A Heuristic Based On Makespan Lower Bounds in Flow Shops with Multiple Processors

Vikram Srinivasan and Daryl L. Santos

Systems Science and Industrial Engineering Department
Binghamton University
4400 Vestal Parkway East
Binghamton, New York 13902-6000

Corresponding author's email: santos@binghamton.edu

Abstract: Minimum makespan scheduling of Flow Shops with Multiple Processors (FSMPs), also known as the Hybrid Flow Shop (HFS), is classified as NP complete. Thus, the FSMP largely depends on strong heuristics to develop solutions to makespan scheduling instances. An FSMP consists of *m* stages wherein each stage has one or more processors through which *n* jobs are scheduled. This paper presents a heuristic based on the lower bound developed in a prior work in order to determine good makespan solutions in the FSMP environment. In the environment studied in this work, the multiple machines available at a particular processing stage are identical processors. In order to evaluate the proposed heuristic, its performance is compared to makespans obtained via the use of modified pure flow shop heuristics. Results show that the proposed heuristic is indeed a strong heuristic for the FSMP and it provides makespans that are better than those provided by some of the already existing pure flow shop heuristics that have been adapted for the FSMP environment.

Keywords: Flow Shop with Multiple Processors, Hybrid Flow Shop, Scheduling, Makespan, Lower Bound, Simulated Annealing.

1. Introduction

The FSMP is considered as one of the toughest problems in the scheduling arena (Pinedo, 2012; Brah, 1988). The FSMP has not been given as much attention in the literature as other scheduling environments although it has broad applications in various industries such as the petrochemical industry, Printed Circuit Board (PCB) assembly industry, automotive industry, space shuttle pre-flight processing and the food and textile industry (Ruiz and Vazquez-Rodriguez, 2010; Edwards and Santos, 2016). A graphical representation of the FSMP is shown in Figure 1.

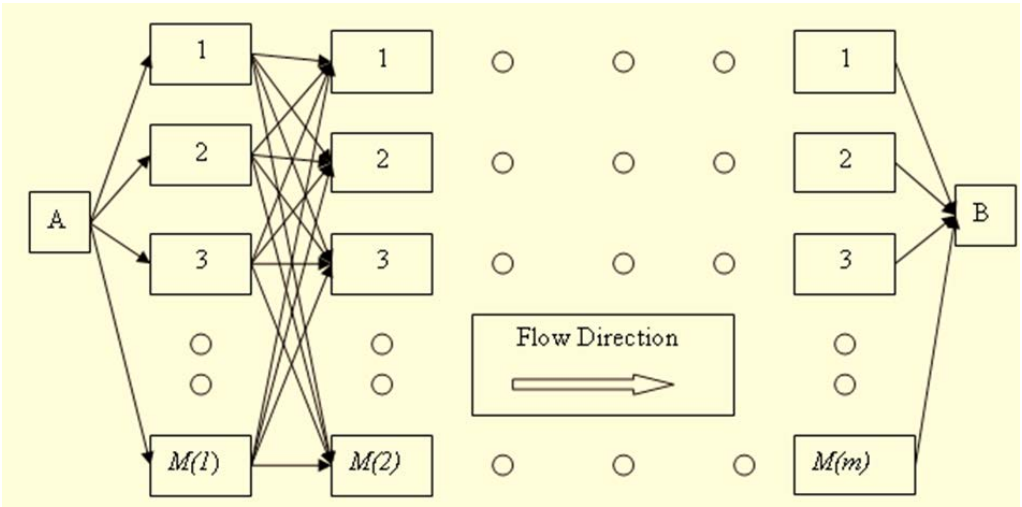

Figure 1. A Graphical Representation of the FSMP

There is a rising need to develop strong and efficient heuristics in order to find near optimum solutions to non-deterministic polynomial time (NP) problems. Finding a polynomial time algorithm to solve such problems is considered impossible (Garey *et al.*, 1979).

The FSMP, also known as the Hybrid Flow Shop (HFS), is classified as NP complete. Although algorithms exist to find optimum solutions for FSMP problems, they are only useful in small problem sizes; these algorithms fail as the problem size (e.g., number of jobs or stages) gets larger. Thus, FSMP problems largely depend on strong heuristics to find good solutions. The term "heuristic" is used herein to represent a step-wise procedure designed to give good, feasible solutions but they are not guaranteed to provide an optimum solution. Apart from depending on heuristics to provide feasible, hopefully near–optimal, and occasionally optimal solutions, feasible FSMP makespan solutions can also be obtained using meta-heuristics. A few examples of meta-heuristics are genetic algorithms and simulated annealing (SA). These techniques are applied when the search space is discrete and they have been applied to FSMP problems in the literature (Ruiz and Maroto, 2004, Younes, Santos and Maria, 1998) and SA is considered later in this work.

The total number of schedules possible for an *n* job *m* stage FSMP problem instance with *M(j)* machines at each stage (*j* varies from 1 to *m*) was developed by Brah (1988) and is shown below:

$$\textbf{Total number of schedules} = \prod_{j=1}^{m} C(n-1, M(j)-1) * {}^{n!}\!/_{M(j)!} \qquad (1)$$

Consider a 5 job, 3 stage problem with 2 identical machines at each stage. Using the above formula, we determine the total number of schedules to be 13,824,000. This shows how complex an FSMP is for even small-sized problems.

Santos, Hunsucker, and Deal (1995a) developed a global lower bound for the makespan given the following: all jobs are ready at the same time in the system; at a particular stage, the multiple processors are identical in speed; and *p(i,j)* represents the processing time of job *i* at stage *j*. The approach involved developing a lower bound for each of the *m* stages along with the obvious lower bound $LB(0) = \max\left\{\sum_{j=1}^{m} p(i,j)\right\}$ (which is based on the fact that all the jobs cannot finish before the total processing time of the job with the longest duration) and choosing the maximum value of the lower bounds computed at each stage (ranging from 1 to *m*).

The formula to determine the lower bound at each stage *j*, denoted by *LB (j)* is shown below.

$$LB(j) = \frac{1}{M(j)}\left(\sum_{y=1}^{M(j)} LSA(y,j) + \sum_{i=1}^{n} p(i,j) + \sum_{y=1}^{M(j)} RSA(y,j)\right) \qquad (2)$$

$RS(i,j) = \sum_{j'=j+1}^{m} p(i,j')$ for $j<m$ and zero if $j=m$ $\qquad$ $LS(i,j) = \sum_{j'=1}^{j-1} p(i,j')$ for j>1 and zero if j=1

For a fixed *j*, *RSA(j)* is the ordered list (in ascending order of magnitude) of *RS(i,j)*-values taken over all jobs. *RSA(1,j)* is the first value in the list and corresponds to *RS(k,j)* for some particular job number, *k*.

$$RSA(j) = \{RSA(1,j), RSA(2,j)....., RSA(n,j)\} \qquad RSA(1,j) \le RSA(2,j)...,\le RSA(n,j)$$
$$LSA(j) = \{LSA(1,j), LSA(2,j)....., LSA(n,j)\} \qquad LSA(1,j) \le LSA(2,j)...,\le LSA(n,j)$$

The lower bound for the entire problem is denoted by *LBMAX* and is computed as follows: $LBMAX = \max\{LB(0), \max_j LB(j)\}$. We will refer to the lower bound generated by this method as the "SHD technique."

In presenting the efficacy of the lower bound, the relative error of the lower bound from the optimal solution was calculated for 653 problems – wherein the optimal solutions were obtained – and the results were desirable. In their findings, 37.8% of the lower bounds calculated for the 653 problems accurately predicted the optimal makespan (Santos, Hunsucker,

and Deal, 1995a). The mean relative error was 4.6%. Thus, the global lower bound is considered to be a very strong lower bound for the FSMP environment.

In the literature, no heuristic has been developed based on the lower bound calculations from that work. Since the mean relative error of the lower bound from the optimal solution in their study was only 4.6%, a heuristic based on the lower bound calculations may indeed be a strong heuristic to minimize the makespan.

2. The FSMP Model

As part of this study, various FSMP configurations are considered by varying the number of jobs, number of stages and the number of processors at each stage. The problems with 5 jobs are classified as small while those with 15 jobs are classified as medium. Problems with more than 15 jobs are classified as large problems. Once the job order from a particular heuristic is obtained, the jobs are sent in the sequence obtained and are then processed FIFO through the rest of the stages.

In order to evaluate the proposed heuristic, already existing heuristics applied to the FSMP in the literature are used. Two very important heuristics that are based on Johnson's Algorithm (1954) but have subsequently been applied to the FSMP (see Santos *et al.*, 1996) are used. These heuristics are the widely known Palmer heuristic (Palmer, 1965) and the heuristic by Campbell *et al.* (1970). In a study by Santos, Hunsucker and Deal (1996), CDS applied to FSMP problems was shown to be a very strong heuristic and found makespans within 15% of the optimal makespan. Palmer's heuristic also performed well in that study.

Apart from these heuristics, two very commonly used dispatching, or priority rules are also used, namely, the Shortest Processing Time (SPT) and the Longest Processing Time (LPT). In the studies by Brah, Santos and Hunsucker (1989), Hunsucker and Shah (1994) and Brah and Wheeler (1998), SPT proves to be the best priority rule for the makespan criteria among the other usual dispatching techniques.

Heuristics developed by Nawaz *et al*. (1983), and Ho and Chang (1991) for the FSMP environment are also very strong and efficient heuristics. Since these heuristics are computationally more complex than heuristics such as the CDS or Palmer's heuristic, they are not used as part of this study to evaluate the proposed heuristic but certainly can be an area for future comparisons. The various job configurations considered as part of this study are shown in Table 1.

Table 1. FSMP Configurations

Jobs	Stages	Machines/Stage	No. of Problems
5	3	2	500
	5	4	500
15	3	2	500
	5	4	500
		10	500
25	3	2	500
	5	4	500
		10	500
		20	500
50	3	2	500
	5	4	500
		10	500
		20	500

The code used to generate the job sequence and makespans for the proposed heuristic and the heuristics used for evaluation is C++. Random numbers for the processing times are generated using the random (int) function available in C++. The processing times take values between 1 and 25 (inclusive). Once a set of processing times is generated, the seed of the random number function is changed in order to obtain different problem sets every time the code is run.

For each of these configurations, the values of makespans from the proposed heuristic and the heuristics used for evaluation are obtained. The relative error of the makespan obtained for each heuristic from the lower bound is calculated from the SHD technique. These values are used to perform the following tests:

1) A general linear model (as shown in Table 2) at 95% confidence interval with 4 fixed factors, namely, the heuristics, number of jobs, number of stages and the number of processors at each stage is developed and a design of experiments (DOE) is applied to it. In this analysis, problems with only 2 and 4 processors (refer to Table 1 for details of the problem configurations) are considered (the reason for limiting the number of processors to 4 in this analysis is provided in Section 4 under Results and Discussion). Tukey's test with a confidence level of 95% is performed to see if there is a significant difference between the makespans obtained from the heuristics considered in this study and to identify the heuristics with significant differences in their means.
2) A one-way ANOVA at 95% confidence interval is performed on problems with more than 4 processors. The reason for sectioning the analysis into a general linear model approach and the one-way ANOVA is discussed in detail in Section 4.
3) A paired t-test at a confidence level of 95% is performed (on those cases for which Tukey's test indicate no significant difference between the makespans of the heuristics) in order to further investigate if there is a significant difference between the means of the proposed heuristic and the other heuristics considered to evaluate it.

Simulated annealing is applied to the proposed heuristic and the already existing heuristics by considering their initial schedules as the initial seed sequence for the simulated annealing procedure. The relative improvement of the makespan after simulated annealing is calculated and analyzed.

3. Algorithm

The basis of the concept described below is obtained by observing problems in which the lower bound equals the optimal solution and trying to identify a possible pattern within these problems. In fact, a few patterns were observed within these problems and it largely depended on the occurrence of the lower bound at a particular stage. The following is a stepwise procedure of the proposed heuristic that will be adapted to the makespan scheduling problem in the FSMP.

STEP 1: The processing times for the problem, denoted by *p[i][j],* are taken as input from the user along with the number of jobs to be processed *(n),* the number of stages *(m)* and the number of identical processors at each stage (*mj[1…m]).*

STEP 2: The lower bound for the given problem is calculated *(lbmax)* and the occurrence of the lower bound *(lbmaxn)* is determined.

STEP 3: If $(1 \le lb \max n \le \frac{m}{2}$ [dls1]) then[VS2] repeat Steps 4 to 10 until *i*<*n* starting from *i=0* incrementing *i* by 1 each time; else go to Step 21.

STEP 4: Repeat Steps 5 to 6 until *j*<*lbmaxn* starting from *j=0* incrementing *j* by 1 each time.

STEP 5: Declare 4 arrays *sum[], suma[], sumo[]* and *po[].* Array *suma[]* stores the sum of the processing times of all jobs passing from the first stage to *lbmaxn.* Array *sum[]* stores the sum of the processing times of all jobs passing from stage *lbmaxn+1* to m. Arrays *sumo[] and po[]* are used to store the job positions arranged in ascending order of *suma[]* and *sum[],* respectively.

STEP 6: Compute *suma[i] = suma[i] +p[i][j] and sumo[i]=i+1.*

STEP 7: End *j* loop.

STEP 8: Repeat Step 9 until *j*<*m* starting from *j=lbmaxn* incrementing *j* by 1each time.

STEP 9: Compute *sum[i] =sum[i] +p[i][j] and po[i]=i+1.*

STEP 10: End *j* loop.

STEP 11: End *i* loop.

STEP 12: Sort *sum* and *suma* in ascending order and sort its corresponding job positions stored in *sumo* and *po,* respectively.

STEP 13: Repeat Steps 14 to 16 until *i*<*mj[lbmaxn-1]* starting from *i=0* incrementing *i* by 1 each time.

STEP 14: Declare *joborder[]* to store the job order for the proposed heuristic and initialize count to *n*-1.

STEP 15: Do *joborder [count] =sumo[i]* and assign a large number say 10000 to *suma[sumo[i]-1]*

STEP 16: Decrease count by 1.

STEP 17: End *i* loop.

STEP 18: Repeat Step 19 until *i*<*n- mj[lbmaxn-1]* starting from *i=0* incrementing *i* by 1 each time.

STEP 19: Assign *po[i]* to *joborder[i]*.
STEP 20: End *i* loop.
STEP 21: If ($lb \max n > \frac{m}{2}$) repeat Steps 22 to 28 until *i<n* starting from *i=0* incrementing *i* by 1 each time; else go to 39
STEP 22: Repeat Steps 23 to 24 until $j \geq 0$ starting from *j=lbmaxn-2* decreasing *j* by 1 each time.
STEP 23: Assign sum[i] and suma[i] to 0.
STEP 24: Compute *sum[i] = sum[i] +p[i][j] and sumo[i]=i+1*.
STEP 25: End *j* loop.
STEP 26: Repeat Step 27 until *j<m* starting from *j=lbmaxn*-1 incrementing *j* by 1 each time.
STEP 27: Compute *suma[i] =suma[i] +p[i][j] and po[i]=i+1*.
STEP 28: End *j* loop.
STEP 29: End *i* loop.
STEP 30: Sort *sum* in ascending order and *suma* in descending order and sort its corresponding job positions stored in *sumo* and *po*, respectively.
STEP 31: Repeat Steps 32 to 34 until *i<mj[lbmaxn-1]* starting from *i=0* incrementing *i* by 1 each time.
STEP 32: Declare *joborder[]* to store the job order for the proposed heuristic and initialize count to 0.
STEP 33: Do *joborder[count] =sumo[i]* and assign 0 to *suma[sumo[i]-1]*
STEP 34: Increase count by 1.
STEP 35: End *i* loop.
STEP 36: Repeat Step 37 until *i<n* starting from *i=mj[lbmaxn-1]* incrementing *i* by 1 each time.
STEP 37: Assign *po[i-mj[lbmaxn-1]]* to *joborder[i]*.
STEP 38: End *i* loop.
STEP 39: If (*lbmaxn=0)* then go to Step 3 treating *lbmaxn* as 1.
STEP 40: STOP.

4. Results and Discussion

The FSMP minimum makespan problem, as stated earlier, is an NP complete problem and determining the optimum solution for most problems is impossible in polynomial time. Hence, in order to evaluate the quality and performance of the proposed heuristic, the SHD technique is used. The Average Relative Error (ARE) or the deviation of the makespan from the lower bound is calculated in percentage form. The ARE is calculated as the average of the relative errors for each problem configuration (mentioned in Table 1). The analysis of the configurations chosen for this study is divided into two sections. In the first section, a DOE of the configurations consisting of problems with only 2 and 4 processors is performed and in the second section, a one-way ANOVA on the remaining configurations is performed. Sectioning the analysis is done as the number of processor levels across the various problem sizes is not the same (we do not consider cases where the number of processors for a given FSMP configuration exceeds the number of jobs to be processed in that configuration) and thus a DOE for all problem sizes cannot be simultaneously performed.

4.1. The General Linear Model

Table 2. The General Linear Model for DOE Analysis

				Processor									Processor			
				2									4			
		Jobs			Jobs					Jobs			Jobs			
	5	15	25	50	5	15	25	50	5	15	25	50	5	15	25	50
		Stages					Stages			Stages					Stages	
	3	3	3	3	5	5	5	5	3	3	3	3	5	5	5	5
Proposed Heuristic	16.04	11.88	8.45	5.34	15.19	20.95	16.28	11.30	0.76	22.46	15.70	9.64	0.48	23.71	25.87	17.65
Palmer	15.75	11.47	6.99	3.63	13.79	18.90	13.39	7.46	2.20	23.64	16.72	9.01	1.60	23.09	24.36	14.64
SPT	21.52	17.67	12.57	7.56	16.94	25.50	21.02	14.49	4.82	33.16	25.44	15.41	2.55	28.26	32.33	22.13
LPT	33.88	39.75	35.26	30.63	26.62	46.01	42.07	36.29	2.69	50.37	46.75	38.47	1.70	43.52	51.63	44.50
CDS	12.10	6.59	3.70	1.57	9.87	14.30	9.70	5.81	0.76	15.62	9.26	3.83	0.29	15.66	16.99	9.96

The general linear model as shown in Table 2 is analyzed using Minitab. A K-S test was performed. The K-S test for normality indicates that the data are normally distributed with a p-value>0.15 (see Figure 2).

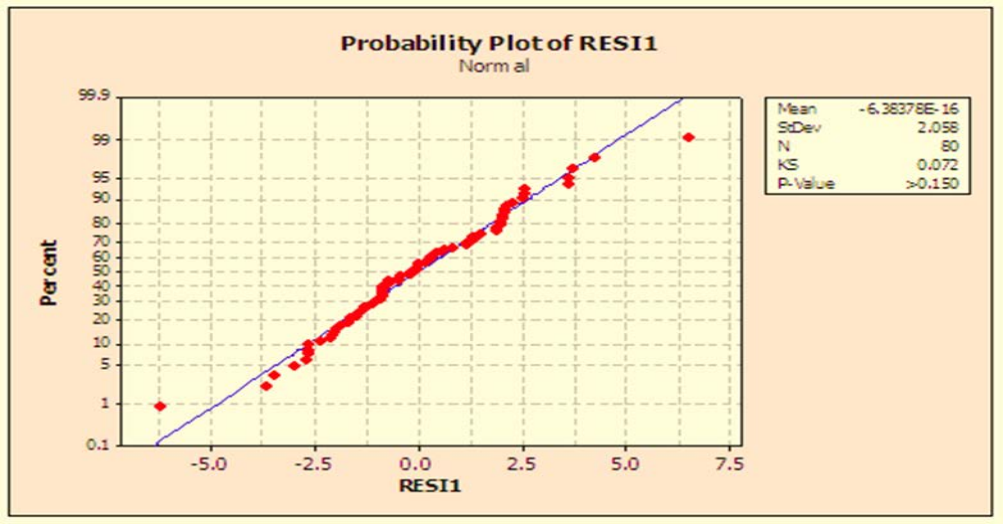

Figure 2. Normal Probability Plot for the General Linear Model

Also, no pattern is observed in the residual plots indicating a constant variance in the data. The model is built considering up to 2 factor interactions between the factors. The p-values obtained indicate that the interactions between the heuristics and number of stages, the heuristics and number of processors, and the number of stages and number of processors are not significant. Also, the number of processors is not significant while the remaining factors are significant. From the interaction plot of the heuristics and number of jobs it is observed that for all heuristics the ARE initially increases as the number of jobs increases from 5 to 15 and then gradually decreases. From the interaction plot (see Figure 3) of the number of jobs and the number of stages it is observed that the ARE increases as the number of stages increases except for small problem sizes where a decrease in ARE values is observed. Tukey's test is performed using Minitab and from the p-values it is observed that there is a significant difference in the ARE between CDS and the remaining heuristics.

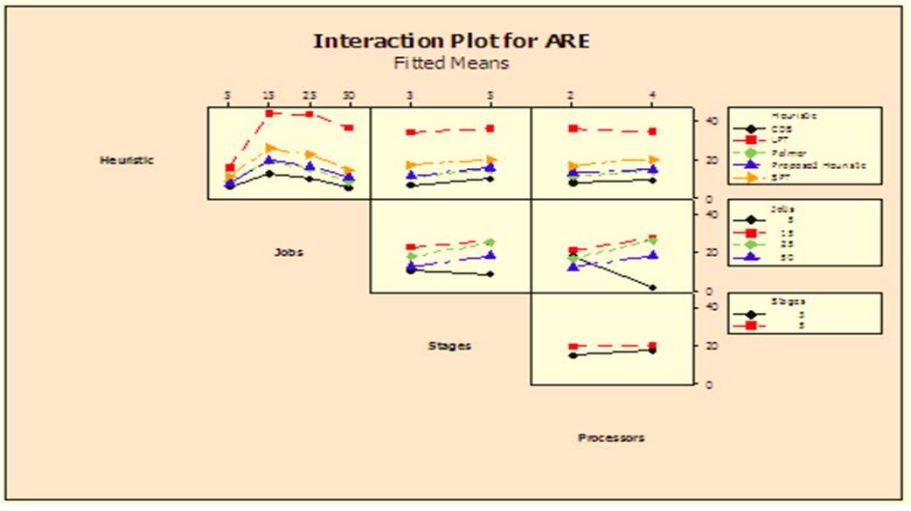

Figure 3. Interaction Plot for General Linear Model

Since the CDS heuristic considers *m-1* job schedules and chooses the best among those schedules, the probability of obtaining a lower ARE is higher in case of the CDS heuristic (Santos *et al.*, 1996). Tukey's test indicates no significant difference in the means of the ARE between the proposed heuristic and Palmer's heuristic (p-value of 0.87). There is a significant difference between the proposed heuristic and both SPT and LPT. The proposed heuristic proves to be better than both SPT and LPT. The worst among all heuristics is the LPT.

4.2. One-way ANOVA

Table 3 shows the ARE for problem configurations with more than 4 processors. A one-way ANOVA is performed on the data using Minitab. The data do not appear to be normally distributed (see Figure 4) and the K-S test for normality indicates that the data are not normally distributed with a p-value=0.037. But since the p-value is close to the alpha value of 0.05, tests that require the data to be normal, such as Tukey's test and the paired t-test, are performed as post hoc tests along with a non-parametric test such as the Mann-Whitney test (that does not assume normality). These tests are performed to investigate any significant differences in the ARE between the heuristics considered in this study.

Table 3. A One-Way ANOVA for Problems with More than 4 Processors

N	M	M(j)	# Problems	Proposed Heuristic	Palmer	SPT	LPT	CDS
15	3	10	500	0.27	3.05	9.08	4.57	0.37
15	5	10	500	0.16	2.30	5.08	4.03	0.06
25	3	10	500	12.69	14.67	25.36	32.84	7.09
25	5	10	500	7.71	9.82	15.42	20.99	3.09
25	3	20	500	0.00	0.58	4.45	0.34	0.00
25	5	20	500	0.00	0.58	2.58	0.52	0.01
50	3	10	500	24.04	25.98	38.85	56.49	13.82
50	5	10	500	30.98	30.23	38.79	54.69	19.31
50	3	20	500	10.88	12.43	25.28	30.02	5.08
50	5	20	500	6.30	8.77	14.95	19.58	2.31
			Average	**9.30**	**10.84**	**17.98**	**22.41**	**5.11**

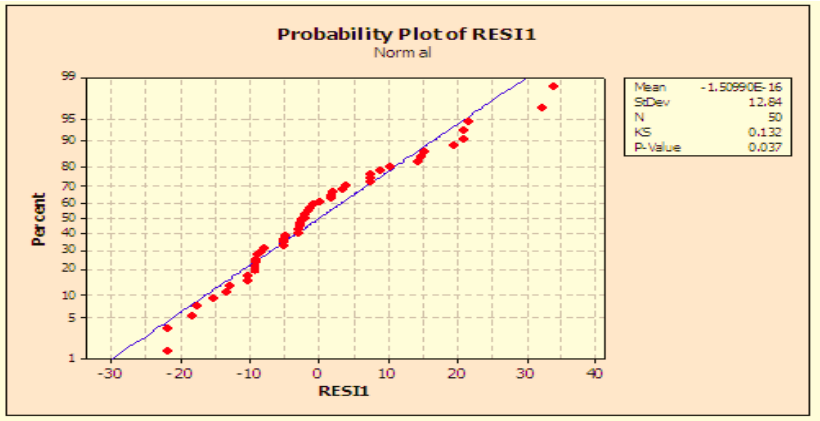

Figure 4. Normal Probability Plot for One-Way ANOVA

- Tukey's test.

From the Tukey's 95% simultaneous confidence intervals obtained from Minitab, a significant difference between only LPT and CDS is observed. There is no significant difference in the ARE between the proposed heuristic, CDS, Palmer and SPT. The pairs for which Tukey's test suggest no significant difference in the ARE were further investigated using the paired t-test.

The ARE values between the proposed heuristic, Palmer, SPT and CDS are further analyzed using a paired t-test for each of the configurations shown in Table 3. The paired t-test indicates that there is a significant difference in the ARE between CDS and the other heuristics namely Palmer, SPT and the proposed heuristic. Since the mean ARE of the CDS is lower than the rest of the heuristics, CDS is the best among all the heuristics.

- Paired t-test

The paired t-test indicates that there is a significant difference in the ARE between the proposed heuristic and Palmer's heuristic for all configurations (except for the 50 job, 5 stage problem with 10 processors at each stage) shown in Table 3. Since the ARE of the proposed heuristic is lower than Palmer's, the proposed heuristic proves to be better than Palmer's heuristic. Also for configurations with 15 jobs, 3 stages and 10 processors in each stage, 15 jobs, 5 stages and 10 processors in each stage, 25 jobs, 3 stages and 20 processors in each stage and 25 jobs, 5 stages and 20 processors in each stage there is no significant difference in the ARE between CDS and the proposed heuristic thus indicating that the proposed heuristic is a strong heuristic. The paired t-test also indicates that the proposed heuristic, CDS and Palmer's heuristic are better (statistically different and have lower ARE) than SPT. The worst among all heuristics is LPT which is consistent with Brah's finding (1989).

- Mann-Whitney test

The results obtained from the Mann-Whitney Test are consistent with the results obtained from the paired t-test. A p-value of less than 0.05 is observed for those pairs of heuristics for which the AREs significantly differ.

4.3. Simulated Annealing

The simulated annealing algorithm developed by Younes et al. (1998) is applied to all the heuristics in this study and the job orders obtained from the heuristics are used as an initial seed sequence for the simulated annealing procedure. For this study, an initial temperature of T=1000 was chosen with a reduction factor of 0.9 and five iterations at each stage for n temperature stages – this is consistent with the SA design by Younes et al. (1998).

The Average Relative Improvement (ARI) is computed for all job configurations with 2 processors (as performing the SA procedure to all the configurations mentioned in Table 1 requires high computational time). The maximum ARI is observed for LPT (as the LPT has the highest ARE values and thus has more scope for improvement). The proposed heuristic responded well to the simulated annealing procedure (although its ARE was relatively low) suggesting a good initial seed sequence for the SA procedure. Tukey's test indicates no significant difference in the means of the ARI between the heuristics except between Palmer's heuristic and LPT. Table 4 shows the ARI and the new ARE values obtained after applying the SA procedure, for each of the heuristics.

Table 4. ARI and New ARE Values for Each Heuristic after Simulated Annealing (only for Problems with 2 Processors)

Heuristic	Proposed Heuristic	Palmer	SPT	LPT	CDS
ARI (%)	7.74	5.70	6.69	11.86	6.84
New ARE (%) after SA	12.16	10.77	16.01	32.01	7.41

4.4. Applying the knowledge of occurrence of the lower bound at a particular stage to the CDS heuristic

As mentioned in Section 4.1, the CDS heuristic generates *m*-1 schedules for a given FSMP problem and chooses the best schedule among those *m*-1 schedules that minimizes the makespan. In this section, the knowledge of the Occurrence of the Lower Bound (OLB) is used to determine if a pattern exists between the best schedule among the *m*-1 schedules obtained from the CDS heuristic and the occurrence of the lower bound at a particular stage. Interestingly, a pattern did exist and a brief description of its algorithm (CDS-OLB algorithm) is described below.

CDS-OLB algorithm

This algorithm is developed for FSMP problems with *m*=3 stages and *m*=5 stages.

STEP 1: If *m*=3 then goto Step 2; else goto Step 3.

STEP 2: If the occurrence of the lower bound is at LB(0), the trivial lower bound, or it is at LB(1), the lower bound as a result of the first stage, or if it is at LB(3), the last stage, then the 2nd sub-problem is chosen from the sub-problems generated by the CDS heuristic and the makespan is calculated. If the occurrence of the lower bound is at Stage 2, then the 1st sub-problem is chosen and the makespan is calculated.

STEP 3: If *m*=5 then goto Step 4; else goto Step 9.

STEP 4: If the occurrence of the lower bound is at Stage 1 then the schedule obtained from the 4th sub-problem is chosen to compute the makespan.

STEP 5: If the occurrence of the lower bound is at Stage 2 or the trivial lower bound LB(0), then the minimum makespan obtained from schedules of the 3rd and 4th sub-problems is chosen.

STEP 6: If the occurrence of the lower bound is at Stage 3 then the minimum makespan obtained from schedules of the 2nd and 3rd sub-problems is chosen.

STEP 7: If the occurrence of the lower bound is at Stage 4 then the minimum makespan obtained from schedules of the 1st and 2nd sub-problems is chosen.

STEP 8: If the occurrence of the lower bound is at Stage 5 then the schedule obtained from the 3rd sub-problem is chosen to compute the makespan.

STEP 9: STOP.

From the 26 FSMP problem configurations considered in this study, 10 problem configurations (which consist of a total of 5,000 FSMP problems) are randomly chosen to study the pattern that exists between the occurrence of the lower bound at a particular stage and the CDS heuristic. Among the 10 randomly chosen FSMP problem configurations, 5 problem configurations were chosen with *m*=3 and 5 problem configurations were chosen with *m*=5. The mean ARE values of the CDS heuristic and the CDS-OLB algorithm described in this section for the 10 FSMP configurations aforementioned are consolidated in Table 5.

Table 5. ARE Values Obtained from the CDS-OLB Algorithm

n	*m*	*M(j)*	ARE of CDS (%)	ARE of CDS-OLB algorithm (%)	% of times CDS-OLB=CDS
25	3	2	3.70	4.21	83.6
50	3	2	1.57	1.80	88
25	3	4	9.26	10.06	79.2
15	3	10	0.37	0.58	94.4
50	3	10	13.82	14.66	85.2
5	5	2	9.87	11.04	71.8
15	5	10	0.06	0.10	98
25	5	10	3.09	3.64	74
50	5	10	19.31	21.14	53.6
50	5	20	2.31	2.71	78.4

As seen from the values in Table 5, the ARE's between the CDS and CDS-OLB are close to each other. The last column in Table 5 represents the percentage of times the CDS-OLB algorithm accurately predicted the minimum makespan of the *m*-1 schedules generated by the CDS heuristic. Except for the 50 job 5 stage problem with 10 identical processors at each stage, all other job configurations performed well on the CDS-OLB algorithm. It is interesting to note that the number of schedules generated by the CDS-OLB algorithm for *m*=3 is only 1 schedule (in general, (*m*-1)/2) unlike the CDS algorithm which generates 2 schedules. Also for *m*=5, the CDS-OLB algorithm generates only 1 schedule for problems in which the lower

bound occurs at the first and last stages and generates 2 (in general, (*m*-1)/2) schedules for all other problems. Thus, with at most half the number of schedules generated by the CDS heuristic for *m*=3 and *m*=5 stage problems, the CDS-OLB heuristic accurately predicts the minimum makespan obtained from the *m*-1 schedules generated by CDS for most of the problems considered.

The CDS-OLB algorithm has good scope of improvement and can be improved to predict the minimum makespan obtained from the CDS heuristic a higher number of times and for all FSMP configurations (with at most half the computation of the CDS heuristic).

5. Conclusions, Recommendations and Future Work

In conclusion, this section describes some of the salient features of the heuristic proposed in this work along with some suggestions for future research. The salient features of the proposed heuristic and the study are summarized as follows:

1. The mean ARE values obtained for the proposed heuristic are low and when compared to other heuristics already being applied to the FSMP, it proves to be better than most of the heuristics (excepting the CDS which indeed is in agreement with Santos, Hunsucker and Deal (1996) as the CDS uses *m*-1 job sequences and chooses the best makespan obtained from those job sequences, giving it a higher probability of obtaining makespans with a lower ARE).
2. The extensive statistical analysis and the application of DOE are very useful in appropriately interpreting the results. This study also uses the paired t-test, Mann-Whitney test and Tukey's test in addition to the ANOVA. This is done as a post hoc test to accurately interpret the results.
3. A large number of problems (500 problems) for each FSMP configuration accumulating to a total of 13,000 problems in this study are considered and thus increasing the validity of the results.
4. The computational efficiency of the heuristic is high as the order of computational complexity is only $O(n^2)$ when compared to the NP-complete nature of the FSMP as discussed in Equation 1.
5. The proposed heuristic has found optimal solutions for certain studies where the lower bound of the problem equals the optimal solution.
6. The DOE analysis sheds light on the significant factors that influence the makespan of a given FSMP problem. Thus, by understanding the effect of significant factors on the makespan, future heuristics can utilize this information to compute makespans with lower ARE values.
7. The proposed heuristic performs well with the simulated annealing procedure. This can be used to further study what makes the proposed heuristic find a good initial seed sequence for the SA procedure.
8. The CDS-OLB algorithm accurately predicts the minimum makespan obtained from the *m*-1 schedules generated by the CDS heuristic by generating at most (*m*-1)/2 schedules for *m*=3 and *m*=5 stage problems.

The following are recommendations and future work opportunities resulting from this effort:

1. The proposed heuristic can be compared with other heuristics that are being applied to the FSMP. Also, various measures of performance (such as the number of tardy jobs, mean flow time, etc.) other than makespan can be used as part of evaluating the heuristic.
2. As mentioned earlier, we did not include other well-known pure flow shop heuristics such as that by Nawaz, Enscore, and Ham (1983) and Ho and Chang (1991). For example, the NEH could be studied for the hybrid flow shop in a similar method as that by Edwards and Santos (2016).
3. Researchers should further investigate the global lower bound for the FSMP and develop new heuristics for the FSMP apart from modifying already existing pure flow shop heuristics and applying them to the FSMP.
4. A study to determine the type of metaheuristic (such as ant colony optimization, particle swarm optimization, etc.) to be applied to a heuristic can be performed.
5. For small and mid-sized FSMP problems, the optimal solution can be computed using the branch and bound method (Brah and Hunsucker, 1991) and the makespan obtained from the heuristic can be compared to the optimal solution instead of the lower bound to see how close the calculated makespan is to the optimal solution of the problem.
6. Although it is preferred to have data following a normal distribution for analysis, there may be instances when the data are not normal (as in Section 4.2). This may be due to a smaller sample size considered (which was the case for the data mentioned in Section 4.2). It would thus be of interest to analyze data with larger sample sizes in order to satisfy the normality assumption that is required to perform tests other than non-parametric tests.

7. The proposed heuristic can be modified to compute makespans for FSMP problems with uniform (non-identical) processors.

The results show that the proposed heuristic is indeed a strong heuristic for the FSMP and proves to be better than most of the heuristics considered in this study excepting the CDS. The main difference between the proposed heuristic and the other heuristics considered in this study is that the proposed heuristic is based on the global lower bounds developed for the FSMP environment by Santos *et al.* (1995a) and the underlying concept of the heuristic is directly related to the FSMP unlike the other heuristics in this study which are essentially pure flow shop heuristics modified to the FSMP environment. We observe that the CDS outperformed all the other heuristics which is indeed in agreement with Santos, Hunsucker and Deal (1996). An important fact to repeat is that for certain job configurations it is observed that there is no significant difference in the means of the ARE between the proposed heuristic and the CDS emphasizing the fact that the proposed heuristic is indeed a strong and effective heuristic. Results herein show promise of pursuing further improvements in developing an FSMP minimum makespan heuristic that leverages the mechanics of the global lower bound developed in Santos *et al*. (1995a).

6. References

Brah S.A. (1988). *Scheduling in a Flow Shop with Multiple Processors.* unpublished Ph.D. dissertation, University of Houston, Houston, TX.

Brah, S. A., Santos, D. L., & Hunsucker, J. L. (1989). Heuristic programming study of a flow shop with multiple processors scheduling. In *ORSA/TIMS Joint National Meeting, New York City*.

Brah, S. A., & Hunsucker, J. L. (1991). Branch and bound algorithm for the flow shop with multiple processors. *European journal of operational research*, *51*(1), 88-99.

Brah, S. A., & Wheeler, G. E. (1998). Comparison of scheduling rules in a flow shop with multiple processors: A simulation. *Simulation*, *71*(5), 302-311.

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management science*, *16*(10), B-630.

Edwards C., & Santos D.L. (2016). Efficacy of the NEH Heuristic in a Hybrid Flow Shop Environment, *Proceedings of the 2016 Annual World Conference of the Society for Industrial and Systems Engineering*, Bay Area, CA, October.

Garey M.R., & Johnson D.S. (1979). Computers and Intractability: A guide to the theory of NP-Completeness. San Francisco, Freeman Press.

Ho, J. C., & Chang, Y. L. (1991). A new heuristic for the n-job, M-machine flow-shop problem. *European Journal of Operational Research*, *52*(2), 194-202.

Hunsucker, J. L., & Shah, J. R. (1994). Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. *European Journal of Operational Research*, *72*(1), 102-114.

Johnson, S. M., (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics (NRL)*, *1*(1), 61-68.

Montgomery D.C. (2006). Design and Analysis of Experiments. 5th edition. ISBN 9971-51-329-3.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Journal of the Operational Research Society*, *16*(1), 101-107.

Pinedo, M. L. (2012). Scheduling: Theory, Algorithms, and Systems. New York: Springer.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1-18.

Santos, D. L., Hunsucker, J. L., & Deal, D. E. (1995). Global lower bounds for flow shops with multiple processors. *European journal of operational research*, *80*(1), 112-120.

Santos, D. L., Hunsucker, J. L., & Deal, D. E. (1996). An evaluation of sequencing heuristics in flow shops with multiple processors. *Computers & Industrial Engineering*, *30*(4), 681-691.

Santos, D. L., Hunsucker, J. L., & Deal, D. E. (2001). On makespan improvement in flow shops with multiple processors. *Production Planning & Control*, *12*(3), 283-295.

Younes N., Santos D.L., & Maria A. (1998). A Simulated Annealing Approach to Scheduling in a Flow Shop with Multiple Processors. *Industrial Engineering Research Conference Proceedings*. Banff, Canada.