Sādhanā Vol. 32, Part 5, October 2007, pp. 479–500. © Printed in India

Heuristic algorithms for scheduling heat-treatment furnaces of steel casting industries

M MATHIRAJAN[1,*], V CHANDRU[1] and A I SIVAKUMAR[2]

[1]Department of Management Studies, Indian Institute of Science, Bangalore 560 012
[2]Singapore-MIT Alliance, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639 798
e-mail: msdmathi@mgmt.iisc.ernet.in

Ms received 25 August 2005; revised 4 July 2006

Abstract. This paper addresses a research problem of scheduling parallel, non-identical batch processors in the presence of dynamic job arrivals, incompatible job-families and non-identical job sizes. We were led to this problem through a real-world application involving the scheduling of heat-treatment operations of steel casting. The scheduling of furnaces for heat-treatment of castings is of considerable interest as a large proportion of the total production time is the processing times of these operations. In view of the computational intractability of this type of problem, a few heuristic algorithms have been designed for maximizing the utilization of heat-treatment furnaces of steel casting manufacturing. Extensive computational experiments were carried out to compare the performance of the heuristics with the estimated optimal value (using the Weibull technique) and for relative effectiveness among the heuristics. Further, the computational experiments show that the heuristic algorithms proposed in this paper are capable of obtaining near (statistically estimated) optimal utilization of heat-treatment furnaces and are also capable of solving any large size real-life problems with a relatively low computational effort.

Keywords. Heat-treatment furnaces; heuristic algorithms; Weibull technique; scheduling batch processors.

1. Introduction

In the late 1970s and the early 1980s, market pressure for greater product variety forced a gradual shift from continuous manufacturing to batch manufacturing (Roberts *et al* 1999). As a sequel to this, in the last decade, deterministic manufacturing batch scheduling problems have attracted the attention of researchers. The earliest work in the deterministic scheduling of batch processors appears to be that of Ikura & Gimple (1986).

In this paper, we consider the problem of scheduling jobs on heat-treatment furnaces (HTF) in the post-casting stage of foundry manufacturing. This is an extension to our earlier study

*Corresponding author

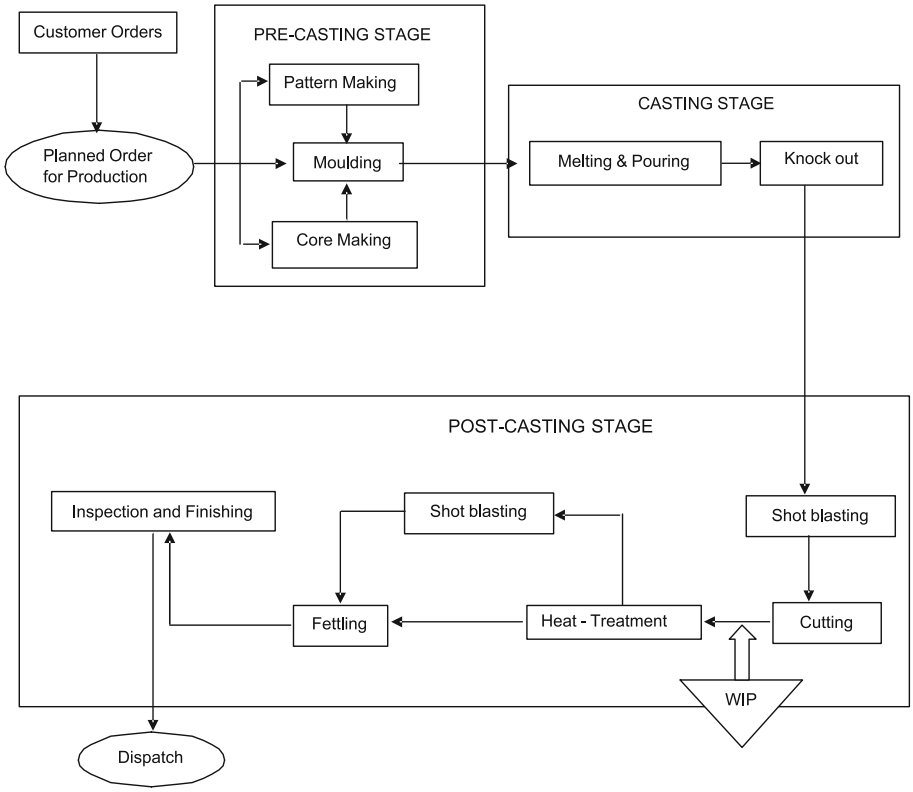

Figure 1. A typical steel-casting manufacturing process sequence.

(Mathirajan *et al* 2001). Although this work is related to the application in the steel casting manufacturing, similar problems are encountered in other industrial settings, such as diffusion/oxidation in the semiconductor manufacturing [see Fowler *et al* (1992), Uzsoy *et al* (1992) and ovens used for hardening of the synthetic parts in aircraft industries [see Zee *et al* (1997)].

A fundamental feature of foundry manufacturing is its extreme flexibility, enabling castings to be produced with almost unlimited freedom in design over an extremely wide range of sizes, quantities, and materials suited to practically every environment and application. Furthermore, the foundry manufacturing industry is capital-intensive and highly competitive. The latter forces a greater emphasis on customer service.

1.1 *Heat treatment operations*

Like all foundries, a steel foundry is a flow line production system in which the sequence of operations is fixed and the workflow is in a single direction. A typical sequence of operations in a steel foundry is given in figure 1. The working mechanism of the steel-casting foundry studied is briefly described here.

Based on planned orders, moulds (and cores) are prepared using patterns. These moulds are moved to the pouring area where molten metal from melting furnaces are poured and allowed to cool. In the next operation, castings are knocked-out of the mould cavity either manually

or mechanically. The knocked-out rough castings are then shot blasted and cut to the finished castings which are generally moved to storage prior to the next process; heat-treatment. The stored castings are grouped into batches depending on the type and family of castings, and loaded on the furnaces for the process-controlled heat treatment operation. Subsequently, the heat-treated castings are fettled, finished and inspected prior to dispatch to the customers.

From the viewpoint of throughput and utilization of the important and costly resources, it was felt that the process-controlled furnace operations for the melting and pouring operations as well as the heat-treatment furnace operations are critical for meeting the overall production schedules. The two furnace operations are batch processes that have distinctive constraints on job-mixes in addition to the usual capacity and technical constraints associated with any industrial process. The benefits of effective scheduling of these batch processes include higher machine utilization, lower work-in-process (WIP) inventory, shorter cycle time, and greater customer satisfaction (Pinedo 1995).

Recently production planning and scheduling models for a steel foundry, considering the melting furnace of the pre-casting stage as the core foundry operation were proposed (Voorhis *et al* 2001), Krishnaswamy *et al* (1998) and Shekar (1998). Even though the melting and pouring operations may be considered as the core of foundry operations and their scheduling is of central importance, the scheduling of heat-treatment furnaces is also of considerable importance. This is because the processing time required at the heat treatment furnace is often longer compared to other operations in the steel-casting foundry and therefore considerably affects the scheduling, overall flow time and WIP inventory.

Further, the heat-treatment operation is critical because it determines the final metallurgical properties that, enables the components to perform under demanding service conditions such as large mechanical load, high temperature, and in corrosive environment. Generally, every type of casting has to undergo more than one form of heat-treatment operation, where the total processing times changes. For control purposes, castings are primarily classified into a number of job-families based on the alloy type such as low-alloy castings and high-alloy castings. These families are further classified into various sub-families based on the type of heat-treatment operations required. Figure 2 gives a sample classification of castings (jobs) for heat-treatment operation in the steel foundry. The castings (jobs) from different families *cannot* be processed together in the same batch due to technical reasons, such as type of alloy, temperature level and the expected combination of heat-treatment operations. These job families are therefore mutually incompatible for processing together.

Trinder & Watts (1973) indicated that individual centers at the post-casting stage could be scheduled separately. Hence, in this paper we have considered the scheduling of heat-treatment furnaces in a steel-casting foundry, a special problem of batch processor scheduling, as an independent problem worthy of investigation. Furthermore, a major concern of foundry production management is to maximize throughput and reduce flow time and WIP. This motivated the choice of maximizing the utilization of the batch processors as the primary scheduling objective and minimizing the overall flow time and the average waiting time per job as secondary objectives in this study.

In the following section, we present the problem definition and assumptions. Section 3 reviews previously reported work on scheduling batch processors (BP). Section 4 presents briefly the heuristic algorithms proposed for this specific scheduling problem. We then present the computational experiments carried out to compare the performance of the heuristics with the estimated optimal solution and evaluate their relative effectiveness based on various performance measures in § 5. A summary and discussion of future research directions concludes the paper.

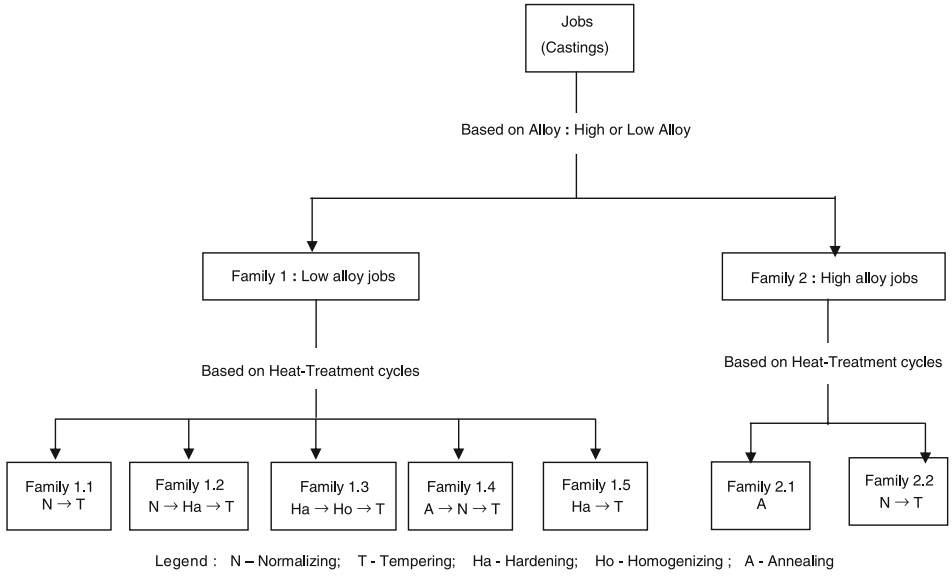

Figure 2. A sample classification of job-families for heat-treatment operations.

2. Research problem definition and assumptions

2.1 *Definition*

Suppose that there are F, $F \geq 1$, job-families of which a family f contains $N(f)$ jobs with different size $S(j, f)$ and different priority $P(j, f)$, for all $1 \leq f \leq F$ and $1 \leq j \leq N(f)$. In addition, a job 'j' in family 'f' has the same processing time $PT(f)$. Due to technical reasons, it is not possible to process jobs from different families together in the same batch. We shall call these job-families incompatible. Furthermore, these jobs will have to be processed without interruption on parallel and non-identical BPs (BPs with different capacities), which are available continuously with an objective of maximizing the utilization of the BPs.

2.2 *Assumptions*

- At the beginning of every fixed interval of time (in our analysis, every 24 hours[1]) from the starting time of scheduling, a number of jobs will arrive for operations at the BPs (that is, a full knowledge is available about future arrival of jobs).
- Scheduling planning period is one week. At the beginning of the planning period, the number of castings corresponding to first day of the planning period is in WIP. Furthermore, the entire jobs corresponding to the planning period have to be scheduled first before considering the jobs associated with the next planning period and no re-scheduling is allowed.
- All batch processors are continuously available and all jobs must pass through the operation(s) to be carried out at the BPs.

[1]This parameter is given as a variable in the computer implementation of the proposed greedy heuristics and thus the algorithms accepts new jobs dynamically in any fixed interval of time.

- Any job in the WIP for processing at BPs can be processed in any one of the parallel, non-identical BPs (BPs with different capacities).
- The batch size of the BP is independent of the shape of a job but is dependent on the size of a job and the capacity of the BP.
- The number of trays in which the jobs are normally placed for a specific operation at the BP are assumed to be unlimited and, thus, do not affect the scheduling decisions.
- 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.
- The set-up times of the operations are included in the processing time and are sequence independent.
- Machine breakdowns are ignored and manpower of uniform skill is continuously available.
- Processing time of job-families is considered constant and independent of the number of jobs in a batch.

3. Related work

Though considerable literature is available on the technical aspects of casting processes, there is scant treatment of foundry operation scheduling and the application of Operations Research methodologies to foundry scheduling [Voorhis *et al* (2001) and Shekar (1998)]. Law & Green (1970) demonstrated the use of computers for foundry scheduling and production control with small numerical examples, applied separately to melting, core making, molding, casting, annealing, and finishing processes. Scheduling of the total foundry production system is not dealt with and the scalability for practical application is not discussed. Further, Trinder & Watts (1973) outlined the general facets of the production control systems as currently found in many foundry organizations and discussed in general how computer-aided systems might improve on this situation.

Without detailing any model development, a working group of the Institute of British Foundrymen wrote a series of articles (Law *et al* 1983, 1985, 1988–1990) on topics like production control, functional overview and database requirement, production planning and scheduling, production monitoring and data capture, and management information. Southall & Law (1980) discussed some approaches to improving job scheduling in foundries; and Trinder & Moss (1984) discussed the necessity of real-time systems for foundry production control. These articles provide some broad requirements for production planning and control systems for foundries. Further, Law *et al* (1985) presented five factors to indicating that the day-to-day implementation of production scheduling, planning and control in a foundry is a difficult task.

Ram & Patel (1998) modelled the heat treatment furnace operations of manufacturing plant using simulation and heuristics. The heuristic part of the model provides a decision support to the furnace operator to help as to which orders to load and to find a possible match of orders that can be nested together in the batch to increase furnace utilization.

To the best of our knowledge, no study (other than our own earlier study) has addressed the scheduling of heterogeneous heat treatment furnaces under conditions such as incompatible job families, dynamic job arrivals and non-identical job sizes, as observed in the steel-casting foundry. The objective of our earlier study was to illustrate the operations of the proposed algorithms in comparison with our own earlier four algorithms. This was accomplished using a single problem configuration for purpose of evaluation. We could not however firmly generalize a conclusion, as at that time rigorous evaluation procedures for heuristics had not

evolved. In this paper, we will use more in-depth experimental evaluation procedures for the heuristics developed. Accordingly, we interacted with the management of a large-scale steel casting industry in Tamil Nadu, India (because of confidentiality issues, the name of the foundry is not quoted) and developed an experimental design, which is very close to the reality, to evaluate the proposed heuristic algorithms. The evaluation of the proposed algorithm is carried out using the estimated optimality principle.

Though no study is reported on scheduling of heat treatment furnaces, there are some studies related to other similar industries such as semiconductor manufacturing. A brief review of deterministic scheduling of BP with incompatible job-families is given in table 1. Though, these studies are closely related, it is observed that in the semiconductor manufacturing, the capacity of the BP is defined by the number of jobs whereas in steel casting, each job has a certain capacity requirement and the total size of a batch cannot exceed the capacity of the BP.

4. Heuristic algorithms

4.1 *Decision problem*

The decision problem defined in this paper involves three interrelated sets of decisions: (1) Batch processor selection (BPS)—selection of a BP from the available parallel, non-identical batch processors for the next scheduling; (2) Job-Family selection (JFS)—selection of a job-family from the given set of incompatible job-families for processing in the selected BP; and (3) Batch construction (BC)—selection of a set of jobs from a selected job-family to form a batch for the selected BP.

Dobson & Nambimadom (2001) proved that the configuration of the problem defined in their paper is NP-hard. Since the problem defined in this paper subsumes the configuration of the problem defined in Dobson & Nambimadom (2001), it is quite difficult to optimize exactly. Thus, there is good reason to look for heuristic and approximate methods that will produce solutions that are efficient and effective.

4.2 *Heuristic algorithms*

The heuristics proposed are related to managerial considerations observed mostly in the scheduling of heat treatment operations in steel casting manufacturing. That is, (a) selecting a BP which has maximum capacity when a tie occurs for scheduling jobs at time 't' so that maximum number of jobs will be completed as early as possible, (b) the jobs with high priority and also with the maximum job-size should be completed, so that maximum repletion of working capital is achieved.

There are four heuristic algorithms proposed for the scheduling problems described in this paper with a primary scheduling objective of maximizing average utilization of batch processors (AUBP). In each of the variants of the four heuristic algorithms, we kept a fixed heuristic for both decisions of 'batch processor selection' and 'batch construction' and vary only the heuristics for the decision of job family selection. The variation in the heuristics for 'job-family selection' is based on four criteria and they are: (1) the weighted average job-size of job-family, (2) the weighted average job-priority of job-family, (3) the simple average job-priority of job-family, and (4) the simple average job-size of job-family.

Further, for the maximum utilization obtained from the heuristics the overall flow time (OFT) and the weighted average waiting time (WAWT) are computed as secondary scheduling objectives, which will also be used for evaluating the performance of the proposed heuristic algorithms.

Table 1. A review on scheduling BPs with incompatible job-families.

	Scheduling		
Researcher	Problem (Application)	Algorithm	OBJ.
Uzsoy (1995)	Single BP with identical job sizes and assuming that all the jobs are available at time = zero. Also single BP and parallel identical BPs with identical job sizes with dynamic job arrivals (Semiconductor industry)	Proposed exact and approximate solution procedures	2, 4 & 5
Fanti *et al* (1996)	Single BP with identical job sizes and assuming that all the jobs are available at time = zero. (Shoe manufacturing)	Proposed heuristic algorithm	8
Hung (1998)	Single BP as well as parallel identical BPs with identical job sizes and assuming that all the jobs are available at time = zero (Semiconductor industry)	Dynamic programming formulation was proposed	3
Kempf *et al* (1998)	Single BP with different job sizes and assuming that all the jobs are available at time = zero (Semiconductor industry)	Several heuristic algorithms were proposed	1 & 5
Mehta & Uzsoy (1998)	Single BP with identical job sizes and assuming that all the jobs are available at time = zero. They consider an additional constraint in addition to the default capacity constraint of the BP. (Semiconductor industry)	Developed DP algorithms and provided heuristic algorithms	3
Kim *et al* (2000)	Single BP with identical job sizes and assuming that all the jobs are available at time = zero. (Semiconductor industry)	Heuristic algorithm was proposed	3
Azizoglu & Webster (2001)	Single BP with different job sizes and assuming that all the jobs are available at time = zero. (Semiconductor industry)	Proposed a branch and bound procedure for BP model discussed in Dobson & Nambimadom (2001)	2
Dobson & Nambimadom (2001)	Single BP with different job sizes and assuming that all the jobs are available at time = zero (Semiconductor industry)	Integer program model was developed; proved that the problem is NP hard and proposed a number of heuristic algorithms	7
Zee *et al* (2001)	Parallel non-identical BPs with identical job sizes and with dynamic job arrivals. (Aircraft industry)	Proposed a heuristic algorithm	9

Objectives: 1 = Min. completion time; **2 =** Min. weighted completion time; **3 =** Min. total tardiness; **4** = Min. maximum lateness; **5** = Min. makespan**; 6 =** Min. total weighted tardiness **7** = Min. weighted flow time; **8** = Max. BPs' utilization; **9** = Min. logistics cost; **10** = Min. number of tardy jobs

Table 1. (Continued).

	Scheduling		
Researcher	Problem (Application)	Algorithm	OBJ.
Jolai (2005)	Scheduling of a single BP assuming that (a) all jobs are available at time = zero, and (b) jobs of the same family are indexed in non-decreasing order of their due dates. (Semiconductor industry)	Proved that this problem is NP-hard and proposed Dynamic Programming algorithm	10
Koh *et al* (2004 & 2005)	Scheduling of 'bake-out' operation in the MLC manufacturing process with different volumes of jobs. (Multi-layer ceramic capacitor manufacturing)	Proposed IP model and a number of heuristics and genetic algorithms	1, 2, 5
Monch *et al* (2005)	Scheduling of parallel batch machines with unequal ready times of the jobs (i.e with dynamic job arrivals). (Semiconductor industry)	Proposed two different decomposition approaches based on Genetic Algorithm and simple heuristic algorithms	6
Perez *et al* (2005)	Single BP with different job sizes and assuming that all the jobs are available at time = zero. (Semiconductor industry)	Proposed heuristic algorithm	6
Monch *et al* (2006)	Scheduling BPs found in the diffusion and oxidation areas of semiconductor wafer fabrication facilities with dynamic job arrivals. (Semiconductor industry)	Proposed heuristic algorithms along machine learning techniques for estimating values of parameters.	6
Malve & Uzsoy (2007)	Parallel identical BPMs with n jobs (a) representing multiple and incompatible job-families, and (b) having different release time and due date. (Semiconductor industry)	Proposed genetic algorithm	4

4.2a *Formula for scheduling objective – Average utilization of the batch processors (AUBP):*

$$\mathbf{AUBP} = \frac{\sum_{BP=1}^{NBP} \{CAP(BP) * UT(BP)\}}{\sum_{BP=1}^{NBP} CAP\ (BP)}$$

where

$$UT(BP) = \frac{\sum_{BP=1}^{NB(BP)} Tot\_Job\_Size\ (B, BP)}{NB(BP) * CAP(BP)} \qquad \text{For all } \mathbf{BP}$$

$$Tot\_Job\_Size\ (B, BP) = \sum_{J=1}^{NJ(B,BP)} Size\ (J, B, BP) \qquad \text{For all } \mathbf{B} \text{ and For all } \mathbf{BP}$$

4.2b *Formula for scheduling objective – Overall flow time (OFT):*

$$\textbf{OFT} = \text{Max}\{\text{End-Time (NB, BP)}, \text{BP} = 1, 2, \ldots\ldots \text{NBP}\}$$

4.2c *Formula for scheduling objective – Weighted average waiting time (WAWT):*

$$\textbf{WAWT} = \frac{\sum_{\text{BP}=1}^{\text{NBP}} \text{CAP(BP)} * \text{TAWT(BP)}}{\sum_{\text{BP}=1}^{\text{NBP}} \text{CAP (BP)}}$$

where

$$\text{TAWT (BP)} = \sum_{\text{B}=1}^{\text{NJ(BP)}} \text{AWT (B, BP)} \qquad \text{For all } \textbf{B} \text{ and } \textbf{BP}$$

$$\text{AWT(B, BP)} = \{\text{TWT(B, BP)/NJ (B, BP)}\} \qquad \text{For all } \textbf{B} \text{ and } \textbf{BP}$$

$$\text{TWT (B, BP)} = \sum_{\text{J}=1}^{\text{NJ(B,BP)}} \text{WT (J, B, BP)} \qquad \text{For all } \textbf{B} \text{ and } \textbf{BP}$$

$$\text{WT(J, B, BP)} = \text{ST-TIME(B, BP)} - \{(\text{Day(J)} - 1) * 24\} \qquad \text{For all } \textbf{J}, \textbf{B} \text{ and } \textbf{BP}$$

4.3 *Heuristic algorithm 1 – A1*

Decision 1: Batch processor selection – (BPS):

Step 1. Whenever 'tie' occurs in selecting a BP for scheduling, select a BP, which has a maximum capacity; When there is no 'tie' situation for selecting a BP for scheduling, select the batch processor which is going to be available early for the next scheduling.

Decision 2: Job-family selection – (JFS):

Step 2. For each family, temporarily construct a 'batch' using the following steps:

Step 2.1. Sort, all the jobs of the job-family 'F' based on *job's arrival day* in ascending order, and within a day based on *job's priority* in ascending order, and all daily jobs within a priority based on *job's size* in descending order.

Step 2.2. Select a set of feasible jobs from the top of a 'sorted-list' until the selected BP capacity is utilized to the maximum extent.

Step 2.3. Do steps 2·1 and 2·2 for all job-families. Further, store temporarily all the details of the jobs considered for the batch of jobs corresponding to each family for the selected batch processor.

Step 2.4. Compute for each job-family the index, $INDEX(F) = \{PT(F)/\,WASJ(F, BP)$

Step 3. Select a job-family, which has the *Min* $\{$*INDEX (F)*, $F = 1, 2, \ldots NF\}$.

Decision 3: Batch construction – (BC):

Step 4. Confirm the temporarily constructed batch corresponding to the selected job-family as per step 3 for scheduling at time 't'.

Step 5. Update the 'time matrix' (for each batch processor, a time matrix is created and this matrix contains information on starting time of each batch and ending time of the corresponding batch) of batch processor availability and the 'sorted-list' of the selected job-family.

Step 6. Repeat steps 1 to 5 until all jobs for the given planning period (= one week) are scheduled.

Note: Heuristic algorithms 2 to 4 that follow are identical to '**A1**' except in Step 2·4 of Step 2 of the algorithm. Therefore, only the *modified Step 2·4* is detailed below for each of the variants:

4.4 *Heuristic algorithm 2 – A2*

Step 2.4. Compute for each job-family the index, $INDEX(F) = \{PT(F)/WAPJ(F, BP)\}$

4.5 *Heuristic algorithm 3 – A3*

Step 2.4. Compute for each job-family the index, $INDEX(F) = \{PT(F)/APJ(F, BP)\}$

4.6 *Heuristic algorithm 4 – A4*

Step 2.4. Compute for each job-family the index, $INDEX(F) = \{PT(F)/ASJ(F)\}$

All the above four variants of the greedy heuristic were implemented using programming language Turbo C++.

5. Computational experiments

A computational experiment is appropriate in order to provide a perspective on the relative effectiveness of any proposed heuristic algorithm (Baker 1999). In this study, computational experiments were carried out with two objectives. The first objective was to evaluate the absolute quality of the solutions obtained by the proposed heuristic algorithms by comparing them with estimated optimal solutions. The second objective was to compare the relative performance of the proposed heuristic algorithms in terms of both computational effort and quality of the solutions. The analysis of the experimental data is presented for both cases. An experimental approach of this type relies on two elements; an experimental design and a measure of effectiveness. These are discussed first as a prelude to the discussions of the evaluation of heuristics.

5.1 *The experiment*

5.1a *Measure of effectiveness:* The performance of the algorithms may vary over a range of problem instances. Therefore, the performances of the proposed heuristic algorithms are compared using the measures, viz., (1) average proximity (AP), (2) average relative percentage deviation (ARPD), indicating the average performance of heuristics and (3) maximum relative

Table 2. Summary of the experimental design.

Sl. No.	Problem factor	Levels	# Levels
1	Number of jobs (n)	L1: $n = 861 =$ {123, 123, 123, 123, 123, 123, 123} L2: $n = 943 =$ {125, 132, 144, 123, 150, 142, 127} L3: $n = 1003 =$ {123, 180, 143, 157, 130, 140, 130} L4: $n = 1107 =$ {152, 144, 168, 163, 135, 176, 169} L5: $n = 1260 =$ {180, 180, 180, 180, 180, 180, 180}	5
2	Job-priority (P)	L1 : $P \in [1, 8]$ with equal probability of assigning each of the priorities. L2 : $P \in [1, 8]$ with un-equal probability of assigning each of the priorities.	2
3	Job-family (F)	L1 : $F \in [1, 5]$ with equal probability of assigning each family ID. L2 : $F \in [1, 5]$ with un-equal probability of assigning each family ID.	2
Problem configurations			$5 \times 2 \times 2 = 20$
Problem instance per configuration			15
Total problem instances			300

percentage deviation (MRPD), indicating the worst case performance of heuristics. Further, these measures are defined as follows:

$$\mathrm{AP(k)} \equiv \left\{ \sum_{\mathrm{i=1}}^{\mathrm{N}} (\mathrm{U_1} - \mathrm{U_K}) \right\} \Big/ N$$

$$\mathrm{ARPD(k)} = \left\{ \sum_{\mathrm{i=1}}^{\mathrm{N}} [\,(\mathrm{U_1} - \mathrm{U}_K) / \mathrm{U_1}] \times 100 \right\} \Big/ N$$

$$\mathrm{MRPD(k)} \equiv \underset{\mathrm{N}}{\mathrm{Max}} \; \{[(\mathrm{U_1} - \mathrm{U_K}) / \mathrm{U_1}] \times 100\}$$

5.1b *Experimental design:* The experimental design is the process of planning an experiment to ensure that the appropriate data will be collected/generated. We identified three important problem parameters based on our observation made in the steel casting industry. They are number of jobs (n), job-priority (P), and job-family (F). It was learnt during the interaction with a user organization that these parameters may affect the performance of the heuristic algorithms. Accordingly, an experimental design was developed to study the performance of the proposed heuristic algorithms, the summary of which is given in table 2.

The parameter, number of jobs 'n' (which indicates the load on heat-treatment furnaces), was assumed based on the field data collected over one week period for heat treatment furnaces of a steel casting industry. Out of five levels of 'n', one is related to the observed week (that is, $n = 1003$ jobs), two are related to two extremes of the observed week (that is, extreme 1: 123 jobs per day $\times$ 7 days and the extreme 2: 180 jobs per day $\times$ 7 days), and the other

two levels are randomly decided. The number of jobs, proposed in this design exceeds all the computational experiments reported in the literature of batch processor scheduling problem, including the recent one of Qi & Tu (1999).

Further, it is assumed that the size of the jobs vary and are uniformly distributed between (100 Kg, 1000 Kg), as most of the job-sizes fall within this range, in the observed steel casting industry. Furthermore, the uniform distribution was chosen because it is a relatively high-variance distribution, which would allow the heuristics to be tested under conditions relatively unfavourable to them (Chandru *et al* 1993).

It was observed that the foundry assigned priorities (P in table 2) right at the beginning. The management uses three criteria (value, alloy type and market status of the order) with two alternatives for each criterion (that is, high value vs. low value; high alloy vs. low alloy; and export order vs. domestic order) for assigning a priority to a job. Thus, the priority varies from 1 to 8. It was observed that each of the priorities (1 to 8) is not assigned always with equal probability. For example, out of (say) 180 jobs expected to arrive for heat-treatment operations, 30 jobs has priority 1, 20 jobs has priority 2, 35 jobs has priority 3, 45 jobs has priority 4, 20 jobs has priority 5, 10 jobs has priority 6, 20 jobs has priority 7 and no jobs with priority 8. This type of non-uniformly distributed priorities has an influence on the criterion to be used for job-family selection at time 't' for scheduling. Thus, the parameter 'P' has two levels in the experiments.

The parameter, job-family 'F' used in table 2 is based on the observed job families in the foundry, where the jobs are classified into five different groups (families). These five families are based on the expected combination of heat treatment operations required by the job. It was also observed that, over a long period, jobs were not equally distributed over the families. For example, out of (say) 180 jobs expected to arrive for heat-treatment operations, 50 jobs belong to job-family 1; 30 jobs belong to job-family 2; 35 jobs belong to job-family 3; 45 jobs belong to job-family 4 and 20 jobs belong to job-family 5. This type of non-uniformly distributed job-families has an influence on the criterion to be used for job-family selection at time 't' for scheduling. Thus, the parameter 'F' has two levels in the experiments.

The experimental design for generating test problems was implemented in programming language Turbo C++. For each combination of values for $\{n, P, \text{and } F\}$ 15 problem instances were randomly generated, yielding a total of 300[$= 5 \times 2 \times 2 \times 15$] problem instances.

In addition to the input parameters mentioned in table 2, we assume, based on the observation made in the foundry, that there are two batch processors with capacities: 1500 Kg and 5000 Kg respectively, and five incompatible job-families with the processing times: 13, 9, 8, 7, and 10 Hrs., respectively, for scheduling.

5.2 *Absolute evaluation of heuristic solutions*

In this evaluation, heuristic solutions of the proposed algorithms are compared with an estimated optimal solution. There are many procedures available in the literature for estimating optimal value for combinatorial optimization problems. We have used the procedure discussed in Rardin & Uzsoy (2001) for the statistical estimation (based on the Weibull distribution) of optimal (minimum) value (with reference to a problem having minimization as the objective), appropriately for our maximization situation. Accordingly, for obtaining an estimated optimal solution for each problem instance, we need to generate a number of feasible solutions. To achieve this, a procedure is developed (which is named as RSA) and the systematic procedure of this algorithm is as follows:

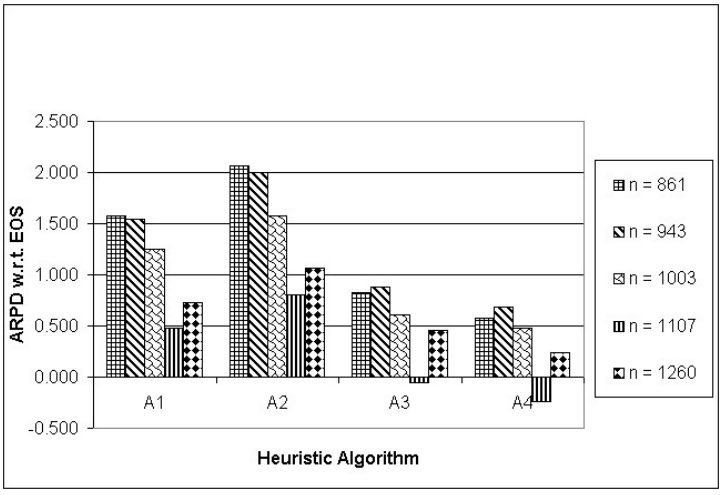

Figure 3. Average performance of heuristics [Average {ARPD (AUBP)} with respect to estimated optimal solution (EOS)].

Step 1. Randomly select a BP whenever 'tie' occurs in selecting BP for next scheduling. Otherwise, select the one, which is going to be available for next scheduling.

Step 2. Randomly select a job-family from a set of feasible job-families. The criterion of feasibility may be a threshold of perhaps such as 'above 75%' of the selected BP's capacity, when we add the entire jobs in the selected job-family.

Step 3. As given in heuristic algorithm 1 for the decision: Batch construction [BC].

The RSA was coded in programming language Turbo C++. For each of the randomly generated 300 problem instances, 15 feasible solutions (i.e., the average utilization of batch processors (AUBP) were obtained using the RSA. The 15 feasible solutions obtained using RSA were used to estimate the optimal solution using the procedure highlighted in Rardin & Uzsoy (2001). It is to be noted that the generated 15 feasible solutions using the RSA are expected to provide the estimated optimal solution, [that is, estimated optimal AUBP] of the problem instance with a very high probability of approximately 0·9999996941 $[= (1-e^{-15})]$.

Further, for each variant of the heuristic and for each problem instance belonging to the level of (n, P, and F), the solution of maximum AUBP is obtained. For the maximum AUBP obtained for each problem instances, the corresponding minimum weighted average waiting time (WAWT) and minimum overall flow time (OFT), were also computed.

5.2a *Results:* For the primary scheduling objective of maximizing the average utilization of the batch *processors, the value of 'ARPD' and 'MRPD' were computed with respect to each level of* (n, P, and F). Furthermore, for each 'n' with all levels of (P, and F) the average of {ARPD} and the maximum of {MRPD} were computed. These average of {ARPD} and maximum {MRPD} are shown in figures 3 and 4 respectively.

From the perspective of average performance of the heuristics (figure 3), a negative ARPD indicates that on average, the corresponding heuristic algorithm found a better result than the estimated optimal value. From this, it is possible to conclude that the heuristic algorithms A3 and A4, on average yielded better utilization than the estimated optimal value. This

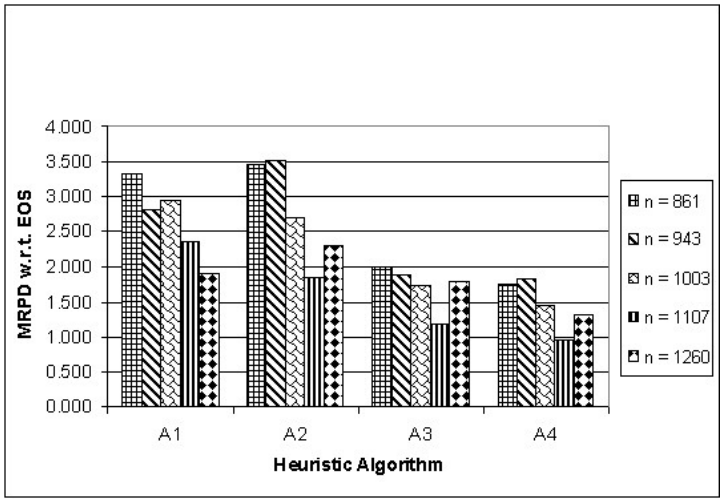

Figure 4. Worst case performance of heuristics [Maximum {MRPD (AUBP)} with respect to estimated optimal solution (EOS)].

indicates that the Weibull-based technique may have yielded conservative estimates of the optimal value. Further, a positive ARPD indicates that the heuristic algorithm found inferior solutions when compared with the estimated optimal value. However, the deviation is very small. In addition, same inference can be drawn using the worst-case performance analysis (see figure 4).

From the figures 3 and 4, it can be observed that, on average, the difference between the solution obtained from the heuristic algorithms and the estimated optimal value is not significant, and therefore we may infer that all the proposed heuristic algorithms are highly satisfactory heuristics to achieve maximum utilization of the batch processors.

5.3 *Relative evaluation of heuristics algorithms*

The relative performance of the proposed heuristics is discussed in terms of their computational effort and quality of the solutions relative to each other.

5.3a *Computational effort:* The computational time (in seconds) on a Pentium 200 MHz processor required for each algorithm for solving five sets of 60 problem instances together, corresponding to each level of n, is reported in table 3. From the table 3, it is clear that the computational burden increases with the number of jobs considered for scheduling. Though it increases, on average, from the point of view of computational burden of the heuristics, it appears that the proposed heuristic algorithms will take relatively low computational times even for very large size problems. Thus, from the point of view of computational effort, we can conclude that one could run several heuristics, proposed in this paper, on each problem instance and could take the best solution found.

5.3b *Quality of the solution:* For each problem instance, every heuristic produces (a) maximum AUBP, which is the primary objective, (b) corresponding minimum OFT and minimum WAWT. Furthermore, average and standard deviation of AUBP as well as average proximity and average standard deviation of proximity (which indicates the average performance of the

Table 3. CPU time (in seconds) required to obtain the solutions of a set of 60 problem instances.

Number of problem instances	# jobs (n) per instance	CPU Time for the heuristic algorithms			
		A1	A2	A3	A4
60	861	339	356	339	344
60	943	386	476	388	391
60	1003	432	531	435	440
60	1107	511	628	513	517
60	1260	645	787	648	644

heuristics), were computed for each level of factors (n, P and F). The results are shown in table 4 (related to AUBP) and appendices 1 and 2 (related to OFT and WAWT respectively).

On average, if we observe the quality of the solution for the primary scheduling objective of maximizing AUBP, the difference between the best solution and the worst solution is not substantial (see table 4). Therefore, it is possible to conclude that any heuristics can be chosen from the four (proposed). However, for the secondary scheduling objectives, minimizing OFT and minimizing WAWT, the differences in performance are reasonably significant (appendixces 1 & 2). This particular observation becomes obvious from the results, when problem factor 'n' increases.

Further, the average of (a) 'maximum AUBP', (b) 'minimum OFT' and (c) 'minimum WAWT' were computed over 'n'. The result is shown in table 5. From the table 5, it may be useful to consider heuristics that are inferior with respect to AUBP but superior with respect to both OFT and WAWT. That is, if we were to select a single heuristic it is clear that the trade-offs between AUBP, OFT and WAWT will be in favour of heuristic A1. That is, on average, the criterion, viz. *'the weighted average job-size of job-family, with priority as weight'*, used for job-family selection is expected to yield an acceptable and efficient solution for the three scheduling objectives addressed in this paper. The same inferences hold for the results obtained based on average performance analysis using average relative percentage deviation as well as based on the worst-case performance analysis using the value of maximum relative percentage deviation.

From the detailed results obtained on primary as well as secondary scheduling objectives, it is observed that there is an influence of problem instance on the performance of the heuristic algorithms (that is, there is variability in the performance of the heuristics with the problem factors (n, P and F)]. This inference is further verified using the analysis of variance (ANOVA) technique. The window-based statistical package 'SIGMASTAT' is used for this ANOVA. The package initially tests for the assumptions behind the ANOVA-analysis. Since, our data failed in the normality test but passed in the equal variance test, the package suggested for non-parametric analysis. Accordingly, the package constructs the required rank matrix from the basic data and then does the non-parametric ANOVA. It is observed from the result obtained from the package that the problem factors influence the performance of heuristic algorithms.

6. Conclusion

This paper has examined the problem of scheduling jobs on parallel, non-identical batch processing machines with incompatible job-families and non-identical job sizes to maximize

Table 4. Performance of the heuristics – primary scheduling objective: AUBP.

Problem factor				AUBP by heuristic algorithm				Proximity in % by heuristic algorithm			
n	*P*	*F*	Criterion	A1	A2	A3	A4	A1	A2	A3	A4
861	L1	L1	Avg.	94·1	93·8	94·5	94·4	0·8	1·0	0·3	0·4
	L1	L2		94·3	93·8	94·8	95·0	0·9	1·4	0·4	0·2
	L2	L1		94·5	94·1	94·5	94·5	0·4	0·7	0·3	0·3
	L2	L2		94·0	94·0	94·2	94·4	0·6	0·6	0·4	0·2
	L1	L1	Std. dev	2·0	2·3	1·9	2·0	0·6	0·7	0·4	0·5
	L1	L2		1·8	1·9	2·0	2·1	0·8	0·6	0·4	0·3
	L2	L1		2·0	2·3	2·1	2·1	0·3	0·4	0·5	0·4
	L2	L2		2·0	2·1	2·1	2·0	0·6	0·6	0·3	0·4
943	L1	L1	Avg.	96·1	95·9	96·0	96·0	0·3	0·5	0·5	0·5
	L1	L2		96·1	95·8	96·6	96·6	0·7	1·0	0·2	0·2
	L2	L1		96·0	95·7	95·9	95·8	0·3	0·7	0·4	0·6
	L2	L2		96·2	96·0	96·4	96·3	0·6	0·7	0·3	0·5
	L1	L1	Std. dev	0·4	0·6	0·6	0·5	0·4	0·4	0·6	0·3
	L1	L2		0·9	0·9	0·4	0·4	0·8	1·0	0·4	0·2
	L2	L1		0·7	0·6	0·4	0·4	0·4	0·6	0·5	0·5
	L2	L2		0·7	0·5	0·4	0·5	0·7	0·7	0·4	0·5
1003	L1	L1	Avg.	96·2	96·0	95·9	96·0	0·2	0·4	0·5	0·4
	L1	L2		96·2	95·9	96·1	96·3	0·3	0·6	0·4	0·2
	L2	L1		96·3	96·0	96·1	96·1	0·1	0·4	0·3	0·3
	L2	L2		96·3	96·1	96·3	96·3	0·2	0·4	0·3	0·2
	L1	L1	Std. dev	0·5	0·4	0·4	0·3	0·3	0·2	0·3	0·4
	L1	L2		0·8	0·5	0·3	0·2	0·5	0·3	0·2	0·3
	L2	L1		0·5	0·3	0·3	0·3	0·2	0·3	0·3	0·3
	L2	L2		0·5	0·5	0·4	0·4	0·3	0·4	0·2	0·3
1107	L1	L1	Avg.	96·4	96·0	96·5	96·5	0·4	0·7	0·2	0·3
	L1	L2		96·6	96·3	96·8	96·9	0·5	0·8	0·3	0·2
	L2	L1		96·4	96·2	96·5	96·4	0·5	0·7	0·4	0·4
	L2	L2		96·6	96·2	96·8	96·8	0·4	0·7	0·1	0·2
	L1	L1	Std. dev	0·5	0·5	0·4	0·4	0·5	0·6	0·3	0·4
	L1	L2		0·5	0·6	0·4	0·4	0·4	0·6	0·3	0·3
	L2	L1		0·4	0·6	0·6	0·6	0·4	0·6	0·4	0·5
	L2	L2		0·5	0·5	0·4	0·4	0·3	0·6	0·3	0·2
1260	L1	L1	Avg.	96·2	96·3	96·7	96·6	0·7	0·6	0·2	0·3
	L1	L2		95·5	95·6	96·7	96·7	1·4	1·2	0·1	0·1
	L2	L1		96·4	96·3	96·7	96·9	0·6	0·7	0·3	0·1
	L2	L2		95·6	96·0	96·8	96·7	1·4	1·1	0·2	0·3
	L1	L1	Std. dev	0·4	0·4	0·4	0·3	0·5	0·5	0·3	0·3
	L1	L2		0·6	0·6	0·3	0·4	0·6	0·7	0·2	0·2
	L2	L1		0·4	0·6	0·4	0·2	0·4	0·5	0·3	0·2
	L2	L2		0·5	0·6	0·4	0·4	0·6	0·6	0·3	0·4

the average utilization of batch processing machines, and has suggested a few fast and efficient heuristics. The motivation for this research is the heat-treatment operation at the post casting stage of steel casting manufacturing. This problem is of considerable practical value, because the heuristics proposed in this paper can be used in modelling a large number of heat-treatment

Table 5. Net average performance of heuristics and scheduling objectives.

# Jobs (n)	Scheduling objective	Average performance of the heuristic algorithm				{Best solution-worst solution}
		A1	A2	A3	A4	
	A	95·0	94·5	95·7	**96·0**	1·5%
861	B	737	741	731	**727**	4 hrs
	C	**60**	**60**	72	73	13 hrs
	A	95·3	94·9	96·0	**96·1**	1·2%
943	B	804	808	799	**796**	12 hrs
	C	**66**	67	80	81	15 hrs
	A	95·6	95·3	96·2	**96·3**	1·0%
1003	B	854	857	849	**846**	11 hrs
	C	**72**	**72**	86	88	16 hrs
	A	96·0	95·6	96·5	**96·6**	1·0%
1107	B	939	943	934	**930**	13
	C	**80**	**80**	97	98	18 hrs
	A	96·4	96·0	96·6	**96·8**	1·8%
1260	B	1060	1066	1058	**1054**	12 hrs
	C	**93**	**93**	111	113	20 hrs

A - Average {Maximum AUBP} in %; B - Average {Minimum OFT} in Hrs; C - Average {Minimum WAWT} in Hrs

operations as well as chemical processing operations such as diffusion and oxidation in wafer fabrication, hardening of synthetic parts in aircraft industries, etc.

On comparison with the statistically estimated optima (based on the Weibull technique), it appears that all the proposed heuristic algorithms are, on average, better ones for scheduling large scale heterogeneous batch processors in the presence of dynamic job arrivals with incompatible job-families and non-identical job sizes. From the point of view of computational effort, we can further conclude that one could run several heuristics, proposed in this paper, on each problem instance and could take one that gives the best solution. Further, the heuristic algorithm A1, particularly, the job-family selection based on *'weighted average job-size of job-family with priority as weight'*, turns out to be the best choice if we trade-off all three scheduling objectives maximizing AUBP, minimizing OFT and minimizing WAWT, considered in this study. Finally, it is observed from the results (as well as statistically verified) that the performance of the heuristic is seen to be sensitive to the problem factor (n, P, and F) used in the experimental design.

There are a number of interesting extensions of the problems that can be pursued. One interesting extension is to evaluate whether the inferences obtained in this study are stable when we allow the changes in the input parameters such as (a) job-size distribution, (b) processing time for each incompatible job-family, and (c) a specific capacity combination of the two non-identical batch processors. Additional important extensions could be (i) to include the due date related performance measures in the model, and (2) relaxing some of the assumptions, mentioned in this paper, one-by-one, and studying its impact on the proposed algorithms (for example, studying the impact of relaxing the first assumption mentioned in this paper could be an interesting extension. That is, changing the current assumption from '*every 24 hours*, a set of new jobs arrive to WIP area' to various input such as 'every 3 hours', '6', 9, '12 hours', etc.). Finally, the proposed heuristic algorithms in this paper can be extended by incorporating today's standard job shop or assembly scheduling rules to sequence the batches, constructed by the proposed heuristic algorithms for other scheduling criteria based on completion time.

This work was partly supported by the Singapore-MIT Alliance, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. The authors would like to thank anonymous reviewers for their helpful comments and suggestions.

Notations

WIP	Work-in-process inventory
BP	Batch processor 'BP' and $BP = 1, 2, \ldots NBP$
NBP	Maximum number of batch processors or last BP
B	Batch B and $B = 1, 2, \ldots NB$
AUBP	Average utilization of the batch processors
CAP(BP)	Capacity of the batch processor 'BP'
UT(BP)	Utilization of the batch processor 'BP'
NB(BP)	Number of batches, processed in 'BP'
Tot_Job_Size(B, BP)	Total job size of the batch 'B' of the batch processor 'BP'
NJ(B, BP)	Number of jobs in the batch 'B' of the batch processor 'BP'
Size(J, B, BP)	Size of job 'J' in the batch 'B' of the batch processor 'BP'
OFT	Overall flow time
End_Time(NB, BP)	Ending time of the last batch 'NB', processed in 'BP'
WAWT	Weighted average waiting time
TAWT(BP)	Total average waiting time of jobs, processed in 'BP'
AWT(B, BP)	Average waiting time per job in batch 'B', processed in 'BP'
TWT(B, BP)	Total waiting time of jobs of batch 'B', processed in 'BP'
WT(J, B, BP)	Waiting time of job 'J' of batch 'B', processed in 'BP'
ST-TIME(B, BP)	Starting time of batch 'B', processed in 'BP'
Day(J)	Arrival day of the job 'J'
MaxRun	Maximum number of runs
F	Job-family 'F' and $F = 1, 2, \ldots NF$
NF	Number of job-families
ASJ(F, BP)	Average job-size of job-family 'F' based on a subset or batch of jobs consistent with the selected 'BP'
APJ(F, BP)	Average job-priority of job-family 'F' based on a subset or batch of jobs consistent with the selected 'BP'
WASJ(F, BP)	Weighted average job-size of job-family 'F' based on a subset or batch of jobs consistent with the selected 'BP' with 'priority' as weight
WAPJ(F, BP)	Weighted average job-priority of job-family 'F' based on a subset or batch of jobs consistent with the selected 'BP' with 'job-size' as weight
AP(k)	Average proximity of heuristic 'k'
ARPD(k)	Average relative percentage deviation of heuristic 'k'
MRPD(k)	Maximum relative percentage deviation of heuristic 'k'
N	Number of problem instances
U_k	Average utilization of batch processors, yielded by k^{th} heuristic
U_1	Estimated optimal 'average utilization of batch processors' OR $U_1 = \max\{U_k, k = 1, 2, 3, 4)$ with respect to the type of evaluation

Appendix 1. Performance of the heuristics – secondary scheduling objective: OFT.

Problem factor				OFT by heuristic algorithm				Proximity by heuristic algorithm			
n	*P*	*F*	Criterion	A1	A2	A3	A4	A1	A2	A3	A4
861	L1	L1	Avg.	840	845	836	835	8	13	4	3
	L1	L2		935	942	931	929	9	16	5	2
	L2	L1		838	842	837	837	5	9	4	4
	L2	L2		976	978	974	970	8	10	6	2
	L1	L1	Std. dev.	81	84	81	80	7	8	5	4
	L1	L2		76	78	80	81	8	8	5	5
	L2	L1		81	82	81	81	6	6	6	4
	L2	L2		63	66	62	60	8	8	6	5
943	L1	L1	Avg.	817	818	818	817	5	5	6	4
	L1	L2		920	922	915	912	10	12	5	3
	L2	L1		817	821	818	820	3	8	5	6
	L2	L2		913	914	910	911	7	8	4	5
	L1	L1	Std. dev.	18	17	20	17	5	5	6	5
	L1	L2		18	21	20	18	10	11	6	4
	L2	L1		18	14	17	18	4	6	6	7
	L2	L2		17	14	15	14	9	7	6	5
1003	L1	L1	Avg.	845	850	847	845	4	9	6	4
	L1	L2		944	950	942	940	7	13	5	2
	L2	L1		842	850	844	844	2	9	3	3
	L2	L2		939	946	939	937	6	13	5	3
	L1	L1	Std. dev.	27	27	29	26	6	6	7	4
	L1	L2		22	20	22	19	9	7	5	4
	L2	L1		25	28	27	29	3	8	4	4
	L2	L2		26	25	26	30	6	9	5	6
1107	L1	L1	Avg.	944	949	942	942	6	10	4	4
	L1	L2		1061	1065	1058	1056	7	11	5	3
	L2	L1		945	947	943	942	6	8	4	3
	L2	L2		1051	1058	1048	1048	6	13	3	3
	L1	L1	Std. dev.	23	23	25	25	8	6	4	6
	L1	L2		30	28	27	27	6	9	6	4
	L2	L1		24	26	25	26	4	8	6	5
	L2	L2		15	16	16	17	6	8	4	4
1260	L1	L1	Avg.	1238	1239	1232	1232	9	9	3	3
	L1	L2		1297	1294	1285	1282	18	15	6	3
	L2	L1		1235	1240	1231	1230	8	12	4	2
	L2	L2		1296	1290	1279	1279	20	14	3	3
	L1	L1	Std. dev.	20	21	21	22	7	8	4	4
	L1	L2		26	24	22	18	10	11	6	4
	L2	L1		22	22	21	22	6	7	4	3
	L2	L2		19	20	16	18	6	9	4	5

Appendix 2. Performance of the heuristics – secondary scheduling objective: WAWT.

Problem factor				WAWT by heuristic algorithm				Proximity by heuristic algorithm			
n	*P*	*F*	Criterion	A1	A2	A3	A4	A1	A2	A3	A4
861	L1	L1	Avg.	75·2	78·4	82·4	83·2	0·4	3·6	7·6	8·4
	L1	L2		86·9	91·0	91·9	92·3	0·5	4·6	5·6	6·0
	L2	L1		75·7	75·5	82·9	84·4	1·5	1·2	8·6	10·1
	L2	L2		93·0	93·6	99·3	98·3	1·8	2·4	8·1	7·1
	L1	L1	Std. dev	13·2	15·4	14·4	14·6	0·7	4·1	3·7	3·6
	L1	L2		13·5	13·4	13·3	14·6	1·2	3·1	3·4	3·0
	L2	L1		13·3	13·5	14·8	15·5	1·8	3·1	3·8	3·7
	L2	L2		14·5	14·5	13·7	12·9	2·7	2·5	2·4	2·7
943	L1	L1	Avg.	71·0	71·2	77·0	78·2	2·1	2·2	8·0	9·2
	L1	L2		82·9	85·3	88·5	88·8	1·0	3·3	6·6	6·9
	L2	L1		70·2	70·6	77·9	79·4	1·2	1·5	8·8	10·3
	L2	L2		83·0	84·5	88·9	89·2	0·5	2·1	6·5	6·8
	L1	L1	Std. dev	4·1	4·6	4·3	3·3	2·7	4·2	3·0	2·1
	L1	L2		2·7	5·5	5·5	4·2	2·3	3·1	3·9	3·5
	L2	L1		3·3	2·7	3·5	3·5	1·8	2·5	2·4	2·8
	L2	L2		3·7	3·9	3·9	3·2	0·9	2·7	3·2	3·0
1003	L1	L1	Avg.	72·3	72·8	76·7	78·2	0·4	0·8	4·8	6·3
	L1	L2		84·6	86·3	88·5	89·5	0·1	1·9	4·1	5·0
	L2	L1		75·0	74·0	77·7	79·1	1·4	0·5	4·1	5·6
	L2	L2		85·4	86·4	88·9	90·0	0·2	1·2	3·7	4·8
	L1	L1	Std. dev	5·1	5·0	5·1	5·0	0·9	1·0	1·7	1·9
	L1	L2		5·0	4·7	4·4	4·2	0·3	1·6	2·5	2·6
	L2	L1		5·0	4·5	4·7	4·7	1·4	0·8	1·4	1·6
	L2	L2		4·7	4·9	5·0	5·3	0·3	1·5	1·6	1·9
1107	L1	L1	Avg.	82·8	82·1	90·1	91·8	1·7	1·0	9·0	10·6
	L1	L2		97·1	100·7	103·7	104·1	0·7	4·3	7·3	7·8
	L2	L1		83·6	83·2	91·0	92·0	1·7	1·2	9·1	10·1
	L2	L2		97·3	98·3	103·6	103·6	1·1	2·1	7·4	7·4
	L1	L1	Std. dev	4·8	3·6	3·8	4·5	2·8	1·8	2·7	2·2
	L1	L2		5·9	3·6	4·9	4·4	1·8	3·5	2·4	2·2
	L2	L1		4·4	5·1	5·3	5·1	2·7	2·1	3·0	3·8
	L2	L2		3·3	4·8	4·8	4·2	2·1	3·6	3·1	3·6
1260	L1	L1	Avg.	109·6	110·1	134·1	136·3	2·3	2·9	26·9	29·1
	L1	L2		121·4	121·9	134·0	137·3	2·2	2·7	14·8	18·1
	L2	L1		109·0	107·9	133·6	136·7	2·2	1·1	26·8	29·9
	L2	L2		121·7	118·7	134·3	137·5	3·8	0·8	16·4	19·6
	L1	L1	Std. dev	3·6	5·7	5·6	4·1	3·1	4·8	5·6	3·9
	L1	L2		5·1	6·0	3·7	3·8	2·2	6·3	4·0	3·2
	L2	L1		4·5	4·2	4·8	5·7	2·6	2·2	4·1	4·6
	L2	L2		3·8	3·4	3·9	3·9	3·7	1·7	3·7	3·1

References

Azizolglu M, Webster S 2001 Scheduling a batch processing machine with incompatible job families. *Comput. and Ind. Eng.* 39: 325–335

Baker K R 1999 Heuristic procedures for scheduling job-families with setu-ps and due-dates. *Naval Res. Log.* 46: 978–991

Chandru V, Lee C Y, Uzsoy R 1993 Minimizing total completion time on a batch processing machine. *Int. J. Prod. Res.* 31: 2097–2121

Dobson G, Nambimadom R S 2001 The batch loading and scheduling problem. *Operat. Res.* 49: 52–65

Jolai F 2005 Minimizing number of tardy jobs on a batch processing machine with incompatible job families. *Euro. J. Operat. Res.* 162: 184–190

Fanti M P, Maione B, Piscitelli G, Turchiano B 1996 Heuristic scheduling of jobs on amulti-product batch processing machine. *Int. J. Prod. Res.* 34: 2163–2186

Fowler J W, Hogg G L, Phillips D T 1992 Control of multi product bulk service diffusion/oxidation processes. *IIE Trans.* 24: 84–96

Hung, Yi-Feng 1998 Scheduling of mask shop e-beam writers. *IEEE Transactions on Semiconductor Manufacturing* 11: 165–172

Ikura Y, Gimple M 1986 Efficient scheduling algorithms for a single batch processing machine. *Operations Res. Lett.* 5(2): 61–65

Kempf K G, Uzsoy R, Wang C S 1998 Scheduling a single batch processing machine with secondary resource constraints. *J. Manufacturing Syst.* 17(1): 37–51

Kim H-U, Kim Y-D, Kim J-G 2000 Batching and scheduling at a batch processing workstation in a semiconductor fabrication facility producing multiple product types with distinct due dates. *Proc. Int. Confer. on Modelling and analysis of semiconductor manufacturing* 151–156

Koh S-G, Koo P-H, Ha J-W, Lee W-S 2004 Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job-families. *Int. J. Prod. Res.* 42(19): 4091–4107

Koh S-G, Koo P-H, Kim D-C, Hur W-S 2005 Scheduling a single batch processing machine with arbitrary job-sizes and incompatible job-families. *Int. J. Prod. Econ.* 98: 81–96

Krishnaswamy K N, Raghavendra B G, 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

Law T D, Green R J 1970 Computers for foundry scheduling and production control. *The British Foundryman*, May 1970, 138–153

Law T D *et al* 1983 Outline requirements for a production planning and control system for a foundry, first report of Working Group E7: Production control. *The British Foundryman* 76: 56–59

Law T D *et al* 1985 What is production control? 3rd Report of Working Group E7. *The British Foundryman* 78: 509–510

Law T D *et al* 1988 Foundry production control for the 80s and 90s: Functional overview and database requirements. Fourth Report of Working Group E7. *The British Foundryman* 81: 308–312

Law T D *et al* 1989 Foundry production control for the 80s and 90s: Production planning and scheduling. Fifth Report of Working Group E7. *The British Foundryman* 82: 9–13

Law T D *et al* 1990 Foundry production control for the 80s and 90s: Production monitoring and data capture. Sixth Report of Working Group E7. *The British Foundryman* 83: 306–312

Law T D *et al* 1990 Foundry production control for the 80s and 90s: Management information. Report of Working Group E7. *The British Foundryman* 83: 363–367

Malve S, Uzsoy R 2007 A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job-families. *Comput. & Operations Res.* 34: 3016–3028

Mathirajan M, Chandru V, Krishnaswamy K N 2001 Modified heuristic algorithms for scheduling multiple batch processors with incompatible job families. *Asia-Pacific J. Operat. Res.* 18: 89–102

Mehta S V, Uzsoy R 1998 Minimizing total tardiness on a batch processing machine with incompatible job families. *IIE Trans.* 30: 165–178

Monch L, Balasubramanian H, Fowler J W, Pfund M E 2005 Heuristic scheduling of jobs on parallel bacth machines with incompatible job families and unequal ready times. *IIE. Trans.* 32: 2731–2750

Monch L, Zimmermann J, Otto P 2006 Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. *Eng. Application Artificial Intell.* 19: 235–245

Perez I C, Fowler J W, Carlyle W M 2005 Minimizing total weighted tardiness on a single batch process machine with incompatible job-families. *Comput. & Operations Res.* 32: 327–341

Pinedo M 1995 *Scheduling: Theory, algorithms, and systems*, (New York: Prentice-Hall) USA

Qi X, Tu F 1999 Earliness and tardiness scheduling problems on a batch processor. *Discrete Appl. Math.* 98: 131–145

Ram, Patel 1998 Modelling furnace operations using simulation and heuristics. *Proc. 1988 Winter Simulation Conference.* 957–963

Rardin R L, Uzsoy R 2001 Experimental evaluation of heuristic optimization algorithms: A tutorial. *J. Heur.* 7(3): 261–304

Roberts C A, Dessouky M M, Dessouky Y M 1999 A Virtual plant modeller (for batch-chemical processes. *J. Intelligent Manufacturing* 10: 211–223

Shekar G L 1998 *Planning and scheduling systems for steel casting production-A new paradigm*. Ph.D Dissertation, Department of Management Studies, Indian Institute of Science, Bangalore, India

Southall T, Law T D 1980 Approaches to improved job scheduling in foundries. *The British Foundryman* 73: 287–291

Trinder C V, Moss P 1984 Real-time systems for foundry production control. *The British Foundryman* 77: 429–435

Trinder C V, Watts G A 1973 Production control in the non-ferrous castings industry. *The British Foundryman* 66: 237–241

Uzsoy R 1995 Scheduling batch processing machine with incompatible job-families. *Int. J. Prod. Res.* 33(10): 2685–2708

Uzsoy R *et al* 1992 Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine. *Naval Res. Logistics* 39: 369–388

Voorhis T V, Peters F, Johnson D 2001 Developing software for generating pouring schedules for steel foundries. *Comput. and Ind. Eng.* 39: 219–234

Zee D J, Van Harten A, Schuur P C 1997 Dynamic job assignment Heuristics for multi-server batch operations — A cost based approach. *Int. J. Prod. Res.* 35(11): 3063–3093

Zee D J, van der, Van Harten A, Schuur P C 2001 On-line scheduling of multi-server batch operations. *IIE Trans.* 33: 569–586