF)* is efficient. However, the computational time required to obtain efficient solution increases as problem size increases.
- In case, the decision maker is interested in choosing one among the proposed GHA, the algorithm *sort by width and construct batch* (*SWB), SWB by considering additional restriction on non-agreeable release times and due dates(SWB(NARD&MIJF))* is relatively better algorithm for the research problem.

The present study considers only a single BP for the study. In reality, however, there will be multiple BPs with non-identical capacity restrictions in some small-scale casting industries. This study does not attempt to address the research problem with multiple non-identical BPs. However, the proposed heuristic algorithms for scheduling the single BP can be used whenever only one BP among the multiple BPs becomes free and the jobs can be scheduled on this BP. In the case of a tie, a BP can be selected based on the manager's decision rule such as: select the BP with the maximum capacity or select the BP which has been scheduled less number of times so far, *etc*. Then, the proposed heuristic algorithms can be used for this selected BP and the jobs can be scheduled accordingly.

Also, in this study, only *GA* has been proposed as meta-heuristic solution method. Solution methodology involving any meta-heuristic such as ant colony optimisation etc. could be another interesting research direction.

Reference:

Boschetti, M. A. (2004). 'New Lower Bounds for the Three-dimensional Finite Bin Packing Problem', *Discrete Applied Mathematics* **140**, 241-258.

Chang, P. C. and Wang, H. M. (2004). 'A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes', *International Journal Of Advanced Manufacturing Technology* **24**(7-8), 615-620.

Chang, P. C., Chen, Y. S. and Wang, H. M. (2005). 'Dynamic scheduling problem of batch processing machine in semiconductor burn-in operations', *Computational Science And Its Applications - Iccsa 2005, Vol 4, Proceedings* **3483**, 172-181.

Chou, F. (2007). 'A joint GA plus DP approach for single burn-in oven scheduling problems with makespan criterion', *International Journal Of Advanced Manufacturing Technology* **35**(5-6), 587-595.

Chou, F., Chang, P. and Wang, H. (2006). 'A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem', *International Journal Of Advanced Manufacturing Technology* **31**(3-4), 350-359.

Damodaran, P.,Velez-Gallego, M.C., (2012). A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times, Expert Systems with Applications. **39**, 1451-1458.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman.

Kashan, A., Karimi, B., 2007, "Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework", Journal of Operational Research Society, **59**, 1269-1280**.**

Koh, S. G., Koo, P. H., Kim, D. C. and Hur, W. S. (2005). 'Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families', *International Journal Of Production Economics* **98**(1), 81-96.

Krishnaswamy, K. N., Raghavendra, B. G. and Srinivasan, M. N. (1998). 'Development of DSS for production planning and control for SECALS', Project Report, Department of Management Studies, Indian Institute of Science, Bangalore, India.

Krishnaswamy, K. N., Sivakumar, A. I. and Mathirajan, M. (2006). *Management Research Methodology: Integration of Principles, Methods and Techniques*, Pearson Education.

Lee, C. Y. and Uzsoy, R. (1999). 'Minimizing makespan on a single batch processing machine with dynamic job arrivals', *International Journal Of Production Research* **37**(1), 219-236.

Li, S. G., Li, G. J., Wang, X. L. and Liu, Q. M. (2005). 'Minimizing makespan on a single batching machine with release times and non-identical job sizes', *Operations Research Letters* **33**(2), 157-164.

Martello, S., Pisinger, D. and Vigo, D. (2000). 'The Three dimensional Bin Packing Problem', *Operations Research* **48**(2), 256-267.

Mathirajan, M. (2002). 'Heuristic scheduling algorithms for parallel heterogenous batch processors', *PhD Thesis*, Department of Management Studies, Indian Institute of Science.

Mathirajan, M., Sivakumar, A. I. and Chandru, V. (2004a). 'Scheduling algorithms for heterogeneous batch processors with incompatible job-families', Journal Of Intelligent Manufacturing 15(6), 787-803.

Mathirajan, M. and Sivakumar, A. I. (2006a). 'Minimizing total weighted tardiness on heterogeneous batch processing machines with incompatible job families', *International Journal Of Advanced Manufacturing Technology* **28**(9), 1038-1047.

Mathirajan, M. and Sivakumar, A. I. (2006b). 'A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor', *International Journal Of Advanced Manufacturing Technology* **29**(9-10), 990-1001.

Nong, Q., Ng, C. T. and Cheng, T. C. E. (2008). 'The bounded single-machine parallel-batching scheduling problem with family jobs and release dates to minimize makespan', *Operations Research Letters* **36**(1), 61-66.

Pinedo, M. L. (1995). *Scheduling: Theory, Algorithms and Systems*, Prentice Hall.

Ramasubramaniam, M. (2008). 'Batch Processor Scheduling – A Class of Problems in Steel Casting Industries', *PhD thesis*, Department of Management Studies, Indian Institute of Science, Bangalore, India.

Shekar, G. L. (1998). 'Planning and Scheduling systems for steel casting production -A New Paradigm', *PhD thesis*, Department of Management Studies, Indian Institute of Science, Bangalore, India.

Venkatramana, M. (2006). 'Deterministic scheduling of parallel discrete and batch processors', *PhD Thesis*, Department of Management Studies, Indian Institute of Science.

Wang, H. M., Chang, P. C. and Chou, F. D. (2007). 'A hybrid forward/backward approach for single batch scheduling problems with non-identical job sizes', Journal Of The Chinese Institute of Industrial Engineers 24(3), 191-199.

APPENDIX 1

LB procedure for computing LB on makespan by considering NIJD characteristics

Notations:

l_j	-	Length of job
w_j	-	Width of job
h_j	-	Height of job
v_j	-	Volume of job (= l_j * w_j * h_j)
L	-	Length of BP
W	-	Width of BP
H	-	Height of BP
p, q	-	Integer pairs
L_1^{LW}	-	Lower bound for Length and Width by reduction to one dimension
$L_2^{LW}(p,q)$	-	Intermediate Lower bound for dimensions Length and Width by considering integer pairs p and q
L^{LW}	-	Lower bound for dimensions Length and Width
L^{WH}	-	Lower bound for dimensions Width and Height
L^{LH}	-	Lower bound for dimensions Length and Height
C_{max}^{NIJD}	-	Makespan obtained using only NIJD

Procedure:

Step 1: Compute LB on number of batches as proposed by Martello et al. (2000) by considering only two dimensions, namely Length (L) and Width (W) as follows:

$$L^{LW} = \max_{1\le p\le L/2;1\le q\le W/2}\left\{L_2^{LW}(p,q)\right\}$$

Where,

$$L_2^{LW}(p,q) = L_1^{LW} + \max\left\{0, \left\lceil \frac{\sum_{j\in K_l(p,q)\cup K_s(p,q)} v_j - \left(HL_1^{LW} - \sum_{j\in K_v(p,q)} h_j\right)LW}{B} \right\rceil\right\}$$

where,

$$L_1^{LW} = \left|\left\{j \in J^{LW} : h_j > \frac{H}{2}\right\}\right|$$

$$+ \max_{1\le p\le H/2}\left\{\left\lceil \frac{\sum_{j\in J_{s(p)}} h_j - \left(|J_l(p)|H - \sum_{j\in J_l(p)} h_j\right)}{H}\right\rceil, \left\lceil \frac{|J_s(p)| - \sum_{j\in J_l(p)}\left\lfloor \frac{H-h_j}{p}\right\rfloor}{\left\lfloor \frac{H}{p}\right\rfloor}\right\rceil\right\}$$

and $B = LWH$

$$J^{LW} = \left\{j \in J : l_j > \frac{L}{2} \text{ and } w_j > \frac{W}{2}\right\},$$

$$J_l(p) = \left\{j \in J^{LW} : H - p \ge h_j > \frac{H}{2}\right\},$$

$$J_s(p) = \left\{j \in J^{LW} : \frac{H}{2} \ge h_j \ge p\right\},$$

$$K_v(p,q) = \{j \in J : l_j > l - p \text{ and } w_j > W - q\},$$

$$K_l(p,q) = \left\{j \in J \setminus K_v(p,q) : l_j > \frac{L}{2} \text{ and } w_j > \frac{W}{2}\right\},$$

$$K_s(p,q) = \left\{j \in J \setminus K_v(p,q) \cup K_l(p,q) : l_j \ge p \text{ and } w_j \ge q\right\}$$

Step 2: Similar to the above computation of LB on number of batches considering the two dimensions: Length and Width, compute LB on number of batches by considering the other combinations of two dimensions: Width and Height (L^{WH}) and Length and Height (L^{LH}).

Step 3: Using the lower bound: L^{LW}, L^{WH} and L^{LH} on number of batches, compute the LB on makespan by considering NIJD as:

C_{max}^{NIJD} = Processing time of the job family * $maximum\{L^{LW}, L^{WH}, L^{LH}\}$

APPENDIX 2

Results of the SIGMASTAT package for Multi-factor ANOVA

Normality Test: Failed (p <0.001)

Equal Variance Test: Failed (p <0.001)

Factor Vs. RPD	Name of the Non-Parametric Test	Name of Test Statistic	Test Statistic Value	p-value (95% confidence level)
n	Kruskal -Wallis	H	2685.397	< 0.001
f	Mann -Whitney	T	381721807.5	< 0.001
r	Mann -Whitney	T	403262759	< 0.001
s	Mann -Whitney	T	273569473	< 0.001
h	Mann -Whitney	T	412250642	< 0.001
w	Mann -Whitney	T	412179153	< 0.001
l	Mann -Whitney	T	400282143	< 0.001
heuristic algorithm	Kruskal -Wallis	H	4496.915	< 0.001